A Recipe Based On-Line Food Store

Martin Svensson, Jarmo Laaksolahti, Annika Waern, Kristina Höök
Swedish Institute of Computer Science
Isafjordsgatan 22
Box 1263, S-164 29 Kista, Sweden
{martins, jarmo, annika, kia}@sics.se

ABSTRACT
In this paper we present a recommender system design for recipe based on-line food shopping. Our system differs in two major ways from existing system. First we use an editor that labels clusters of users, such as meat lovers and vegetarians; based on what recipes they have chosen. Secondly, these clusters are available to users, so they can not only choose recipes based on their own user group but also navigate among other user groups.

Keywords
On-line shopping, recommender systems, social navigation, collaborative filtering

INTRODUCTION
Existing on-line food stores are all 'dead' spaces where users fill in how many milk packages, etc. they want to have delivered to their doorstep. In a study by Richmond (1996) on shopping in a VR environment, results showed that users also want to be able to access the social aspects of a physical store, they want to socialise with other people and have a multi-user experience.

How can the ideas of social navigation be made central and be used to inform design of on-line food stores? One trail that we can follow is to recommend recipes using collaborative filtering techniques (Resnick, 1997). Recipes are interesting accumulated pieces of knowledge in this context. Through which recipes we cook from we convey a lot of information about our personality, which culture we belong to, our habits, etc.

Making recommendations on which food to buy based on recommending recipes is an interesting functionality in itself. Imagine that we on top of that add accumulation of user behaviour so that we understand which groups are most likely to choose which recipes. We have designed such a system that works as follows: as a (by the system) known user logs onto the system, it will put up a recommended recipe. This recipe is the highest ranking recipe at that point in time, for the category of users to which the user belongs. The user can add the recipe to his/her shopping basket, which in turn adds the ingredients from the recipe to the list of items that will be delivered to their doorstep. The user can then ask for the next-best recipe that fits with his/her category of users - much in the same way as Amazon.com recommendations ("other people who bought this book also bought these books"). The recommended recipe will be chosen by the system on the basis of three different characteristics that the user can manipulate: user groups, the category of food (Italian, Thai, etc.), and any particular ingredient that should be included (shrimp, beef, etc).

The rest of this paper will describe the design of our on-line food store in more detail. We start off by taking a closer look on how our recommender system works and then move on to discuss some additional functionality that may be feasible to add. Finally we finish it off by putting it all together in an imaginary user interface.

A MIXED APPROACH
A common problem with existing recommender systems, such as, GroupLens (Konstan et al., 1996), Firefly (Shardanand and Maes, 1995), and Phoakes (Terveen et al., 1997), is that they give little or no feedback to a user on what user group they belong to, or what user groups a recommendation is built upon. Since recommender systems base their recommendations on what other similar users have done in the past we believe that this is a very important piece of information, that should be provided to the user. A problem is of course the rather complex task of automating “labelling” of user-groups. For instance, it would be extremely difficult for the Firefly system to label a cluster of users as “reggae lovers with a flavour of ska”. However, if this could be done we would get a much richer recommender system. Imagine using the Phoaks system and getting information on what type of users recommend certain links. So, for example, an expert user would probably not follow links the novices often recommend.

Our solution to the labelling problem is to put an "editor" back into the loop. The editor will examine the clusters of users (based on which recipes they have chosen) and "name" those with fuzzy names that convey somewhat of their content: "vegetarians", "light food eaters", "spice lovers", etc.

Labelling of user groups not only tell something abut the user’s own group, but also give information about other

user groups. This will allow a user to not only navigate from the highest ranked piece of information to the lowest (based on his/her user group), but also to navigate among groups of users. In the recipe domain this seems like a sensible idea; a recipe that is rather low ranked because the user is classified as a "meat lover" can still be the recipe to choose since it is highly ranked for "thai food lovers".

The recommended recipe will be chosen on the basis of three different characteristics that the user can manipulate: user groups, the category of food (Italian, Thai, etc.), and any particular ingredients that should be included (shrimp, beef, etc). The idea is that the collection of user groups will evolve over time as the editor finds more and more groups of users.

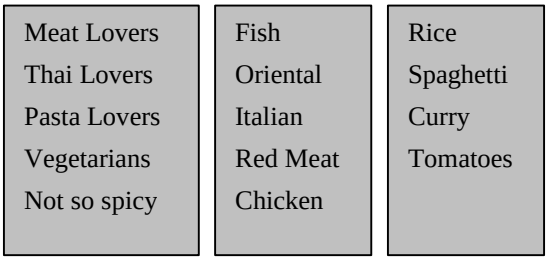

Figure 1. User groups, categories, and ingredients respectively

Imagine we have the following scenario; in our recommender system there are currently five user groups, five categories, and four ingredients (see Figure 1). A user tells the system that s/he wants recipes based on what meat lovers think, based on the oriental category, and with the ingredient curry (the user could herself be classified as a vegetarian). The system will create a list of recipes (not visible to the user) based on what meat lovers have chosen in the category oriental with the ingredient curry. The system then ranks the list according to the user's own group (vegetarian). Since vegetarians have previously chosen only one recipe in the list this will be the highest ranking, all others will be equally ranked. The user can now start to traverse the list based on his/her group, or more interestingly chose another user group to base the ranking upon (e.g. Thai lovers). Also, imagine the "not so spicy" group have never chosen any of the recipes in the list. It will therefore not be possible to rank the list according to that group.

