Intelligent Menu Planning: Recommending Set of Recipes by Ingredients

Fang-Fei Kuo
Department of Computer Science
National Chiao-Tung University, Hsinchu, Taiwan
ffkuo@cs.nctu.edu.tw

Cheng-Te Li
Graduate Institute of Networking and Multimedia
National Taiwan University, Taipei, Taiwan
d98944005@csie.ntu.edu.tw

Man-Kwan Shan
Department of Computer Science
National Chengchi University, Taipei, Taiwan
mkshan@nccu.edu.tw

Suh-Yin Lee
Department of Computer Science
National Chiao-Tung University, Hsinchu, Taiwan
sylee@cs.nctu.edu.tw

ABSTRACT

With the growth of recipe sharing services, online cooking recipes associated with ingredients and cooking procedures are available. Many recipe sharing sites have devoted to the development of recipe recommendation mechanism. However, there is a need for users to plan menu of meals by ingredients. While most research on food related research has been on recipe recommendation and retrieval, little research has been done on menu planning. In this paper, we investigate an intelligent menu planning mechanism which recommending sets of recipes by user-specified ingredients. Those recipes which are well-accompanied and contain the query ingredients are returned. We propose a graph-based algorithm for menu planning. The proposed approach constructs a recipe graph to capture the co-occurrence relationships between recipes from collection of menus. A menu is generated by approximate Steiner Tree Algorithm on the constructed recipe graph. Evaluation of menu collections from Food.com shows that the proposed approach achieves encouraging results.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications; H 3.3. Information Storage and Retrieval]: Information Search and Retrieval

Keywords

Menu Planning, Recipe Recommendation, Cooking, Ingredient, Minimum Steiner Tree.

1. INTRODUCTION

With the growth of recipe sharing services, online cooking recipes associated with ingredients and cooking procedures are available. Many recipe sharing sites have devoted to the development of recipe recommendation and retrieval mechanism.

However, there is a need for users to plan menu of meals by ingredients. The menu can be used to organize several recipes to make whole meals for daily dinner, holiday events, party planning, and so on. A dinner menu could consist of Chicken Salad as the salad, Coconut Shrimp as the appetizer, Vegetable Lasagna as the main dish, and Blueberry Cake as the dessert. Many recipe sharing websites have provided users the functionality to share menus of recipes and to search for menus by ingredients.

One may come up with an intuitive manner for menu planning: by searching the menu database and exploiting some ingredient matching method, we can easily find those menus possessing similar ingredients as the user-specified ones. However, such method suffers from two points. First, direct ingredient matching will neglect how well the recipes of a menu accompany with each other. While the co-occurrence of recipes in collection of menus provides information about which recipes fit each other, it will be useful to take the collective knowledge about menu plans into consideration. Second, the user-specified ingredients might not be covered by any menus in the database. Instead, a satisfactory menu for such kind of query will be a novel one, which never exactly appears in existing menus. And thus we need to compose a new menu, composed by partial recipes from existing menus.

While most research on food related research has been on recipe recommendation and retrieval, to the best of our knowledge, little research has been done on menu planning. In this paper, we investigate an intelligent menu planning mechanism which recommending sets of recipes by user-specified ingredients. Those recipes which are well-accompanied and contain the query ingredients are returned. We propose a graph-based algorithm for menu planning. The proposed approach constructs a recipe graph to capture the co-occurrence relationships between recipes from collection of menus. A menu is generated by approximate Steiner Tree Algorithm on the constructed recipe graph. Evaluation of menu collections from Food.com shows that the proposed approach achieves encouraging results.

The contributions of this paper are the following. First, we propose the menu planning problem which recommending a set of recipes by ingredients. Second, to generate the set of recipes based on collective knowledge, the recipe graph is developed to capture the accompaniment between recipes from menus of food sharing web sites. Third, we propose to utilize the minimum Steiner Tree algorithm on the constructed recipe graph to solve the menu planning problem and present an approximate Steiner Tree Algorithm for the large recipe graph.

2. RELATED WORK

Recipe recommendation and retrieval has been the subject of cooking related research. One of the earlier works is Kalas, a social navigation system for food recipe, developed by Svensson et al. [11]. Ueda et al. proposed a personalized recipe recommendation method based on user's food preferences [12, 13]. A user's food preference in terms of ingredients is derived from his/her recipe browsing activities and menu planning history. Xie et al. [16] proposed a hybrid semantic item model for recipe search by example. The hybrid semantic item model represents different kinds of features of recipe data. Forbes presents an approach for recipe recommendation to incorporate recipe content into matrix factorization method [1]. Experimental results showed the algorithm not only improves the recommendation accuracy but is also useful for swapping ingredients and creating recipe variations. While most research models recipes in terms of ingredients, Wang et al. [15] model cooking procedures of Chinese recipes as directed graphs and proposed a substructure similarity measurement based on the frequent graph mining.

