Design of a recommender system for the Talking-Points system.

Introduction to Talking-Points

Talking-Points is a student project with the objective of designing a community driven orientation system for the blind (and also sighted people). The system will consist of two major parts: an internet database with location information and a mobile device that will read this information to the user. The database is going to store some moderated core information (titel, description, address, coordinates, ...) and certain types of additional user-generated information (comments, news, reviews, ...) for each location. The locations will be tagged with a Bluetooth beacon to make it identifiable. When a user passes a tagged location, his mobile device will detect the Bluetooth beacon, query the database for all the location information and present it to him. This will be done through different user interface modules. The main interface module is going to be a headset audio interface, but the user can also use the graphical user interface of his mobile device, depending on the context. The system will present the core information for each location and then let the user navigate through all the additional information, either by voice commands (e.g. "option 1", "skip", "more") or through the device's keypad. The system's purpose is to let a blind person experience the locations he is passing on his journey, which he would normally not notice at all. In case of sighted users the purpose would be to get to know new, unfamiliar locations on the way.

Objectives

There are two objectives for a recommender system within the Talking-Points system. The first objective is to **personalize and filter the additional information items** for each individual location, especially when there are too many. For example, only read the comments of a location that will probably be most relevant to the user, or at least order them by relevancy.

The second objective would be to **recommend locations** that the user will probably find interesting. For example, if there is a high location density in one area the system would only detect those locations with highest probability of being interesting instead of reacting to every single one. Another example would be to recommend other interesting nearby locations.

In summary, the objective of the system is to account for the limited attention of the user by "prefiltering" the information for him. Due to the limits of this paper, I will only consider the first of the two objectives above: personalizing and filtering of additional information. I choose that objective, because during our initial user research we found out that users seem to be interested in very different aspects about locations and are asking for a high level of customization [1].

Input information

Which input information can be used for our recommender system? The answer to this question and many others related to the implementation of the recommender system highly rely on how the system handles privacy. See more about this topic in the Privacy section of this paper.

Generally, the information that could be available ranges from simple **implicit** information **from using the system** (which locations does the user pass, how often, when, ...), over information **from interacting with the system** (which location information does the user listen to, for how long, what does he skip, ...), to **explicit** information given to the system (commenting on, rating or tagging locations and their additional information). In addition to that, there is also the information about the locations itself (title, description, distance, tags, ...).

For recommending additional information items for locations, it is basically the following input information that is available:

- the time and number users listened to the information item
- whether the user skipped the information item
- tags for that information item (when a user types in additional information he can classify it through tags like "blind" or "tourist")
- maybe ratings or comments on the information item (in a later version)

Privacy

Before I get any further I want to talk about privacy because it is one of the most crucial factors in designing this recommender system. Privacy is so important for the Talking-Points system because of its ubiquitous character. People are already worried about Amazon tracking all their shopping activity to the last detail. This system could theoretically create detailed profiles about much more and wider activities. It could track where the user has been going, when and in which order, what he thought about all the locations and more. Therefore, it should be the goal of the system to collect as little user behavior data as possible and if it is absolutely necessary for the recommender system, there should be an option for the user to keep it to himself (on his client).

Algorithm

Because of the mentioned privacy issues, I choose an approach inspired by the concept of Niwa et al 2006 [1] for a web page recommender system based on folksonomy mining. It would **recommend additional information items to users based on their relation to tags**. Its main advantage is that it enables a clean separation between server and client. The classification of items happens on the server, while the preference of the user for certain items is calculated and stored on the client. To get a recommendation score for a new item, the system looks at which items the user liked and how they relate to the classification of the new item.

On the server the classification of an item happens through the calculation of its "affinity level" to tags. Instead of using single tags, the system uses tag clusters to face the problem of tag redundancy (different tags that mean the same). The calculation happens analogous to that of the affinity level between users and tag clusters in Niwa et al [1], you just have to replace users with items. For every item we end up with its affinity scores to all of the tag clusters which will be the transmitted to the client together with the additional location information.

On the client the affinity of the user to certain tags is logged. The affinity is represented as a score calculated based on how long the user listened to each information item associated with the tag or when he skipped it. The score could be based on the percentage of the information the user listened to. Thus, if he listened to 50% of the information of an item with the tag 'blind' the score would be 0.5. Those scores are then aggregated to one score for each tag, for example the mean of the individual scores. Finally, the user's affinity scores for each individual tag allow us to determine his affinity towards the tag clusters. This would be done by adding up all the affinity scores for tags within each cluster. The tag clusters would have to be synced with the clients for that purpose.

When recommending items, we take an item's affinity scores for each of the tag clusters and multiply them with the user's affinity scores for the clusters. From those weighted scores we create a final recommendation score by which we can order or filter the information items. The ones with the highest scores should be the most relevant to the user.

This is only a rough outline of how the recommendation score would be calculated. Some of the scores mentioned have not really been clearly defined and they will most likely have to be normalized in one way or the other in order to get really good results. But the described design should be a good start that can be further refined once implemented.

Besides a clean and privacy-enabling separation between client and server, there are also other advantages of the approach I chose. I think that basing recommendation on content in form of tags actually yields better results than basing them on user behavior. I think we are dealing with very specific user profiles in our system that might be difficult to detect through a user-user based approach. Not all blind users are behaving the same, but they are probably all interested in information tagged with "blind". Another advantage is that this type recommender system is quite transparent and can be easily adjusted by the user. Even before/without collecting any usage data the user could subscribe or unsubscribe to certain tags or tag clusters.

