Distributed Cooking Recipe Recommendation and Adaptation

Qing Li [1], Wei Chen [2, 1], Lijuan Yu [1]
[1] Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China
[2] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
Email: itqli@cityu.edu.hk, wchen@bit.edu.cn, lijuanyu2@student.cityu.edu.hk

Abstract—In this paper, we propose a distributed recipe recommendation mechanism that utilizes social information for adaptive recipe recommendation in a peer-to-peer (P2P) network. A recipe flavor model is first proposed for modeling recipes and validating recipe adaptations. Peers in the network group themselves into communities in which members share common preferences of recipe data. This is helpful to visit more relevant peers when the query scope is fixed thus improve the performance of recommendation. Based on a graph-based recipe representation, we propose a recipe similarity measure and a filtering algorithm to generate candidates of cooking recipes to be recommended. A recipe adaptation method is also proposed in order to better match users' preferences. Experiments are conducted for the evaluation of the proposed model and a prototype system for cooking recipe recommendation is also presented.

***Index Terms*—recommender system, cooking recipe, distributed information retrieval, recipe adaptation**

I. INTRODUCTION

As more and more multimedia information such as cooking recipes are digitalized into images, videos and texts, how to build an effective and efficient information system that satisfies users' information demand by finding relevant information needed or interested by them becomes a serious issue.

Recommender system is a kind of information filtering technology. It attempts to solve the problem by predicting users' preferences and presenting them items that they are likely to be interested in.

Cooking is important in our lives. However, information technologies in the cooking recipe domain have not been extensively studied [1]. One of the characteristic of cooking recipes is that recipe data are often naturally distributed over a network of autonomous computer systems. Therefore, distributed cooking recipe recommendation has attracted researchers' attentions recently.

Message routing is an important issue in distributed information systems. Compared with breadth-first search in the network, retrieval efficiency can be greatly improved if peers which most probably contain relevant data are visited first. The assumption is that every peer has its own topics of interest. These interested topics are the reflection of the interests of the user behind the peer. A user tends to collect recipe data of similar topics. Therefore the recipe database of the corresponding peer will contain more data on those topics.

In this paper, we propose a distributed recipe recommendation mechanism that utilizes social information for adaptive recommendation, which strives to achieve the following criteria: (1) the user should be interested in the recommended recipes; and (2) the user should be able to utilize (e.g., cook) the recipes.

Another assumption is that only the basic recipes available in the network may not be sufficient to suit diverse users' tastes/preferences. Therefore, based on our proposed recipe flavor model, we also propose to derive new recipes based on existing ones from the following two aspects: (1) ingredient, and (2) cooking styles/patterns.

The rest of this paper is organized as follows. Section II presents the related work. The flavor-based recipe model is introduced in Section III. Section IV presents our community-based recipe recommendation mechanism. A graph-based recipe filtering and adaptation approach is proposed in Section V. Simulation experiments with a multi-agent system and a prototype recommender system based on the proposed approach are shown in Section VI. We conclude our work in Section VII.

II. RELATED WORK

Due to the large volume of data involved in multimedia processing and the fact that data are often naturally distributed over a network, distributed retrieval techniques have long been studied. In [13], the authors propose a hybrid P2P network structure. Directory nodes are employed to provide regional directory services. Content models of neighboring nodes are constructed and used for routing control in the resource selection. Leaf nodes are utilized to perform content-based retrieval. In [4], the authors propose a P2P routing mechanism over a network in which peers are clustered based on visual content similarity. Similarly, the authors of [3] develop a distributed retrieval system that utilizes a routing scheme based on peer clustering. Each peer records links of most relevant peers, known as attractive links, as well as some random links. Queries that are relevant to the database of a peer get propagated through its attractive links, while irrelevant ones go through its random links. A problem is that the content similarity between the query and the summary of the peer has to be computed before the decision of routing can be made, which may cause efficiency problems.

Social information has been used for multimedia data processing. In [14], the authors propose an event annotation mechanism. Peer annotators are utilized in a social network for the annotation. An annotator trust model is also proposed in which trust can propagate in the network according to a link-based ranking algorithm.

Social information embodied by communities has been used to facilitate information retrieval. Members in a community share common interests. Communities of like-minded individuals provide a backdrop for personalized social applications [15].

The work described in [5] and [6] concerns community construction in P2P networks. When proposing their community formation method, the authors of [5] introduce four sources of self-construction of P2P communities, namely, ontology matching, attribute similarity, trust, and link analysis. In [6], the authors propose a mechanism to form communities for sharing academic papers, in which each peer computes its trust on the peers with whom it interacts.

Recommender system has become an indispensable technique in the information filtering field recent years. Large amounts of recommendation approaches have been proposed by researchers, which can be identified into two broad categories, content-based and collaborative filtering. Content-based method makes recommendations based on assumption that users would probably have interest on items with similar features to the ones they have enjoyed. So the key issue in this method is to predict the content similarity between items [10]. Differently, collaborative filtering focuses on finding out relevant *neighbors* for a user from the whole user community, and predicts the target user's preference of items based on his/her neighbors' ratings on them. In the recent work [11][12], the authors proposed algorithms named Random Walk and Social Trust Ensemble, both are applied to the social network to identify a *neighborhood* that can be trusted by the target user, and to what degree can the user trust on these *friends*.

Several pieces of work have been done for information retrieval in the cooking recipe domain. M. Svensson *et al.* apply their idea of social navigation to allow users to share their recipe collections in [9]. Specifically, they assign users to groups based on their explicit preference information, like ingredients, fat level, time to cook, etc. Sub-structure similarity has also been studied between cooking graphs for searching [2]. The cooking procedure of a recipe is modeled as a cooking graph. Given a query example, recipes with similar cooking procedure and ingredients can be retrieved according to the graph similarity.