Another branch of research has focused on the recipe recommendation for healthy food. Mino et al. investigated the recommendation of cooking recipes for a diet in which the evaluation value of intake or consumption of calorie is considered in the events of a user's schedule during the period of a diet [5]. Linear programming approach is utilized with the constraints of carbohydrate, lipid, protein, salt, and increasing the amount of vegetable intake. Karikome and Fujii propose a system to help users for planning nutritionally balanced menus [3]. Considerations of recipes that correct the user’s nutritional imbalance are incorporated into the recipe retrieval process. Visualization of dietary habits are also provided by this system. Shidochi et al. proposed an approach to extract replaceable ingredients from recipes in to satisfy users' various demands, such as calorie constraints and food availability [9]. In order to develop a strategy for changing users eating and cooking behaviors, Pinxteren et al. proposed a user-centered similarity measure for recommendation of healthier alternatives which are perceived to be similar to users commonly selected meals [7]. The similarity measure can be used to promote new recipes that fit users’ lifestyle. By considering the user’s cooking competence, Wagner et al. presented a context-aware recipe retrieval and recommendation system to motivate users for healthy food preparation [14]. The system tracks the user’s cooking activities with sensors in kitchen utensils and recommends healthy recipes that may increase the user’s cooking competence.

While most work on cooking related research focus on recipe recommendation and retrieval, to the best of our knowledge, little work has been done on the menu planning by ingredients.

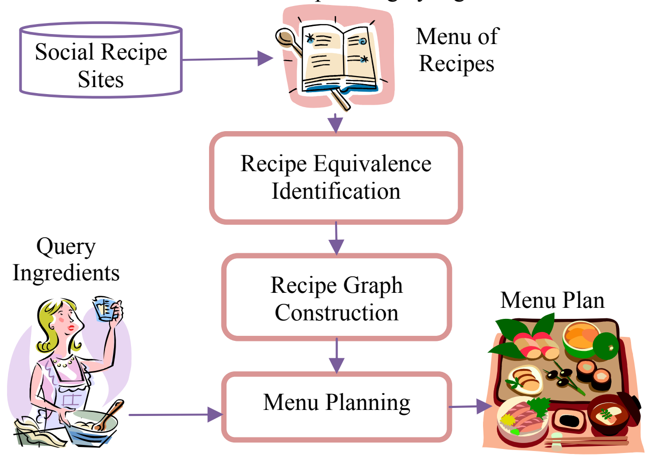


Figure 1. Proposed Framework for Intelligent Menu Planning.

3. PROPOSED FRAMEWORK

The framework of the proposed approach for intelligent menu planning is shown in Figure 1. First, menus of recipes generated by users are collected from social recipe sites such as food.com, allrecipe.com and myrecipes.com. Then, the equivalent recipes are identified. Next, the recipe graph which captures the accompaniment information between recipes is constructed from collected menus of recipes as well as the recipe equivalence information. Finally, given query ingredients, the Menu Planning module with approximate Steiner Tree algorithm on the constructed recipe graph is utilized to generate the menu of recipes satisfying the query ingredients.

Figure 2 shows an example of the recipe graph. The graph consists of eleven nodes; each corresponds to a recipe. For ease of illustration, each node is labeled with recipe name and part of ingredients. There is an edge between two recipe nodes if two recipes appear in the same menu. Each edge is associated with its cost, which will be described in the later section. If two recipes tend to be more fit, the weight of their edge is lower.

4. RECIPE EQUIVALENCE IDENTIFICATION

In the crawled recipes from food.com (details are described in Section 7), since these recipes are manually contributed by users, some recipes, which actually represents for the same one, are considered to be different. To deal with such problem for accurate recipe recommendation, we develop the *recipe equivalence identification* component in our framework. The goal aims at identifying those very similar recipes and regarding them as the same recipe. From the collected data, each recipe is associated with a set of ingredient labels (after some preprocessing on the free texts of ingredient descriptions for each recipe). We consider that if two recipes possess more the same ingredients, they tend to be the equivalence with one another (i.e., have higher potential to be the same recipe). Given two recipes x and y, with the corresponding sets of ingredients S_x and S_y, we use *Jaccard similarity* to measure their extent of equivalence. Specifically, the Jaccard similarity is defined as $J(S_x, S_y) = |S_x \cap S_y|/|S_x \cup S_y|$. If $J(S_x, S_y)$ is higher than a pre-defined threshold τ, the recipes x and y are regarded to be equal one. We apply such similarity computation to all the recipes for aggregating equivalent recipes to the same one. The threshold τ is set to be 0.5 in this work. Note that in fact we can refer to Wang et al.’s sophisticated method [15], which considers the cooking procedure, to measure the similarity between two recipes. Since this is not our main purpose, here we devise the abovementioned simple but effective manner.

