Helsinki University of Technology
Information and Natural Sciences

Eric Malmi

MULTIAGENT SIMULATION AND VISUALIZATION OF SOCIAL INTERACTIONS: FRAMEWORK AND RECOMMENDER SYSTEM

Bachelor's thesis

Espoo 10.12.2008

Thesis instructor:

M.Sc.(Tech.) Juha Raitio

Author: Eric Malmi

Title: Multiagent Simulation and Visualization of Social Interactions: Framework and Recommender System

Date: 10.12.2008 Language: English Number of pages: 6+24

Degree program: Systems and Operations Research

Supervisor: Prof. Harri Ehtamo

Instructor: M.Sc.(Tech.) Juha Raitio

Social networking websites provide a huge source of social data. We have built a multiagent simulation framework to handle this data and to simulate social interactions stochastically. Progress of the simulation is visualized using self-organizing maps. Based on the framework, we propose a recommender system that combines the traditional collaborative filtering and content-based methods. To evaluate the framework and the recommender system we collect music listening data from the Last.fm service and analyze two different example runs of the simulation. The framework proves to be functional but we are lacking history data which could be used to calibrate the free parameters in the simulation and validate its predictions.

Keywords: social modeling, multiagent simulation, stochastic simulation, self-organizing map, collaborative filtering, recommender system, practice theory

Tekijä: Eric Malmi

Työn nimi: Sosiaalisten vuorovaikutusten moniagenttisimulointi ja visualisointi: kehys ja suosittelijajärjestelmä

Päivämäärä: 10.12.2008 Kieli: Englanti Sivumäärä: 6+24

Tutkinto-ohjelma: Systeemi- ja operaatiotutkimus

Vastuuopettaja: Prof. Harri Ehtamo

Ohjaaja: DI Juha Raitio

Internetin verkostoitumissivustot tarjoavat laajan lähteen sosiaaliselle aineistolle. Tämän aineiston käsittelemiseksi on rakennettu simulaatiokehys, joka perustuu stokastiseen moniagenttisimulointiin, ja jolla voidaan simuloida sosiaalisia vuorovaikutussuhteita. Simulaation etenemistä havainnollistetaan itseorganisoivilla kartoilla, jotka kuvaavat agenttien keskinäisiä suhteita sekä agenttien omaksumia käytäntöjä. Käytännöt voivat olla esimerkiksi artisteja tai harrastuksia.

Simulaatio pyrkii ennustamaan, mitä käytäntöjä agentit omaksuvat tulevaisuudessa. Tältä pohjalta on rakennettu suosittelijajärjestelmä, joka suosittelee käyttäjille niitä käytäntöjä, joista he saattaisivat olla kiinnostuneita. Esimerkiksi useat verkkokaupat, kuten Amazon.com, hyödyntävät tällaisia järjestelmiä ehdottaessaan asiakkaalle tuotteita, jotka saattaisivat kiinnostaa asiakasta. Työssä esitelty järjestelmä pyrkii ensin simuloimalla ennustamaan, mitä käytäntöjä agentit omaksuvat, ja sen jälkeen suosittelee näitä käytäntöjä agenteille. Järjestelmä hyödyntää perinteisiä yhteisöllisen suodatuksen (*collaborative filtering*) sekä sisältöpohjaisia (*content-based*) menetelmiä.

Simulaatiokehystä testattiin Last.fm-palvelusta kerätyllä musiikin kuunteluaineistolla. Last.fm-on internetradio, joka tallentaa käyttäjien kuuntelutottumuksia. Kehyksen ja suosittelijajärjestelmän toimintaa havainnollistettiin analysoimalla kaksi Last.fm-aineistolla ajettua esimerkkisimulaatiota. Kehys osoittautui toimivaksi, mutta realististen simulaatioiden aikaansaamiseksi vaadittaisiin historiaaineistoa, jonka avulla voitaisiin määrittää simulaation vapaat parametrit ja todentaa sen antamat ennusteet.