III. COOKING RECIPE DATA MODELING

In this section, we will present our recipe data modeling strategy, that is how to represent the cooking and flavor features of recipes.

A. Cooking Procedure Modeling

Cooking is a procedure of applying heat to food ingredients in an ordered manner to alter the flavor, appearance, texture and digestibility of the original ingredients, so as to produce edible and delicious dish. The involved elements in the procedure include ingredients, seasonings, cooking patterns, etc., and the relationship between them can be represented as in Fig. 1:

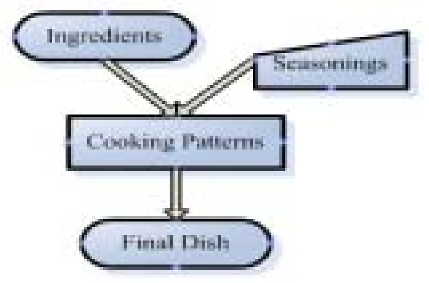


Figure 1. The general components of a cooking activity

Compared to other type of data, cooking recipes have their distinct characteristics: loosely structured, behavior oriented, and constraints bound [1]. Because of the rich content and complex relations between cooking actions, there are few good models proposed for recipe data management. To the best of our knowledge, our previous work—cooking graph model—is the only research in this domain. Here, we review briefly this model for recipe representation. Typically, the cooking procedure of a recipe, denoted as *CG*, is formulated by a graph of 4-element tuple:

$$CG= (V, E, Cons, Ingr) \qquad (1)$$

where V is a set of ingredient or action nodes, E is a set of edges between two nodes, *Cons* is a set of constraints associated with either action nodes or edges, and *Ingr* is the required ingredients to cook the dish. As an example, the graphical presentation of the recipe "Triple Cheese Pasta Primavera" is shown in Fig. 2.

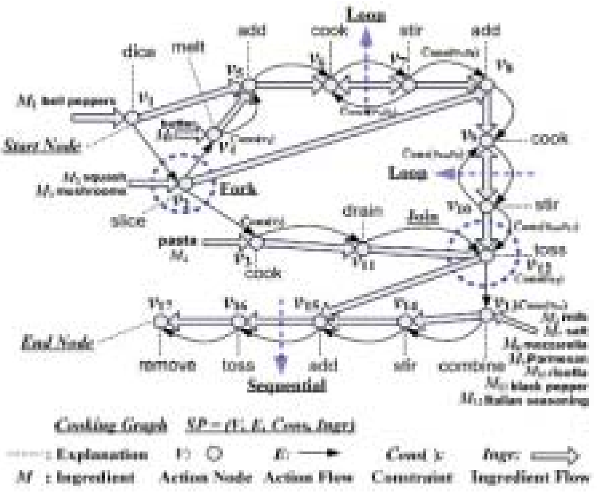


Figure 2. Cooking graph of "Triple Cheese Pasta Primavera"

With the cooking graph, the cooking information of recipes can be well represented in a computer-readable way, thus facilitate users' requirements towards this data.

B. Flavor Features Modeling

Besides the cooking procedure, another aspect users would concern most about a recipe is the flavor features of its final dish. Here we define the flavor features as the properties of a dish that can be directly perceived by human's physical receptors, such as tongues, noses, etc. Basically, the main factors that would influence a person's feeling towards foods can be identified into the following categories: taste, smell, texture, temperature, etc.

Taste: According to theories in the cooking science domain, taste is a combination of five basic elements, respectively sweet, salty, sour, bitter, and umami (also described as savoriness).

Smell: Smell means the odor of food that can be perceived by human's sensing cells in noses. For example, the smell of a dish can be pleasant, offensive, disgusting, etc.

Texture: Texture refers to the physical structure of foods, turning out to be people's feelings when biting it. For example, the texture of a dish could be tender, hard, crisp, etc.

Temperature: Intuitively, temperature describes whether some food is cold or hot.

As a result, we represent the flavor features of a recipe R_i, denoted as $Flv\ (R_i)$, using a feature vector as shown in Table 1:

$$Flv_i = [sweet{:}v_{i,1},\ salty{:}v_{i,2},\ sour{:}v_{i,3},\ bitter{:}v_{i,4},\ umami{:}v_{i,5},\ smell{:}v_{i,6},\ texture{:}v_{i,7},\ temperature{:}v_{i,8}] \qquad (2)$$

where $v_{i,j}$ means the value for each feature. The value types here can be defined in multiple ways, such as numeric values, value ranges, or semantic terms, etc., as long as it is able to represent the features properly and differentiate one recipe from others.

Table 1. Flavor features of recipes.

	Taste					Smell	Texture	Temperature
	Sweet	Salty	Sour	Bitter	Umami			
R_i	$v_{i,1}$	$v_{i,2}$	$v_{i,3}$	$v_{i,4}$	$v_{i,5}$	$v_{i,6}$	$v_{i,7}$	$v_{i,8}$

In conclusion, the overall profile of a recipe R_i in our system is represented by two parts, a cooking graph CG_i and a flavor vector Flv_i:

$$R_i=\{CG_i,\ Flv_i\} \qquad (3)$$

where CG_i is used to present the cooking procedure of the recipe, and Flv_i describes the flavor features of the recipe.

IV. COMMUNITY-BASED RECOMMENDATION

In this section we introduce a community-based algorithm to acquire a candidate recipe set for recommendation which, for our purpose, satisfies the first criterion mentioned, namely, users' interests.

A. Network Structure

The distributed recipe recommendation system consists of a set of computer systems that form a P2P network. Each computer in the network has a recipe database installed and hosts some recipe data in which the owner of the database is interested. The data in the recipe database can be queried by the owner as well as other peers in the network.