In this way, a user can try out being a vegetarian, or a member of some other user group, for a period of time. A user's group is of course in no way static; if a user consistently chooses recipes based on what vegetarians like, s/he will gradually move towards the vegetarian user group.

Our solution will provide the users with more insight into the social trails of their own actions as well as other users' actions that have lead to the recommendations they finally get. It also provides some insight into the inner workings of the recommender system.

ADDING MORE SOCIAL NAVIGATION TO OUR STORE

Based on our underlying recommender system design it is possible to give more clues about the character of a specific recipe, thereby making it easier for users to decide whether they like it or not. Also, we can find more interesting ways to navigate among recipes not based on the recommended list. In the following section we will discuss three additional features that we add to our recipe store: more from the same source, user comments, and overall ranking.

In the introduction we mentioned that the recipes we cook from tells a lot about ourselves. Something that we did not mention is the fact that the recipes we choose are often recommended to us by other people that we trust. It is often the case that we go to the same source when we search for recipes, especially when we want to try out something new that we have never cooked before. Therefore each recommended recipe is attached with a pointer to other recipes from the same source (e.g. a person or a cookbook). To this we add two forms of readware (Hill et al., 1992) to each recipe. Users have the option to both comment on- and rate recipes. The rating will be anonymous, so a user will just see a mean of all users' ratings of a recipe. The comments, on the other hand, will not be anonymous, so a user can see who wrote what.

Finally we are experimenting with different forms of real-time awareness of other users. One approach that we have discussed is to base our shop on a weekly planner, which is visible to all other users. Instead of shopping by a single recipe at a time, a user builds up a weekly schedule on what to eat for breakfast, lunch, and dinner. So, whenever a user enters the store s/he can check someone else's schedule for inspiration, and if the other user is online, maybe even start a real time chat with him/her.

PUTTING IT ALL TOGETHER

Now, how do we merge the pieces outlined above into one coherent interface? First of all we think that it is extremely important that all functionality is accessible from a single frame, e.g. the user should at all times be able to access all functionality without having to step through several screens. Secondly, we feel that it is important with an interface that is intuitive to the user. However, at this stage we are just starting to create a very sketchy user interface, and we do not claim to have found the optimal solution to this problem. Thirdly, we want to experiment with different ways of visualising user groups, not only using text but also characters that represent, for instance, vegetarians and meat lovers. In figure 2. the first design of the user interface is presented.

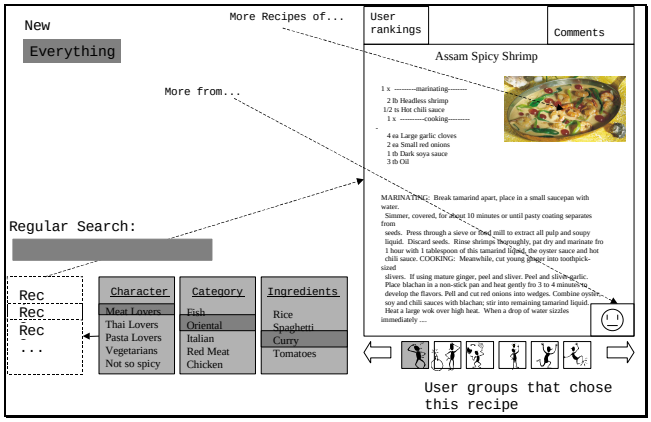

Figure 2. The first design of the interface.

CONCLUSION

In this paper we have tried to present a new and exciting approach to building a recommender system. The key feature that we want to stress is the use of labels for different clusters of users. When the user understands what sort of user that the system classifies him/her as, it becomes easier to understand why some information gets high ratings and some information gets low ratings. Also when the recommender system allows a user to navigate among labeled clusters of users, new and interesting ways of choosing recipes may emerge. Finally, as a side effect of our system, we are actually changing the way people do on-line shopping of food: from shopping groceries to shopping recipes.

REFERENCES

1. Hill, W, Hollan, J, Wroblewski, D, McCandless, T (1992). Edit wear and read wear. *Human factors in computing systems*, 3-9.
2. Konstan, J, Miller, B, Maltz, D, Herlocker, J, Gordon, L, Riedl, J (1997). GroupLens applying collaborative filtering to Usenet news. *Commun. ACM* 40(3), 77-87.
3. Resnick, P, and Varian, H (1997). Recommender Systems. *Communications of the ACM*, Vol. 40, No. 3.
4. Shardanand, U, Maes, P (1995). Social information filtering: algorithms for automating "word of mouth". Human factors in computing systems, 210-217.
5. Terveen, L. G., Hill, W, Amento, B, McDonald, D, Creter, J, (1997). Building Task-Specific Interfaces to High Volume Conversational Data. *Human factors in computing systems*, 1997, 226-233.