The main disadvantage of my approach is its reliance on the availability of tags. If the system and its users are not able to create enough tags for its information items, the whole system will fail. That is a very crucial dependency that might very well bring down the whole system.

Alternatives

If we decide that we neglect privacy and collect user behavior data on the central server, then we would be able to use a **user-user** based algorithm. We could use the percentage based affinity score for information items mentioned above to find similar users. Based on which information they liked the most we could recommend it to the user. This approach would also allow us to join interaction information from the website with interaction information from the mobile device (in my approach we only used the latter).

Instead of connecting items and users through tag clusters, one could also just use an **item-item** approach. The similarity between items could be calculated on the server, shared with the client and the client could then determine whether the user will like an item based on if he liked similar items. The disadvantages compared to my approach would be: that this would be less transparent, that the calculation of similarities would use more resources (especially since it can be expected that the number of items will be very high) and that the system could not be as easily adjusted by the user.

Another alternative would be a privacy-preserving **SVD** approach as described by Canny [3]. In this case every user only shares a precomputed aggregate of his user data, which allows the computation of personalized recommendations but doesn't endanger his privacy. The problem with this approach is that it relies on the users sharing genuine information and therefore is very vulnerable to manipulation.

Presentation and user interface

As I mentioned before there will be different user interfaces modules for the system. So the presentation of the recommendations will differ for all of them.

Voice control + headset

Here the aim is to limit the information read to the user to a minimum, because the attention is the sparsest. So, the recommendations would be used to filter out irrelevant information and read the information first that will most likely be relevant to the user. It would defeat the purpose to add any additional information, for example about how the recommendations were created.

Mobile GUI

If the user gets the location information through the graphical user interface of his mobile device, the presentation will be different. Here it is easier for the user to get an overview over the information and to decide for himself what he is interested in. So I think it would probably be useful to present a relevance score besides each information item. If you are looking at a list of information (e.g. comments), this would give the user additional clues on what might be interesting to him but would not use much space on the small display. In addition there could be separate screens for information that the system would recommend at any given point. For example: "based on your current location you might also want to check out the following other nearby locations". This user interface could also be used to monitor the user's affinity towards tags and give him the ability to adjust them by subscribing/unsubscribing to tags.

Website

Whether you will be getting recommendations on the website relies on the privacy model again. In my approach there will be no connection between the profile on the mobile device and the profile on the website. So it would be a totally separate system on the website.

In general, the presentation on the website would generally be similar to the mobile GUI. The big difference is that there is much more screen space available. This space could be used to tell the user how recommendations were created and give him the chance to correct the system if he disagrees with the recommendations.

Evaluation of recommendations

The aim of the Talking-Points system is to enhance the users' experience while walking around. To tell them interesting things about their location or recommend other interesting locations. It is not a wayfinding system or similar used to find particular locations. Thus, precision would be a much more important factor in evaluating the system's recommendations than recall. Recommendations don't have to be complete, but they have to be good and interesting. There is not necessarily a penalty for missing any particular good recommendation, as long as all the other recommendations given are not uninteresting. Because of that it might be useful to change the algorithm to give priority to recommendations that are based on richer data (e.g. the user liked many items with that tag instead of just one). This would increase the probability of recommendations being good, but it also might affect the diversity and serendipity of the recommendations.

There are multiple methods of determining whether a recommendation was good or not, which depends on the interface used. The standard and most unobtrusive way would be to see if the user wants to hear more information about a recommended information item, or if he skips it. And if yes how long he listens to it. There could also be other ways like a question "was this helpful or not?". Another way that works for pretty much every recommendation system are surveys to gauge the satisfaction with the recommendations. Depending on the resonance, the system could be tweaked to recommend different information.

Manipulation

The main part of the system that would be available to manipulation is the tagging of information items. As I mentioned above, the quality of the recommendations will heavily rely on the quality of the tags. A possible way to manipulate the system is to tag the information that you want users to hear with popular tags. But the damage created this way is limited since the calculation of affinity between tags and items considers the rareness of the tag. Other than that there are really no big opportunities for manipulation, since most of the computation happens on the client.

Conclusion

I think the approach which I described here is a clean and feasible design for a recommender system for the Talking-Points system. It considers the crucial privacy concerns, it is clean and transparent, it should be feasible to implement it, it is very safe from manipulation and most of all should create good, personalized recommendations for its users. The major barrier for its success would be the number and quality of tags in the system. So the tagging interface should have a top priority in the implementation process.

References

(1) Talking-Points blog, What would you like to get from Talking-Points ..., http://talking-points.org/2008/02/09/what-would-you-like-to-get-from-talking-points/, February 9 2008

(2) Niwa S., Doi T., Honiden S., "Web Page Recommender System based on Folksonomy Mining for ITNG '06 Submissions," *itng*, pp. 388-393, Third International Conference on Information Technology: New Generations (ITNG'06), 2006

(3) Canny, J., Collaborative Filtering with Privacy via Factor Analysis, In ACM Conference on Research and Development in Information Retrieval (SIGIR 2002), 2002