The network in the proposed model can be represented as a direct graph defined as follows:

Definition 1 The network of the distributed environment is defined as a directed graph $G = (P, E)$, where $P = \{p_1, p_2, ..., p_n\}$ is a set of nodes, each of which represents a peer database, and $E \subseteq P \times P$ is a set of edges, each of which represents a direct link from one peer to another.

The network is considered as a social network where a peer p_i is an actor and an edge $p_i \rightarrow p_j$ is a relation from p_i to p_j.

In the recipe recommendation model, each peer only maintains direct links to a small number of other peers in the network. A peer gets connected to the network by finding an existing peer and establishing direct connection with it. The neighborhood relations are evolved from time to time according to the previous interactions and experience with other peers in the network.

Social network analysis methods are utilized to facilitate distributed recipe recommendation in our system. Peers combine and group into communities in the network, as shown in Fig. 3. The following definition defines a peer community in our model:

Definition 2 A peer community is defined as a 2-tuple $C = (P^C, T)$, where

- $P^C \subseteq P$ is a set of peers in the network that are the members of a community C;
- T is the preference that all the members of community C shares.

A community in our model has the following constraints:

1. There can only be at most N_{mem} members in a community C;
2. A peer p can only belong to at most N_{com} communities;
3. A peer p has direct connections with the members of all the communities that it belongs to;
4. A peer p may have direct connections with other peers that are not in any community that p belongs to.

Peers in a community share a common interest. Compared with individual links between peers in the network, the use of communities enhances the information exchange between like-minded peers. This can be very useful in our distributed recommendation of recipe data.

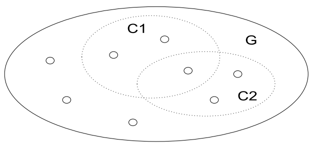


Figure 3. Community-based P2P network structure

B. Distributed Reputation Model

In a distributed system, peers have recipe data that are of different flavors. Their recommendation qualities may also be different because of the different recommender systems. Correctly distinguishing among different qualities of services in a distributed system is useful for resource selection in the system therefore is helpful for improving the overall service quality of the system.

Reputation can be used for collecting evidence of interaction partners, thus useful for distinguishing qualities of services. A distributed reputation model is proposed for establishing trust in the recipe database environment of our system. Reputation can be handled at three different levels, individual level, social level, and system level [16]. The individual level is where a peer is considering only its direct interaction with the other peers to establish reputations. If the peer also uses the information coming from other peers and the social relations, it is the social level. The system level means that evidence is collected and reputations are provided by a central authority.

We adopt this idea and build our distributed reputation system based on the 3-level structure. Support for community-level information is also added to achieve better topic-aware reputation computation.

1) Individual-Level Reputation

A peer interacts with other peers in the network to accomplish the task of recipe recommendation. The interactions form relations between peers that can be used as evidence of quality of services and thus the degree of trustworthy of the peers.

The computation of reputation at the individual level is carried out by individual peers in the system.

The set of all the relations $v \rightarrow b$ occurring between time t_i and t_{i+1} with a specified type τ is defined as follows:

$$\begin{aligned}\Phi_v(b,\tau) = \{v \rightarrow b \mid v \in P \wedge v \rightarrow b \in E \\ \wedge\, typeof(v \rightarrow b) = \tau \\ \wedge\, t_i < timeof(v \rightarrow b) < t_{i+1}\}\end{aligned} \qquad (4)$$

where $timeof(v \rightarrow b)$ is the time that relation $v \rightarrow b$ forms.

The influence of all the relations $v \rightarrow b$ occurring between time t_i and t_{i+1} on the reputation of peer b from v's point of view is defined as follows:

$$\Delta R_{v \xrightarrow{I} b}(t_{i+1}) = \sum_{\tau} (|\Phi_v(b,\tau)| \times w_\tau) \qquad (5)$$

where τ is a type of relations occurring from v to b between time t_i and t_{i+1}, and w_τ is the weight of relation type τ.

In our distributed recipe recommendation model, $\tau = typeof(v \rightarrow b)$ is defined as the relevancy judgment, made by the rating module of v, about the results of recipe data returned to v from b in reply to v's requests. This represents the filtering capability of b from v's point of view. In this case, w_τ is positive for a relevant result and negative for an irrelevant one.

An individual reputation database is used by peers to store individual-level reputations for their direct neighbors. The peers in the network update their individual reputation databases regularly using their direct knowledge to their neighbors. Reputations at the individual level are considered as the most reliable source of trust.

2) Social-Level Reputation

Only perceiving direct interactions can pose a number of problems. For example, in an open system, it can be very difficult for an autonomous peer to select an interaction partner if the peer itself has no sufficient interaction history with other parties.

In the case of lacking historical evidence, a peer has to ask others for advice in order to determine whether a given peer is trustworthy or not. Therefore, a social network analysis approach can be utilized for the estimation of peer reputations in the system. In social network analysis, relational data are represented using a graph called *sociogram*, which is directed and weighted. Each peer is represented as a node and each relation is represented as an edge. The value of a node represents its importance, which is an important factor that forms the corresponding peer's reputation.

A peer v periodically sends a message *Request for Reputation* to all its neighbors requesting reputation references of other peers in the network, especially to those that are in the same communities as *v*. The requesting message propagates in the network with a time-to-live (TTL) value.

In online trade systems, reputation values can affect the selection of transaction partners. Therefore, peers are sometimes motivated to hide information or provide false information about reputations to others. In a distributed recipe recommendation system, it is less likely that this will happen. However, the reputation references from other peers should not be fully trusted, since some of them can still be inaccurate due to various reasons.

The influence of a reputation reference from another peer *x* on the reputation of a target peer *b* at time t_{i+1} from *v*'s point of view is defined as follows:

$$R^{x}_{v\xrightarrow{S}b}(t_{i+1}) = \frac{R_{x\to b}(t_i)}{1+e^{-(R_{v\to x}(t_i)-R_0)}} \quad (6)$$

where $R_{v\to x}(t_i)$ is the reputation of *x* at time t_i from *v*'s point of view, and R_0 is a parameter adjusted to identify the reputations of average peers.