Avainsanat: sosiaalinen mallinnus, moniagenttisimulaatio, stokastinen simulointi, itseorganisoiva kartta, yhteisöllinen suodatus, suosittelijajärjestelmä, käytäntöjen teoria

Acknowledgements

I have had the great opportunity to work on my thesis as an undergraduate researcher, in the group of Computational Cognitive Systems. The whole group, led by docent Timo Honkela, has been very supportive and has given me lots of feedback regarding to my thesis. The funding for my work has come from the KULTA project, which is funded by Tekes, the Finnish Funding Agency of Technology and Innovation. Especially, I would like to thank my instructor, Juha Raitio, for the constant support and for the talks we have had, that have given me a great insight to the topic of my thesis and to the academic world in general.

Otaniemi, 10.12.2008

Eric Malmi

Contents

1 Introduction

Modeling human behavior with computer simulations has influenced several branches of science, e.g. sociology, consumer research and risk analysis. Increase in efficiency of computers and wide accessibility on all kinds of data via the Internet has made it possible for scientists to simulate even complex natural phenomena.

Multiagent system is an approach where intelligent agents model individuals that make their own decisions based on the changes in their environment [1]. Normally, a multiagent simulation uses data describing the initial state and properties of the agents and then simulates how the agent society will evolve. Interactions between agents and an agent and the environment need to be modeled well enough in order to get natural phenomena emerge in the simulation. This can give us better understanding of the phenomena and also help us to predict new phenomena that might emerge.

Recommender systems are an application of social modeling. They try to predict what items a user would find interesting, based on the other items the user has liked and the other items the users with similar interests have liked. These recommendations are widely used in Web shops, such as Amazon.com, and in recommendation Web sites, specialized e.g. in movies such as MovieLens. [2]

Usually recommender systems are classified into three main categories: collaborative filtering, content-based and hybrid systems. Collaborative filtering systems search for users with similar interests and propose the items those users have liked. Content-based systems search for items with similar properties to those that the user has purchased or rated well himself. Hybrid approaches combine these two methods. [2]

Our goal is to build a multiagent simulation framework that should be capable of handling diverse data and diverse interaction models between agents. As input data, the simulation uses a list of practices [3] for each user and a list of properties for each practice.

We also propose a recommender system based on the simulation framework. The system simulates how users' times spent on their practices change when they are in a group where all users interact with each other. Thus, our system predicts what new practices a user will adopt and then recommends those to the user. As test data we use data gathered from Last.fm's Audioscrobbler service [4]. Last.fm keeps track of what artists the users, registered to the system, have listened to and how much. It also provides possibilities for tagging artists. Thus, we can model artists as practices and their tags as the properties of the practices.

To calibrate the simulation and validate its predictions, one would need some long-term history data about how users' practices have evolved in real life. We are lacking this data but we discuss how it could be used if available. To demonstrate the functionality of our framework we show some example runs on the Last.fm data and discuss the emerging phenomena.

2 Simulation Framework

The basic framework behind our simulation is that we have a group of agents which have a group of practices. When the simulation is run, the agents begin to interact and, in consequence, their time spent on practices change and they adopt new practices.

2.1 Objective

The objective of this work is to build a simulation platform where interaction models between agents can be defined to support different simulation cases. Practice, as a concept, should be modeled generally enough so that the framework would suit for variety of purposes.

2.2 Methods

2.2.1 Multiagent Simulation

Agent is an autonomous decision maker that bases its decisions on the environment. That means, it can communicate with other agents and change its behavior according to the changes in the environment. Multiagent simulation is a system that consists of several autonomous agents that begin to interact when the simulation is begun. A multiagent system can be seen as one big problem where agents are solving its subproblems [5].