5. RECIPE GRAPH

In this section, we give the definition of the recipe graph and the formal definition of the menu planning problem.

[Definition 1] (Recipe Graph) Let $A = \{a_1,..., a_m\}$ be a universe of m ingredients. A recipe graph is defined as an undirected weighted graph $G = (V, E)$. Each node i in $V = \{1,..., n\}$ is a recipe that possesses a set of ingredients T_i □ A. Each edge (i, j) in E is the relationship between two recipes, i and j, and edge weight represents the distance between recipes.

[Definition 2] (Recipe Distance) Given a recipe graph $G = (V, E)$ and a collection of menus, the recipe distance between two recipes i and j is defined as the reciprocal of the number of co-occurrences of recipes i and j within the collection of menus. In other words, if two recipes tend to be co-occurring in menus, the weight of their edge is lower.

[Definition 3] (Menu Cost) Given a recipe graph $G = (V, E)$ the cost of a menu, i.e. a set of recipes, P, P □ V, is defined as the sum of the weights of edges of the minimum spanning tree on the induced subgraph $G[P]$, denoted by $C(P)$.

[Definition 4] (Menu Planning Problem) Given a recipe graph $G = (V, E)$ and a query consisting of a set of query ingredients Q, and the designated number of required courses r, the menu planning problem is to return a menu plan, i.e., a set of recipes, P □ V, $|P| = r$, such that (1) $Q \subseteq \bigcup_{j \in P} T_j$, and (2) the menu cost $C(P)$ is minimized.

Take Figure 2 as an example, the recipe node "Italian Bread" has co-occurrence relationship with five recipes. Among the recipes, Tiramisu has the highest co-occurrence relationship since the cost is the lowest, while Stuffed Shells has the lowest relationship with Italian Bread. For the menu {"Mozzarella, Tomato and Basil Salad", "Lasagna", "Italian Bread"}, the menu cost is 0.23 and the minimum spanning tree is shown with blue color in the graph. In this example, given query ingredients {tomato, flour, basil} (shown in red color on the figure), the Menu Planning module will generate the set of recipes {"Mozzarella, Tomato and Basil Salad", "Lasagna", "Italian Bread"} (nodes with blue frame), rather than {"Mozzarella, Tomato and Basil Salad", "Lasagna", "Almond Cake"}.

6. MENU PLAN GENERATION

To solve the menu planning problem, one approach is to transform this problem into the minimum Steiner tree problem [8]. Given an undirected graph with non-negative costs on edges, and some required nodes, the minimum Steiner tree problem is to find the minimum-cost spanning tree that connect the required nodes. We can transform the menu plan problem into the minimum Steiner tree problem by adding a new node w_j for each ingredients a_j. Each new vertex w_j is connected to a node $i \in V$ if and only if $a_j \in T_i$. In other words, a new ingredient node corresponding to ingredient a_j is connected to each recipe node that possesses ingredient a_j. The distance between an ingredient node and a recipe node is a large number, larger than the sum of all the pairwise distances of the nodes.

However, since the Steiner tree problem is NP-hard, it will take much time to generate the menu when the recipe graph is large. To tackle the large recipe graph, this paper presents an efficient algorithm modified from our previous work on people search in attributed social networks [4]. Our proposed method is an approximation algorithm consisting of three major steps. First, according to the query ingredients, an abstract structure, called *group graph* is extracted from the recipe graph. The group graph will be helpful in reducing the search space to generate the set of recipes. Then, a compact recipe graph, called *query relevance graph* is constructed from the original recipe graph based on the group graph. Finally, we propose a *Connector-Steiner algorithm* to compose the menu from the recipe relevance graph.