The point is that peers who have higher reputations should affect other peers' reputations to a greater extent. The parameter R_0 in the above formula is used to adjust how much a peer should affect others, and it is set to represent the reputations of average peers. In the formula, peer *v* receives a reputation reference about *b* from peer *x*. If the reputation of peer *x* is much higher than R_0, the result $R^{x}_{v\xrightarrow{S}b}(t_{i+1})$ is nearly $R_{x\to b}(t_i)$, which indicates that the influence caused by this reputation reference is nearly fully charged on *b*. When the reputation of *x* is much smaller than R_0, the result is nearly 0, which indicates that peers with very low reputations can hardly affect others by a reputation reference.

The social-level reputation of peer *b* from peer *v*'s point of view is the aggregation of all the reputation references acquired, defined as follows:

$$R_{v\xrightarrow{S}b}(t) = \sum_{x} R^{x}_{v\xrightarrow{S}b}(t) \quad (7)$$

The reputation acquired from social references is not as reliable as that from direct experiences. It is only used as a complementary method in case there are no direct interactions to determine the degree of trustworthiness, as shown below:

$$R_{v\to b}(t) = \begin{cases} R_{v\xrightarrow{I}b}(t)\times w_I & ,\text{if } \left|\bigcup_{\tau}\Phi_v(b,\tau)\right| > 0 \\ R_{v\xrightarrow{S}b}(t)\times w_S & ,\text{otherwise} \end{cases} \quad (8)$$

where w_I and w_S are the normalizing factors of individual level and social-level reputations, respectively.

When there are direct interactions from *v* and *b*, instead of assigning social-level reputations a small weight, we choose to solely use individual-level ones. This can save the bandwidth and CPU time used to acquire social reputation references.

The combined reputations are used in the recipe recommendation model for recipe recommendation request routing. If from *v*'s point of view there is no evidence about the reputation of *b* from either the individual or social levels, $R_{v\to b}(t)$ is considered as *not available*, and will not be used for modeling trust in our system.

3) Community-Level Reputation

Individual-level and social-level reputations are representations of trust from an individual's point of view. As we have mentioned before, system-level reputation is an aggregated view of the degree of trustworthiness of a peer.

However, in a distributed recipe recommendation system, different peers hold recipe data of different classes. A specific peer can perform well on one class while poorly on another. Therefore, a single central reputation value cannot truly represent the capability of recipe recommendation. Furthermore, the special peer that acts as the central reputation authority can be a bottleneck and can potentially cause single point of failure in the system.

Due to these reasons, we propose a community-level reputation mechanism that can instead represent the common view of trustworthiness in a community, where all the peers share the same interest and have recipe data on the same class. This can greatly improve the performance of recipe recommendation. Furthermore, limiting the scope of central reputation in a community can be helpful to avoid single point of failure of the whole system.

The community-level reputation of a peer *b* in community *C* is calculated by summing up the reputation knowledge from all the members of the community:

$$R_{C\xrightarrow{C}b}(t) = \sum_{v\in C} R_{v\to b}(t) \quad (9)$$

The community-level reputation acquired will further be used in the self-evolving communities in our model.

C. Community Construction

Community plays an important role in the distributed recipe recommendation model. A community is a group of peers that share a common interest, thus have recipe data of a common class. The proposed community-based routing scheme eliminates the need of generating peer summaries as well as the global clustering of peers, which are problematic as mentioned before.

In our work, we propose a distributed community construction method that uses peer reputations as a clue to group peers that have common preferences. This is based on the assumption that *the information retrieval performance will be better if the sources contain more data of the relevant class*. The community construction workflow is as follows.

Community Construction Workflow

Process 1. Community Creation

Communities can be created by any peer in the system. Peer p_0 is given a *low* probability to initiate a new community if and only if the number of communities that p_0 belongs to keeps being lower than $\frac{N_{com}}{2}$ for at least a certain period of time.

In the case that p_0 creates a new community C, p_0 becomes the owner of the community. The owner of a community maintains the main copy of the list of the members of the community. The list is broadcast in the community so that other peers can know the members in the community. The owner is in charge of the management of the membership. It also serves as the *reputation center* that maintains the *community-level reputations* of the members.

Process 2. Member Invitation

Any member of the community $p_i \in C$ can send invitations to recruit new members to join the community. The invitation contains the identities of the owner and other members of the community. The invitation message is sent to the direct neighbors of p_i that are not current members of C. A receiving party can decide whether to apply for membership or not.

Process 3. Application for Membership

A peer $p \notin C$ that receives a member invitation from community C can decide whether the community is related to one of its preferred class or not. It investigates the combined reputation of individual level and social level, $R_{p \to x}(t)$, of the owner and other members of the community. Their reputations are aggregated using a weighted average, where the owner is assigned with a larger weight. The aggregated reputation is then compared with the average reputation of all the peers in the network. Note that $R_{p \to x}(t)$ values are only counted when they are *available*.

If the aggregated reputation of the community owner and other members are much higher than the average value of all the peers in the network, the community is considered as *relevant* to itself from p's point of view. In this case, p will send an *Application for Membership* message to the owner of the community to apply to become a member of it.

Process 4. Member Acceptance

Upon receiving an application from peer p requesting to join the community, the owner p_0 will decide whether to approve the application, i.e., p will become a member of the community, or not. As mentioned, each community can have N_{mem} members at most. The application cannot be approved if the community is full. The decision is made based on the investigation whether the preferred class of recipes of the applicant is relevant or not, from the point of view of the community.