The "Bottom up"-oriented approach to multiagent simulation means that agents are observed in real life and we attempt to model their behavior as well as possible. Simulation parameters are calibrated so that real life phenomena would emerge. It is used to understand phenomena emerging and to predict new phenomena that might emerge. [6]

2.2.2 Self-organizing Map

Self-organizing map (SOM) is an artificial neural-network algorithm that can be used for data clustering and data visualization [7]. It projects n-dimensional data vectors nonlinearly to lower (usually two) dimensional map. It does not require sample vectors with predefined projections but it takes set of input vectors and learns unsupervisedly how to project them to the map.

To train the map, we initialize each cell with a weight vector of the same dimension than the input vectors. The cells can be initialized with random vectors that represent the input space. With the random initialization, however, the training can take a long time and there are also some more sophisticated initialization methods.

After the initialization, we draw samples from the input vector set. For each sample

vector, we go through all the cells and search for the best-matching unit (BMU), which is the weight vector with the smallest Euclidean distance to the sample. The BMU and its neighbors are then conformed nearer to the sample vector.

After the training, we can project vectors to the map. To project a vector, we get the BMU for the vector and map the vector to the cell of the BMU. SOM tends to preserve the topological properties of the input space and thus the vector neighborhoods in the input space tend to form neighborhoods on the SOM as well. SOM does also a density approximation, which means that it allocates most of the weight vectors to represent the dense parts of the input space.

The maps that we use have different colors as they visualize the U-matrix of the map. The U-matrix colors the cells based on their average distance to the neighboring cells. The distance is calculated in the original input space and the bigger the distance the darker the color. Consequently, we notice that if two cells have a dark stripe between them, they are probably far apart in the input space eventhough they would be nearby on the map. Thus, we are able to distinguish clusters since they are surrounded by dark area.

2.3 Data Model

2.3.1 Agent Data

For each agent, we need a list of all practices they have and the times they spend for the practices. Having obtained the list of all practices and time values we can form a *practice vector* for each user. Putting these practice vectors together, we get an agent-practice matrix, such as one in table 1.

Table 1: Agent input data collected into a matrix. On the y-axis, we have agents and on the x-axis practices (artists). Values in the matrix define how much each agent uses for each practice (how many hours he or she listens to the artist per week).

Agent \ Practice	Coldplay	Dream Theater	Chopin
Bruce	5	0	4
Mike	0	4	4
Lisa	0	10	0

Simulation also takes an influence matrix. The influence matrix determines how much influence each agent has on every other agent. By default, all influence values are 1, in case the influence data is not available. Table 2 shows an example of influence relationships between agents.

It is difficult to define how much influence person A has on person B. However, it can be estimated based on, e.g., the number of friends person A has. Using the

Table 2: An influence matrix that determines how much influence each agent has on every other agent. On the left are the influencers and on the top the conformists.

	Bruce	Mike	Lisa
Bruce	#	0.4	0.7
Mike	1	#	0.9
Lisa	0.7	0.2	#

number of friends we assume that more friends means more influence. Alternatively, we can simply use the information whether A is a friend of B or not and define the influence value, for example, 1 or 0.5 respectively.

2.3.2 Practice Data

For each practice, we need a list of their properties. With the properties, we associate a relevance value which tells how relevant the property is. Of property lists and property relevancies, we can form a *property vector* for each practice. Properties can be, e.g., tags that are associated with binary values (1 = practice has this tag, 0 = practice does not have it) or with float values between 0 and 100 describing the relevance of a property. Putting the property vectors together, we can form a practice matrix, such as one in table 3.

Table 3: Practice data collected into a matrix. On the y-axis, we have practices (artists) and on the x-axis properties (tags). Values in the matrix define how relevant each tag is for each artist.

Practice \ Tag	progressive metal	piano	rock
Coldplay	0	2	100
Dream Theater	100	0	11
Chopin	0	44	0

2.4 Interaction Model

The algorithm for interaction between agents is as follows:

1. Go through all agents and do steps 2-5 for each of them.
2. Choose the influencing partner (*influencer*).
3. Choose one practice from the influencer.

4. Increase agent's time for that practice.
5. Decrease agent's time for other practice(s) so that the total time remains constant.
6. Go back to step 1 and repeat n times.