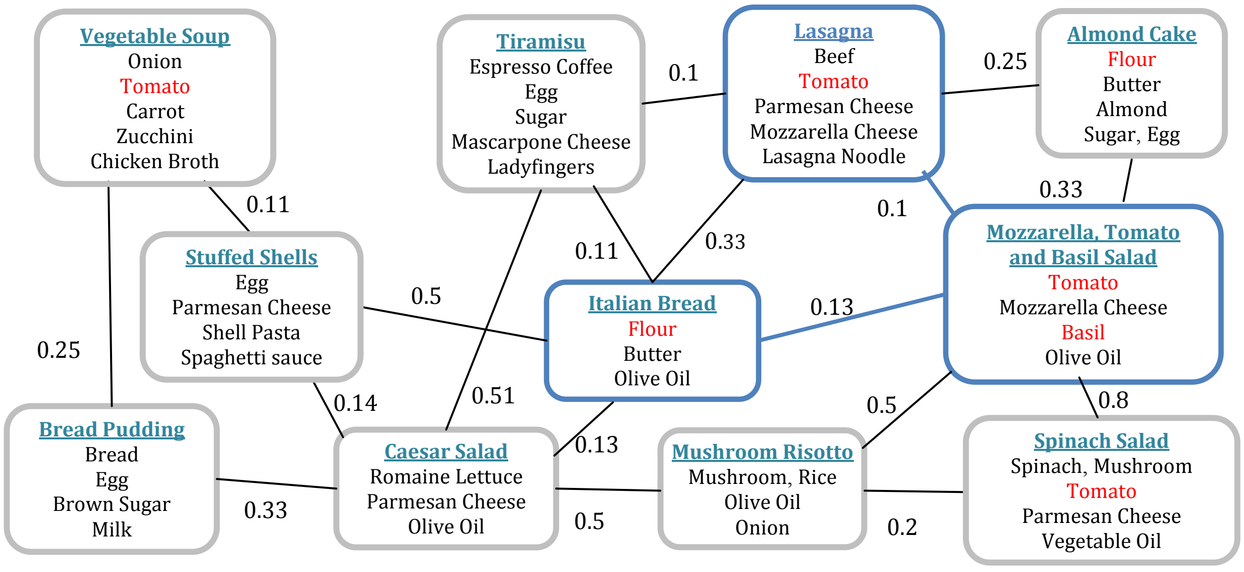


Figure 2. An Example of Recipe Graph for Menu Planning.

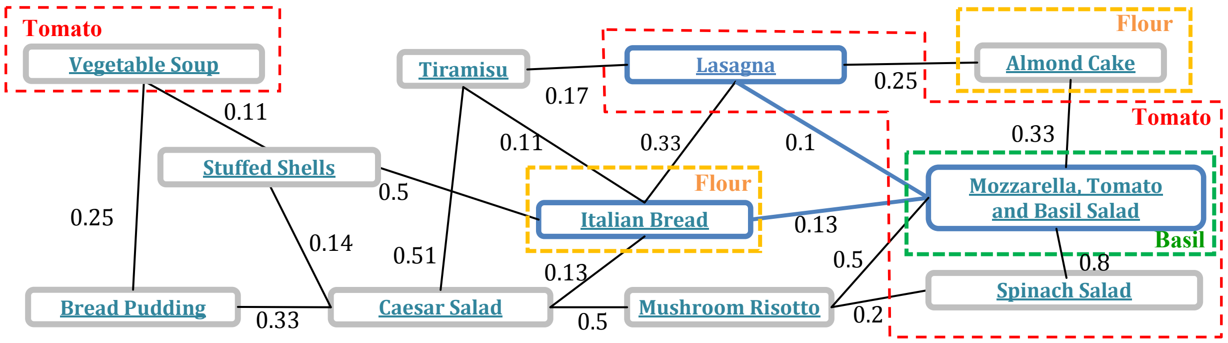


Figure 3. The Query Ingredient Grouping of the Recipe Graph in Figure 2.

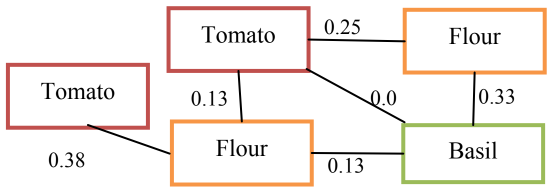


Figure 4. The Group Graph of the Recipe Graph in Figure 2.

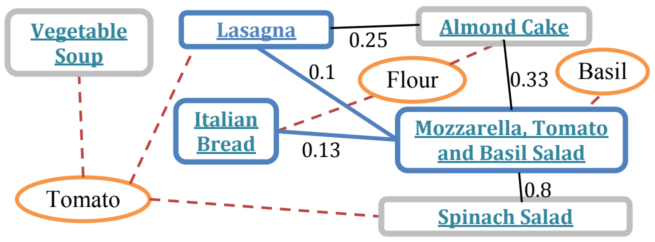


Figure 5. The Query Relevance Graph of the Recipe Graph in Figure 2.