Community-level reputation of the applicant $R_{C \xrightarrow{c} p}(t)$ is used for the measure of relevancy. If the reputation of the applicant is much higher than the average reputation of all the peers in the network (where available), the applicant p is granted with membership of the community C.

Process 5. Leaving a Community

The leaving of a member can be initiated by either the member itself or by the owner peer of the community.

A peer p periodically checks the performance of the communities that it is in. If the average recommendation performance of community C, represented by the reputations $R_{p \to x}(t)$ where $x \in C$, keeps being lower than the average value of all the peers whose reputations are available to p for a certain period of time, p can conclude that the preference of C is no longer of its interest. Peer p will leave the community in this case.

Similarly, the owner of a community keeps checking the performance of its members. If it finds that the performance of a member p, represented by the community-level reputation $R_{C \xrightarrow{c} p}(t)$, keeps being lower than the average value of all the peers in the network (where available) for a certain period of time, it can conclude that p is no longer interested in the class of the community. Thus, p will be excluded from the community.

Process 6. Termination of a Community

The termination of a community is intuitive. When the owner of the community becomes not available, e.g., is down, and its members cannot get access to it for at least a certain period of time, the community will be marked as dead.

A community constructed in this way contains a group of peers that share interests of a common preference. Due to that in an open network of large scale, there may be too many peers that share the same interest (more than the number for which common recipe recommendation tasks can be handled well), the scale of a community is kept limited. There may be communities that share the same or similar preferences, and a certain peer can be a member of more than one community. These peers serve as bridges between communities that share similar interests and play an important role in the network. Communities are handled in a distributed manner so that the termination of a certain community will not affect the function of the whole system.

D. Distributed Recipe Recommendation

1) User Profile Modeling

A peer p sends recipe recommendation (R2) requests to other peers (with a TTL) to retrieve their recipes which are likely to be interested by p. A receiving party compares the user profile with its own profile. If similar enough, i.e., their similarity is higher than a certain threshold, it will return recipes to p according to p's set of preferred recipes acquired in its profile. The returned results are then aggregated by p according to its local constraints for the recommendation.

The profile of a peer user is defined as a tuple:

$$profile = (PI, PR, PI, PS) \tag{10}$$

where PI is the personal information of the peer user such as age, gender, location, etc, PR is a set of recipes preferred by the user, which can be acquired from the user's interaction history with the system. Recipes in PR are represented via the Flavor Model introduced earlier. Similarly, PI and PS represent the user's preferred ingredients and cooking styles, respectively.

2) Request Routing

For the propagation of a R2 request, peers are ranked and those who are more likely to contain relevant data to a given request are selected for the request forwarding according to the communities constructed. This is useful in reducing the scope of the request propagation to retrieve a certain number of relevant recipes.

In traditional distributed information retrieval systems, the routing of queries is usually based on the clustering of peers in the network according to content similarity. This can cause problems in distributed systems of large scale. The clustering requires a summary of each peer in the network to be built based on a consistent ontology, which is not always possible in a distributed environment. Furthermore, it may be a performance bottleneck and can potentially cause single point of failure if a central node is employed for the clustering.

We propose a request routing algorithm for our distributed recipe recommendation model described as follows.

Request Routing Algorithm

Process 1. Initial Request

When the user of a peer p_0 issues a recommendation request, the system sends it to some of the other peers in the network which it has a direct link to. The request contains the information about the identity of itself as well as its issuer p_0, the user's profile, and a TTL value that is used for limiting the propagation of the request in the network.

The target peers are randomly selected from all of its direct links using the jump function defined as follows:

$$jump(p,x) = \Pr(p \to x \mid p) = \begin{cases} \dfrac{w_c}{|V^C(p)| + |V^N(p)|} & \text{if } x \in V^C(p) \\ \dfrac{1 - \dfrac{w_c \times |V^C(p)|}{|V^C(p)| + |V^N(p)|}}{|V^N(p)|} & \text{if } x \in V^N(p) \end{cases} \tag{11}$$

where $V^C(p)$ is the set of peers that share memberships with p in at least one community, $V^N(p)$ is the set of peers that do not share membership with p in any community, and w_c is a weight that indicates the importance of community members where $\frac{|V^C(p)| + |V^N(p)|}{2 \times |V^C(p)|} \le w_c \le \frac{|V^C(p)| + |V^N(p)|}{|V^C(p)|}$.

This method is considered as community-based random walk. We find that if the walk in the network is further biased towards trustworthy peers that have high $R_{p \to x}(t)$ values, the recommendation performance can be further improved. The maximal number of peers selected is limited to N_{init}.

Process 2. Request Forwarding

Any peer p that receives a recommendation request from p_0 should check its identity to see whether it has been processed before to avoid duplicate processing of a single request. Peer p then checks the TTL of the request to see whether it should be forwarded. If the request is to be forwarded, its TTL value is subtracted by 1 and it is forwarded according to the jump function. It is similar to Process 1 except that $V^C(p)$ is the set of peers that share memberships with both p and p_0 in at least one community, and $V^N(p)$ is the set of peers that share membership with neither p nor p_0 in any community. The request forwarding process is done before any analysis to the content of the user profile. This ensures that a request can be propagated as quickly as possible through the network, thus reducing its latency.

V. Recipe Recommendation & Adaptation

The recipe recommendation strategies in our system, either to recommend cooking-similar or flavor-similar recipes, or both, will be introduced in this section. After that, we will also present an adaptation mechanism based on the recommended recipe list.