2.4.1 Stochastic Choosing of Influencer

First we calculate the probability $Pr(S_a = s)$ for each agent s to be chosen as the influencer for agent a. Then we draw the influencer from the distribution of S. We assume that Pr is linearly dependent of the distance between agents $d(a, s)$ and the agent's s influence $f(a, s)$ on agent a, so that the smaller the distance the bigger the probability and the bigger the influence the bigger the probability. Thus, we can write

$$Pr(S_a = s) = k(w_d(b - d(a, s)) + w_f f(a, s)), \tag{1}$$

where w_d is the weight given for the distances, w_f the weight given for the influence values and b a constant. We define b to be $1.1 \cdot max(d(a, s_i))$ so that the probability will never be negative and even the farthermost agent has always a little probability to be chosen the influencer. From the axiom

$$\sum_{i=1}^{n} Pr(S_a = s_i) = 1, \tag{2}$$

we calculate the free parameter k.

The distance between two agents is calculated as Euclidean distance between agents' practice vectors. This metric is also referred as the Mean Squared Difference and used, e.g., in Collaborative Filtering methods when determining similarity of user profiles [8].

2.4.2 Stochastic Choosing of Practice

First, we calculate the probability $Pr(X_s = x)$ for each influencer's practice x to be chosen as the influencing practice for the conforming agent a. Then we draw the influencing practice from the distribution of X. We assume that Pr is linearly dependent of the distance between the conformist and the chosen practice $d(a, x)$ and the time the influencer s spends on the chosen practice $t(s, x)$ so that the smaller the distance the bigger the probability and the bigger the time the bigger the probability. Thus, we can write

$$Pr(X_s = x) = k_2(w_{d2}(b_2 - d(a, x)) + w_t t(s, x)), \tag{3}$$

where w_{d2} is the weight given for the distances, w_t the weight given for the times and b_2 a constant. We define b_2 to be $1.1 \cdot max(d(a, x_i))$ so that the probability will never be negative and even the farthermost practice has always a little probability to be chosen the influencer. From the axiom

$$\sum_{i=1}^{n} Pr(X_s = x_i) = 1, \tag{4}$$

we calculate the free parameter k_2.

The distance d between the influencing practice and the conforming agent's practices has to be defined in the case the agent has more than one practice. Therefore, we determine the distance between practice p and agent's practices as follows:

1. Calculate the distance between p and each practice the agent has.
2. Choose the smallest distance of them.

Distance between two practices is calculated either as Euclidean distance between practices' property vectors or as Euclidean distance on the self-organized practice map.

SOM does a nonlinear projection of practices into two-dimensional map and thus the topological properties cannot be fully preserved in the mapping. That is, on the map The Beatles might be closer to The Rolling Stones than to Queen but in the original tag space it could be vice versa. However, the distance on the SOM can be used as a reasonable estimate for the actual distance. Motivation for this is that calculating the Euclidean distance is much faster in two dimensions than in the original tag space where there might be hundreds or even thousands of dimensions.

2.4.3 Increasing Agent's Time for Chosen Practice

We have two different formulas for adjusting agent's time (t_A) for the chosen practice. *Method 1* adjusts agent's relative time nearer to the influencer's relative time (t_B/t_{totB}) according to

$$t_A = t_A + \alpha(\frac{t_B}{t_{totB}} - \frac{t_A}{t_{totA}})t_{totA}, \tag{5}$$

where t_{totA} and t_{totB} are agents' total times for all practices. α is a constant between 0 and 1. We use a value around 0.2 for α. t_A is changed only if the subtraction in the formula is positive.

Method 2 is similar to the first one, except that it does not take into account the influencer's practice time but it increases t_A by a fraction of the agent's average listening time $\overline{t_A}$

$$t_A = t_A + \alpha\overline{t_A}. \tag{6}$$

Intuitively, the first method is more accurate since the more the influencer does a practice the bigger the influence is. However, if for example user A listens to an artist 2 hours per week and in total he listens to music 10 h/week and user B listens to the artist 10 h/week and in total 100 h/week, it seems incorrect that B could not get A to listen to the artist more because he has smaller ratio (i.e. $t_B > t_A$ but $\frac{t_B}{t_{totB}} < \frac{t_A}{t_{totA}}$).