6.1 Group Graph Construction

The first step is to group the nodes in the recipe graph according to query ingredients. A *group*, with respect to a query ingredient, for example, a_i, is a connected subgraph, in which each node contains at least a_i. Note that for a query ingredient a_i, it can have more than one group since there could exist several disconnected subgraphs belonging to a_i. Figure 4 shows the grouping of the recipe graph in Figure 2.

After aggregating nodes into groups, a *group graph* is constructed to model the connections between groups. The idea is that if the group graph possesses lower costs to connect groups, it will guide the menu plan generation to find subgraphs with lower costs. Therefore, given two groups, we leverage the distance measure in single-link clustering to find the path with minimum sum of weights. Specifically, to connect two groups grp_x and grp_y, the *minimum shortest path*, defined by

$$msp(grp_x, grp_y) = argmin_{i \in grp_x, j \in grp_y}\{dist(path_G(i,j))\}$$

where $dist(path_G(i,j))$ computes the distance of the *i-j* path $path_G(i,j)$ in the original recipe graph G. In addition, to eliminate redundant group connections, if nodes in a certain *msp*, except for $i \in grp_x, j \in grp_y$, belong to grp_z, $(z \neq x, z \neq y)$, we will not consider such *msp* into the construction of group graph. Figure 4 shows the constructed group graph of the recipe graph in Figure 2.

6.2 Query Relevance Graph Construction

Taking the group graph as the guidance, the query relevance graph G^R is constructed from the original recipe graph. Such process has three steps. The first is to restore the group nodes. For each group node, we find the induced subgraph in the original recipe graph corresponding to all nodes contained in it. The second is for the group links. For each group link, we find the corresponding path of recipe nodes in the original recipe graph, and embed the path into the induced subgraph derived from the first step. Then for each query ingredient, a new type of node, ingredient node, is added into the relevance graph. A new edge is connected from the ingredient node to each recipe node that possesses that ingredient. For these newly added edges between ingredient nodes and connectors, we associate a positive weight higher than the sum of all weights in the recipe graph. Figure 5 shows the query relevance graph of the recipe graph in Figure 2.

6.3 Connector-Steiner Tree Algorithm

The last step is the Connector-Steiner Tree algorithm to generate the menu plan with lower cost. We start the graph search by selecting an ingredient node from the query relevance graph. The ingredient node with the highest degree will be selected as the seed node. The central idea of the Connect-Steiner Tree algorithm is to consider the replaceability of the three kinds of connectors. Considering now we have the round-*k* resulting subgraphs (i.e., *k* recipes as candidates to be returned), the members of subgraphs could be overlap, direct, and indirect connectors, and now we aim to find the $(k+1)^{th}$ subgraph. We claim the found nodes with the query ingredients in previous *k* rounds are strongly replaceable. We can find other recipes with the query ingredients in the query relevance graph. Therefore we attempt to avoid the discovered node with the query ingredients occurring in the following-round subgraph results. This action will lead the Steiner Tree algorithm to find alternative subgraphs to return alternative recipes with query ingredients in the following effective subgraphs. For those nodes without satisfying the query ingredients, we regard they are weakly replaceable and allow no occurrence restriction on them in the following rounds.

Algorithm 1. Connector-Steiner Tree Algorithm

```
Input: the recipe graph $G^{LR} = (V^{LR}, E^{LR})$;
       a set of query ingredients $T = \{a_1,...,a_r\}$.
Output: A list of menu plans $P$=<$p_1...,p_n$>.
1:  for $k = 1$ to $n$ do
2:      $U_k \leftarrow s$, where $s \in T$ and $s$ is picked by degree.
3:      while $(T \setminus U_k) \neq \phi$ do
4:          $v^* \leftarrow \text{argmin}_{u \in T \setminus Uk} Dist(u, U_k)$ in $G^{LR}$.
5:          if $Path(v^*, U_k) \neq \phi$ then
6:              $U_k \leftarrow U_k \cup \{Path(v^*, U_k)\}$.
7:      $U_k \leftarrow U_k \setminus \{a_1, ..., a_r\}$.
8:      $p \leftarrow \{v_t | v_t \in U_k$ and $t \in X_t$ and $v_t \neq p_j, j=1,...,k-1\}$.
9:      $weight(e_t=(v_t, t)) \leftarrow weight(e_t=(v_t, t)) \times$ LARGEVALUE(=1000).
10:     $LIST \leftarrow (p, Cost(U_k))$.
11: Sort $LIST$ according to $Cost(U_k)$ and return as the ranked list of
    menus $P$=<($p_1$,$Cost_1$),...,($p_n$,$Cost_n$)>, where $Cost_1$<, ..., <$Cost_n$.
```