A. Recipe Recommendation

1) Cooking Based Recommendation

The first recommendation strategy we provide in the system is cooking-feature-based approach. It aims at recommending recipes with similar cooking procedure to the one provided by a user, so that he/she would be able to cook the dishes based on his/her current cooking conditions or skills. To achieve this, a cooking similarity measurement is needed to evaluate how similar two recipes are from the perspective of their cooking process. Adopting and adapting the metric proposed in [2] to make it suitable for the distributed environment, the cooking similarity between two recipes, denoted as $sim^C(R_1, R_2)$, is defined as:

$$sim^C(R_1,R_2) = \left[\left(\sum_{i=1}^{m} \left|E_{S_i}\right| \left(\mu \left|E_{SA_i}\right| + \gamma \left|E_{SI_i}\right| \right) \right) \square Per(R_1,R_2) \right]^{1/2} \tag{12}$$

where R_1, R_2 are two cooking graphs, and they share m sub-graphs S_i ($i = 1 .. m$); $|E_{S_i}|$ is the total number of edges of the sub-graph S_i; $|E_{SA_i}|$ and $|E_{SI_i}|$ are the numbers of action and ingredient edges in S_i respectively; μ and γ are adjustable weights for action and ingredient edges, since different user queries may emphasize more on either the flavor (affected by ingredient flow) or the cooking skill (affected by action flow); *Per(R₁, R₂)* is the shared percentage of common *ReciSet* (defined as two connected nodes with a directed edge in the graph). The nature of this similarity measure is to discover recipes with similar cooking patterns, which is represented as sub-graphs consisting of a set of ingredient and action nodes and edges in the cooking graph.

To get a cooking-feature-based recommendation, a user needs to provide a recipe as the query example. The request will be forwarded to his/her most relevant neighbors according to the method introduced in Section IV, and recipes with highest cooking similarities to the example will be returned as recommendation result.

2) Flavor Based Recommendation

As more common case, in a recipe recommender system users may expect to get recipes with similar flavors to their preferred ones. To achieve this, we allow users to issue a recommendation request by selecting an interested recipe as the example, and choose the flavor-based approach as the recommendation strategy. Similarly, the request will be broadcasted to the user's *neighborhood*, and the system will craw the most relevant recipes for the user based on the flavor similarity between recipes:

$$sim^F(R_i,R_j) = \frac{Flv_i \cdot Flv_j}{\|Flv_i\| \|Flv_j\|} \tag{13}$$

where Flv_i is the flavor vector of recipe R_i, and Flv_j is recipe R_j's flavor vector.

3) Hybrid Recommendation

As a compromised case, we also provide a hybrid recommendation strategy for users who want recipes similar to their favorite one in both cooking and flavor features. Generally, the request forwarding mechanism in this case in the same to the previous ones, but replacing the recipe similarity metric with a combination of the cooking similarity and flavor similarity:

$$sim(R_i, R_j) = \alpha \cdot sim^C(R_1, R_2) + (1-\alpha) \cdot sim^F(R_1, R_2) \quad (14)$$

where α is a weighting parameter adjusting the importance for each sub-similarity measure. Since users may have diverse preference over the weights, we would derive the values of α from users' explicit inputs, or using a default value set by the system if there are no user inputs.

B. Recipe Adaptation

After getting a set of candidate recipes from a request issuer's *neighborhood*, our recommendation algorithm does not stop as what most existing social recommendation systems would do. Our observation is that, although the returned results have gone through the process of filtering out a large amount of data in the social network, it is only a representation of a certain community's interest, and may not exactly match the individual requester's preference. To address this problem, an additional mechanism is devised to do automatic adaptations to the social recommendation results by considering personal preference. Individual requester's eating interest is learned from his/her historical behavior.

Two kinds of adaptations supported by our system include replacing the ingredients that the user dislikes or cannot eat, and changing the cooking patterns that would probably rejected by the user.

1) Ingredient Replacement

The primary principle of ingredient replacement is that: an ingredient is only allowed to be replaced by the ones of the same class. For example, pork can only be replaced by chicken, duck, beef, or other kinds of meat. To implement this, we have constructed a domain dictionary that defines the classification of food.

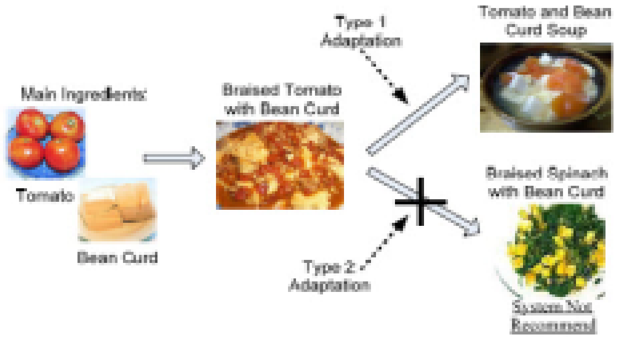


Figure 4. Recipe adaptation example

Another principle is that, after replacing some of the ingredients, the final ones should not conflict with each other and generate bad effects to human's health. For example, spinach is not advised to be cooked together with bean curd, because it would lead to a disease named lithiasis, see Fig. 4. This type of checking can be realized by referring to the knowledge in cooking sciences.

2) Cooking Pattern Replacement.

For replacing a cooking pattern, the most important rule is that the adapted cooking graph should be feasible. Specifically, the input and output ingredients for the original cooking pattern and the replacing one should be, if not exactly matched, at least generally consistent. This is to ensure that the user can still cook the dish out by following the adapted cooking graph. This is implemented by checking if all the nodes, except for the starting nodes and ingredient nodes, are reachable in the graph.

Similarly, after replacing the cooking pattern, a conflict check between ingredients and the cooking method will also be performed according to cooking recipe domain knowledge.

VI. EXPERIMENTS

A. Simulation Experiments

As part of our research, we have carried out simulation experiments of distributed recipe recommendation using a multi-agent system for the evaluation of the proposed model.

Peers interact with each other to simulate the distributed recipe recommendation (R2) requesting activities according to the proposed method. The number of peers in the simulation varies from 200 to 1000, each of which has a database that contains 20 to 40 recipes. Most of the recipes belong to the same category, i.e., they are similar in terms of the Flavor Model, while others are randomly selected from the dataset.