Here we describe the proposed Connector-Steiner Tree algorithm, as shown in Algorithm 1. For each round, initially, the algorithm starts by selecting a query ingredient node based on the degree from ingredient nodes of the query relevance graph (line 2). Then, each round of the sub-procedure (line 3-6) finds the ingredient node v^* with the mini-mum distance to the set of nodes that have already been added to the current result U_k. All the nodes along the shortest path from this ingredient node to the current solution are added to the current solution set. After the above sub-procedure, we remove the terminal ingredient nodes from $G^{LR}[U_k]$ (line 7). In the end of each round (line 8-10), we identify the recipe v_t with the query ingredient t in current result U_k and associate a larger weight value to the edge et be-tween t and v_t to avoid returning v_t as the following answers. In addition, we also record the current answer p_k and the cost of the current effective subgraph. After the n rounds, we sort the answer recipes according to costs and return a ranked list of answer recipes as the result of menu plan formation (line 11).

7. EVALUATION

In this section, we present the performance evaluation of our solution to the proposed menu planning problem. The aim is to compare the quality between our planned menus (i.e., the recommended set of recipes) and the original recipes in the menus.

7.1 Data Collection and Statistics

We collect the menus and recipes from food.com, which is one of the most popular recipe-sharing websites, where users can create, rate, and share menus and cooking recipes. We downloaded 4,754 menus and 226,025 recipes. Each menu contains a set of recipes. And each recipe consists of a set of ingredients. There are totally 5,073 kinds of ingredients in the all recipes. In addition, each recipe is associated with a set of categorical tags, which describes the course it belongs to, main ingredient, cuisine/region, preparation, occasion, diet, and nutrition for each recipe. Totally there are 524 tags in the data. Table 1 shows some statistics about menus, recipes, ingredients, and tags. We also show the frequency distribution for both the numbers of recipes and ingredients in menus and recipes in Figure 6(a) and 6(b) respectively.

Table 1. Statistics about menus, recipes, ingredients, and tags.

Description	Average Value
# of recipes per menu	6.88
# of ingredients per recipe	8.57
# of tags per recipe	13.66
# of ingredients per menu	42.13

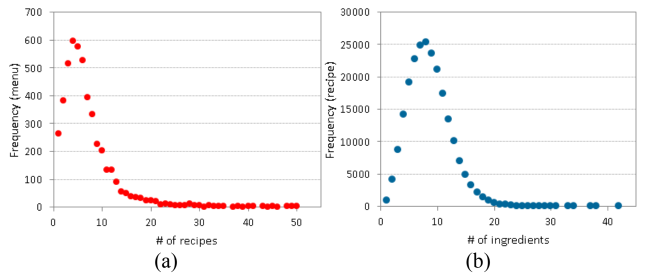


Figure 6. (a) The number of recipes (x-axis) occurs in the number (frequency) of menus (y-axis). (b) The number of ingredients (x-axis) occurs in the number (frequency) of recipes (y-axis).

7.2 Evaluation Setting and Plan

We divide the menus into the training set and the testing set. We randomly choose 70% of menus for training while 30% for testing. All the menus are used to construct the recipe graph, and the weights on edges are computed. The constructed recipe graph contains 20,899 recipes nodes as well as 163,072 edges between them. We use the testing set to generate the query ingredients for the task of menu planning. For each testing menu, the corresponding *original* recipes are considered to be the good-quality ones (i.e., well-accompanied with each other to be a menu). Throughout the evaluation, for each testing menu, we will compute the goodness of both the *planned* set of recipes and the *original* set of recipes for comparison. If the goodness score of a planned set of recipes are close to the scores of the original sets of recipes, the planned menu by our method is considered as an effective and good one. We will show the average goodness scores over the menus in the testing set. We propose two evaluation metrics, *Tag Entropy* (TE) and *Tag Co-occurrence Density* (TCD) described in Section 7.2.1 and 7.2.2, as the goodness measures to understand the quality of the planned menus.

Our evaluation consists of two parts. The **first experiment** aims to understand how the size of a menu (i.e., the number of recipes in a menu) affects the goodness score of the planned set of recipes. We consider each testing menu with k_r recipes to generate a query. If a testing menu has k_i ingredients, all these k_i ingredients are considered as the query ones. We vary the k_r value as $k_r = 4, 6, 8, 10$ to exhibit the effectiveness of our planned results. We will also show the goodness score for the recipes in the testing menu. The **second experiment** is to investigate the effect of the number of query ingredients on the goodness of the planned menu. We fix the number of recipes in a testing menus by setting $k_r = 6$, and vary the number of ingredients that are used as the query ones. For a testing menu containing k_i ingredients, we randomly select $k_i \times X\%$ ingredients to be the query ones. We vary $X\%$ to be 20%, 40%,60%,80%,100% and compute the separated goodness scores.