The system keeps track of the R2 requesting activities of the peers in the experiments. In the experiments, we only take users' preferences into account for the recommendation without any other constraint. In particular, we observe whether a result is in the same category as the user's preference and measure the average precision as the user's satisfaction to the recommendation.

Four different methods are used for the comparison of performance. BFS is a basic breadth-first search method. FQM is to use the firework query model to facilitate request routing, which depends on the global clustering of peers based on their content summaries. FCR1 is the flavor- and community-based recipe recommendation that only takes into account the individual-level reputation. FCR2 is based on FCR1 and takes into account both individual-level and social-level reputations.

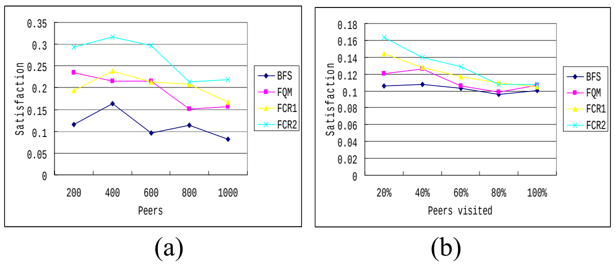


Figure 5. Performance with different number of peers

Fig. 5(a) shows the recommendation performance with different total numbers of peers in the network. From the figure we can see that the overall satisfaction decreases with the increase of the network scale. FCR2 achieves the best performance of the four methods. Fig. 5(b) demonstrates the recommendation performance with different average number of peers visited for an R2 request. The overall satisfaction drops a little with the increment of peers visited because there are also more irrelevant peers visited along with those relevant ones. Again, FCR2 is the best among the four methods.

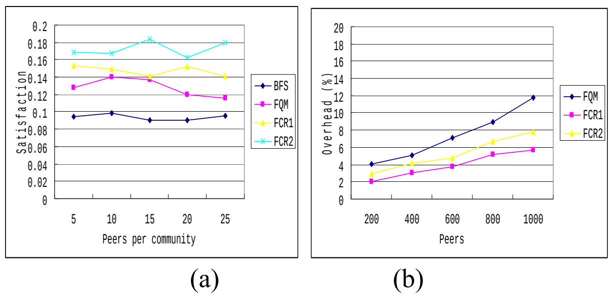


Figure 6. Performance with different community sizes and algorithm overheads

Fig. 6(a) presents the recommendation performance with different average number of peers within a community. It is observed that the recommendation performance remains relatively stable with different community sizes, while FCR2 achieves the best performance in all settings. The overhead traffic introduced by FQM, FCR1 and FCR2 is shown in Fig. 6(b). We can see that the overhead of FCR2 remains at a low level, while that of FQM grows with the size of the network. This is mainly because of the broadcast of content summaries in FQM for peer clustering, while our FCR2 method, due to the merit of the community-based network structure, is able to use messages in relatively small scope to acquire information from most relevant peers. Note that BFS does not need extra traffic to compute its model, hence is not included in Fig. 6(b).

B. Prototype System

A distributed recipe recommendation prototype system has been developed as part of our work. Fig. 7 shows the main interface of the system for one peer in the network, in which all the recommended cooking recipes are displayed in a ranked list. When the user clicks into a recommended recipe, its corresponding multimedia resources, such as dish photo, cooking video, cooking procedure description, etc., will be displayed, as shown in Fig. 8. Besides, the user can also choose to view recipes from the cooking graph perspective (Fig. 9), which facilitates them much to understand the cooking steps of the recipes.

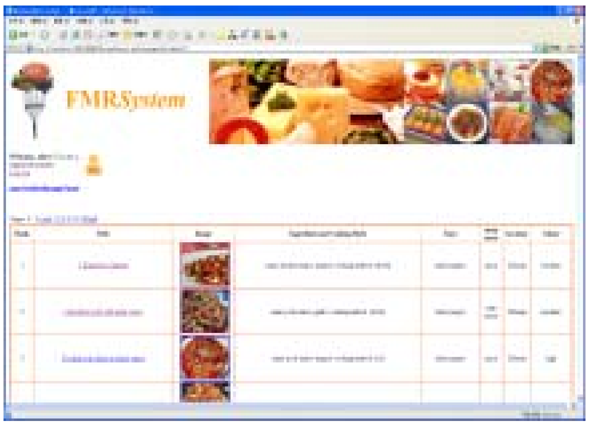
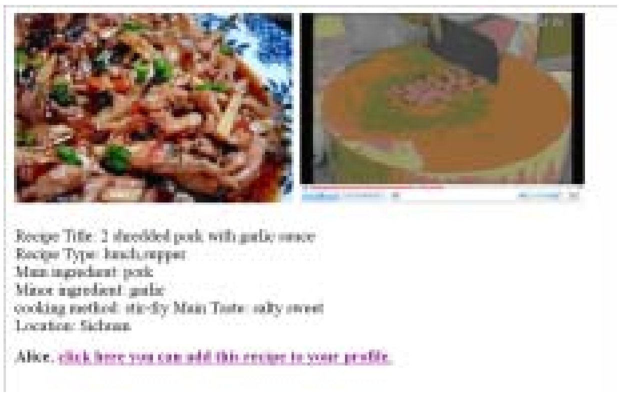


Figure 7. Snapshot of the recipe recommendation prototype system


Recipe Title: 2 shredded pork with garlic sauce
Recipe Type: lunch,supper
Main ingredient: pork
Minor ingredient: garlic
cooking method: stir-fry Main Taste: salty sweet
Location: Sichuan

Alice, click here you can add this recipe to your profile.