7.2.1 Tag Entropy (TE)

We propose *Tag Entropy* to capture the tag distribution in a menu. The tag entropy is defined as

$$TE(T_p) = -\sum_{i=1}^{n} Prob(t_i) \log_r Prob(t_i)$$

where T_p is the set of tags in menu p, $Prob(t_i)$ is the probability of tag t_i in the menu. The r and n values are the number of tracks and tags in p respectively. Note that maximum value of the tag entropy is the number of tags n rather than 1. The idea of tag entropy aims to understand the *coherence* of tags in a menu (each menu contain a set of recipes and each recipe is associated with a set of tags). *Coherence* indicates the degree of homogeneity of recipes in a menu. It is noted that the lower the better for the tag entropy of a menu. Since our collected menu dataset is generated by a large number of users, our goal is to recommend sets of recipes with tag entropies close to the average of user generated ones (i.e., recipes in the testing menu).

7.2.2 Tag Co-occurrence Density (TCD)

We consider the co-occurrence relationship between recipes and tags to design the second goodness metric. To evaluate how well the co-occurrence relationship retained in our planned menus, we define a novel measure *Tag Co-occurrence Density*. For any two tags t_i and t_j, which belong to different recipes in the generated menu p, we compute their *co-occurrence* in the training set P_{train}

$$C(t_i, t_j) = \frac{1}{|P_{train}|} \times \sum_{p \in P_{train}} \left(\left(\sum_{\substack{s_1, s_2 \in p \\ s_1 \neq s_2}} \begin{cases} 1, \text{if } t_i \in s_1, t_j \in s_2 \\ 0, \text{otherwise} \end{cases} \right) \Big/ N_p \right)$$

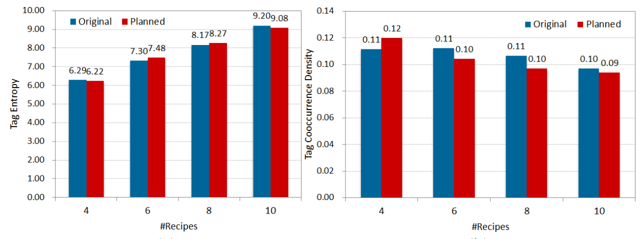
where $s_1$and s_2 are two recipes in menu p, N_p is the total number of pairs of recipes in menu p, and $|P_{train}|$ is number of training menus. Then, we define the tag co-occurrence density of a certain menu p as

$$TCD(p) = \frac{1}{N_t} \times \sum_{\substack{s_1, s_2 \in p, s_1 \neq s_2 \\ t_i \in s_1, t_j \in s_2}} C(t_i, t_j)$$

where N_t is number of tag pairs occurring in different two recipes of menu p. The higher *TCD* value indicates that the recipes in a menu have stronger co-occurrence relationship.

7.3 Experimental Results

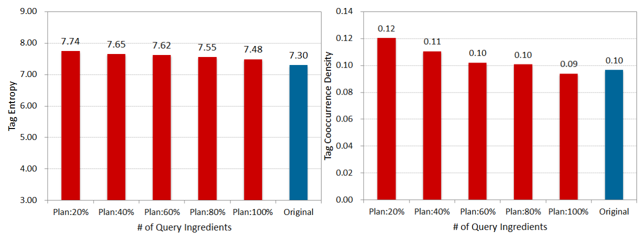
The results of the first experiment (varying the number of recipes of testing menus) are shown in Figure 7(a) and 7(b) for tag entropy and tag co-occurrence density respectively. We can generally find that for both measures, the *planned* menus by our method are very close to the *original* ones. Such results indicate our method is capable of recommending sets of recipes which are well-accompanied with each other, comparing to the user-manually generated menus. We can find that the tag entropy values rise as the number of recipes in the query menu increases. It is because more recipes could include more different tags in a menu. As for the co-occurrence, the values go a bit lower when the number of recipes increases. We think it results from that more recipes could relatively increase the space of tag pairs (i.e., N_t), and then reduce the co-occurrence effects between tags.



(a) (b)

Figure 7. Experimental results by varying the number of recipes for the measures of (a) tag entropy and (b) tag co-occurrence density.