Figure 8. Browsing recipe via multimedia resources

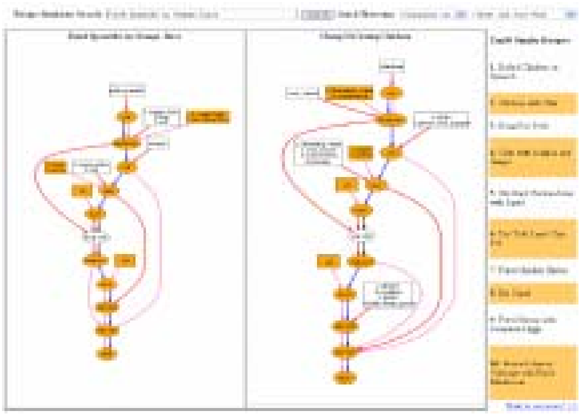


Figure 9. Browsing recipe via cooking graphs

VII. CONCLUSION

In this paper, we have proposed a community-based distributed recipe recommendation and adaptation system. A recipe flavor model is first proposed for modeling recipes and validating recipe adaptations. Our proposed method combines community- and graph-based approaches for recipe recommendation by satisfying certain criteria. Community-based approaches utilize a reputation model to facilitate the construction of peer communities, which are utilized for effective and efficient recommendation in P2P networks. Graph-based methods, which are more tightly related to the application domain, are used for recipe filtering and adaptation. We have exploited procedure flow graphs of cooking recipes to calculate common-subgraph-based recipe similarity, and used it to facilitate recipe filtering. A graph-based ingredient and cooking style replacement approach is devised to adapt recommended candidates that the user may probably want to adapt.

Simulation experiments have been conducted to evaluate the proposed model, and the results show good performance of it.

For our subsequent research, we plan to further study the effect of graph-based filtering and flavor-based adaptation prediction. Other issues of concern include developing more sophisticated routing schemes for distributed recipe recommendation.

ACKNOWLEDGMENTS

This research is supported by the Research Grants Council of the Hong Kong SAR (project no. CityU 117608) and a Strategic Research Grant from City University of Hong Kong (7008043).

REFERENCES

[1] Wang, L., Li, Q. A personalized recipe database system with user-centered adaptation and tutoring support. In ACM SIGMOD Ph.D. workshop on Innovative database research (IDAR), 2007.

[2] Wang, L., Li, Q., Li, N., Dong, G., and Yang, Y. 2008. Substructure similarity measurement in chinese recipes. In Proceeding of the 17th international Conference on World Wide Web. WWW '08. ACM, Beijing, China, 979-988.

[3] J. Chen, C. Hu, and C. Su, Scalable retrieval and mining with optimal peer-to-peer configuration, IEEE Transactions on Multimedia, vol. 10, no. 2, pp. 209–220, 2008.

[4] I. King, C. Ng, and K. Sia, Distributed content-based visual information retrieval system on peer-to-peer networks, ACM Transactions on Information Systems, vol. 22, no. 3, pp. 477–501, 2004.

[5] K. Liu, K. Bhaduri, K. Das, P. Nguyen, and H. Kargupta, Client-side web mining for community formation in peer-to-peer environments, SIGKDD Explorations, vol. 8, no. 2, pp. 11–20, 2006.

[6] Y. Wang and J. Vassileva, Trust-based community formation in peer-to-peer file sharing networks, in WI 2004. 2004, pp. 341–348.

[7] S. Ramchurn, T. Huynh, and N. Jennings, Trust in multi-agent systems, The Knowledge Engineering Review, vol. 19, no. 1, pp. 1–25, 2004.

[8] Wei Chen, Qing Li and Liu Wenyin, Exploiting Peer Relations for Distributed Multimedia Information Retrieval, in Proceedings of ICME 2009, IEEE, 2009, 1154-1157.

[9] M. Svensson, K. Höök, J. Laaksolahti, A. Waern, Social navigation of food recipes, Proceedings of the SIGCHI conference on Human factors in Computing Systems, Seattle, Washington, United States, 2001, pp341-348.

[10] Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Neidl, W. (eds.) The Adaptive Web: Methods and Strategies of Web Personalization. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007).

[11] Konstas, I., Stathopoulos, V., and Jose, J. M. 2009. On social networks and collaborative recommendation. In Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in information Retrieval (Boston, MA, USA, July 19 - 23, 2009). SIGIR '09. ACM, New York, NY, 195-202. DOI= http://doi.acm.org/10.1145/1571941.1571977

[12] Ma, H., King, I., and Lyu, M. R. 2009. Learning to recommend with social trust ensemble. In Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in information Retrieval (Boston, MA, USA, July 19 - 23, 2009). SIGIR '09. ACM, New York, NY, 203-210. DOI=http://doi.acm.org/10.1145/1571941.1571978

[13] J. Lu and J. P. Callan, Content-based retrieval in hybrid peer-to-peer networks, in Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003. ACM, 2003, pp. 199–206.

[14] A. Zunjarwad, H. Sundaram, and L. Xie, Contextual wisdom: social relations and correlations for multimedia event annotation, in Proceedings of the 15th International Conference on Multimedia 2007, Augsburg, Germany, September 24-29, 2007, R. Lienhart, A. R. Prasad, A. Hanjalic, S. Choi, B. P. Bailey, and N. Sebe, Eds. ACM, 2007, pp. 615–624.

[15] Jill Freyne and Barry Smyth. An experiment in social search. In Paul De Bra and Wolfgang Nejdl, editors, Adaptive Hypermedia and Adaptive Web-Based Systems, Third International Conference, AH 2004, Eindhoven, The Netherlands, August 23-26, 2004, Proceedings, volume 3137 of Lecture Notes in Computer Science, pages 95-103. Springer, 2004.

[16] Sabater J, Sierra C. Reputation and social network analysis in multi-agent systems. Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). ACM Press: New York, 2002; 475–482.