The results of the second experiment (varying the percentage of ingredients used as the query ones) are shown in Figure 8(a) and 8(b). For both measures, we can find values of the planned menus (red ones) are very close the original menus (blue ones). Even though fewer numbers of ingredients (i.e., lower X%) are used for the query, the quality of our recommended sets of recipes are competitive to the original ones. In general when the X% goes lower, which implies less information is provided from the testing menu, the resulting quality will become a little worse (higher tag entropy). However, for the co-occurrence density, fewer numbers of ingredients will result in recommending smaller sets of recipes, which reduces the number of possible tag pairs (i.e., N_t), and thus the *TCD* values will go a bit higher.



(a) (b)

Figure 8. Experimental results by varying the percentage *X*% of ingredients used as a testing query, for the measures of (a) tag entropy and (b) tag co-occurrence density.

8. CONCLUSION

This paper presents an intelligent menu planning mechanism to recommend sets of recipes for any user-specified ingredients. Conceptually, we propose to consider that a good menu should not only satisfy user-required ingredients but also contain recipes which are well-accompanied with each other. Technically, we formulate the menu planning to be an optimization problem in a constructed recipe graph. Experimental results on the Food.com data exhibit the promising quality of the planned menus recommended by our approach.

9. REFERENCES

[1] P. Forbes and M. Zhu, Content-boosted Matrix Factorization for Recommender Systems: Experiments with Recipe Recommendation, ACM International Conference on Recommender Systems RecSys, 2011.

[2] J. Freyne and S. Berkovsky, Intelligent Food Planning: Personalized Recipe Recommendation, ACM International Conference on Intelligent User Interface IUI, 2010.

[3] S. Karikome and A. Fujii, A System for Supporting Dietary Habits: Planning Menus and Visualizing Nutritional Intake Balance, International Conference on Ubiquitous Information Management and Communication ICUIMC, 2010.

[4] C. T. Li, M. K. Shan, and S. D. Lin, Context-based People Search in Attributed Social Networks, ACM International Conference on Information and Knowledge Management CIKM, 2011.

[5] Y. Mino and I. Kobayashi, Recipe Recommendation for a Diet Considering a User's Schedule and the Balance of Nourishment, IEEE International Conference on Intelligent Computing and Intelligent Systems ICIS, 2009.

[6] K. Miyawaki, M. Sano, S. Yonemura, and M. Matsuoka, A Cooking Support System for People with Higher Brain Dysfunction, ACM International Workshop on Multimedia for Cooking and Eating Activities CEA, 2009.

[7] Y. van Pinxteren, G. Geleijnse, and P. Kamsteeg, Deriving A Recipe Similarity Measure for Recommending Healthful Meals, ACM International Conference on Intelligent User Interface IUI, 2011.

[8] G. Reich and P. Widmayer. Beyond Steiner's Problem: A VLSI Oriented Generalization. International Workshop on Graph-theoretic Concepts in Computer Science, 1990.

[9] Y. Shidochi, T. Takahashi, I. Ide, and H. Murase, Finding Replaceable Materials in Cooking Recipe Texts Considering Characteristic Cooking Actions, ACM International Workshop on Multimedia for Cooking and Eating Activities CEA, 2009.

[10] J. Sobecki, E. Babiak, and M. Slanina, Application of Hybrid Recommendation in Web-Based Cooking Assistant, 10th Conference on Knowledge-Based Intelligent Information and Engineering Systems KES, 2006.

[11] M. Svensson, K. HooK, and R. Coster, Designing and Evaluating Kalas: A Social Navigation system for food recipes. ACM Transactions on Computer-Human Interaction Vol. 12, No. 3, 2005.

[12] M. Uede, M. Takahata, and S. Nakajima, User's Food Preference Extraction for Personalized Cooking Recipe Recommendation, 2nd International Workshop on Semantic Personalized Information Management: Retrieval and Recommendation SPIM, 2011.

[13] M. Ueda, M. Takahata, and S. Nakajima, Recipe Recommendation Method Based on User's Food Preferences, IADIS International Conference on e-Society, 2011.

[14] J. Wagner, G. Geleijnse, and A. van Halteren, Guidance and Support for Healthy Food Preparation in an Augmented Kitchen, 2011 Workshop on Context-awareness in Retrieval and Recommendation CaRR 2011.

[15] L. Wang, Q. Li, N. Li, G. Dong, and Y. Yang, Substructure Similarity Measurement in Chinese Recipes, International World Wide Web Conference WWW, 2008.

[16] H. Xie, L. Yu, and Q. Li, A hybrid Semantic Item Model for Recipe Search by Example, IEEE International Symposium on Multimedia, 2010.