2.4.4 Decreasing Agent's Time for Other Practice(s)

To avoid the agents just increasing the amount they use for their practices, we make an assumption that agent's total time for practices remains constant. Currently, the simulation chooses randomly a practice from the agent and decreases the time used for it. Decrease in time equals to the increase in the other practice time. If a time is about to go negative, we randomize a new practice and decrease its time. This is repeated so many times that the same amount of time is decreased that was increased.

It may seem more reasonable to drop off the small practices when the time for some practice is increased. However, this would also drop the recently appeared practices and thus it would be very unlikely for a new practice to replace an older one. Therefore, we should probably invent an activity value for a practice and the practices with high activity would be unlikely to decrease. Nevertheless, the activity metric should be one that the needed data is available for.

2.5 Model Calibration and Validation

The simulation attempts to predict how a society of agents will evolve. It predicts how much time each agent uses for each practice in the end of the simulation (*outcome*). To validate the prediction, we should have data about how societies have actually evolved (*history data*) and then compare the outcome to the history data. To compare how well the predicted and the actual outcomes match, we choose to encode an outcome into a vector by concatenating each user's practice vector to one vector. This is called the *outcome vector*.

There are free parameters in our model that need to be calibrated so that the simulation gives the best possible predictions. The history data is used also for the calibration. Simulation outcomes are fitted to the history data iteratively by changing the parameters after each simulation run and minimizing the error between actual outcome and the simulated outcome. The error (E) can be defined in the least squares sense [9]

$$E = ||\mathbf{v_a} - \mathbf{v_p}||^2, \tag{7}$$

where $\mathbf{v_a}$ is the actual outcome vector and $\mathbf{v_p}$ the predicted outcome vector.

In calibration and validation the history data is divided into two sets. The first set is used to calibrate the model as described above. The other set is used to validate the model. Validation is done by examining how well the output of the simulation fits to the validation data which has not been used in the calibration.

2.6 Stochastic Stability of Predictions

Stochastic simulation is reasonable if the system tends to converge to similar states using similar initial conditions. That is, the variance of the predictions should not be too large, else the confidence interval of a prediction becomes wide and the prediction has little relevancy. Variance or standard deviation alone, however, do not tell much if not related to the mean [10]. Therefore, we use the coefficient of variation (CV) [10], which is a dimensionless statistic and thus gives us better understanding of the predictions. CV is defined as the ratio of the standard deviation σ and the mean μ

$$c_v = \frac{\sigma}{\mu}. \tag{8}$$

We run the simulation for n times with the same initial parameters and get n different outcome vectors. CV is first calculated for each outcome vector element separately. Then we take the mean of the element CVs and the formula for the mean CV becomes

$$\overline{c_v} = \frac{1}{N_1} \sum_{i=1}^{N_1} \frac{\sqrt{\frac{1}{N_2-1} \sum_{j=1}^{N_2} (x_{ji} - \overline{x_i})^2}}{\overline{x_i}}, \tag{9}$$

where N_2 is the number of outcome vectors, N_1 is the number of elements in an outcome vector, x_{ji} is the ith element in the jth outcome vector and $\overline{x_i}$ is the mean of ith elements in all outcome vectors.

2.7 Visualization of Simulation

Visualization can be a powerful approach for handling large volumes of data [11]. Animating the progress of the simulation, rather than using pure statistics, helps the user to follow the progress. It can also provide the developer a very useful tool for debugging the system.

In our simulation we use two self-organizing maps: the agent map and the practice map. The agent map visualizes how agents' practice times change over time. All agents are mapped on the SOM based on their practice vectors. As the practice vectors change while agents interact, they are mapped to new cells on the SOM according to their new BMUs. The agent map is trained with the initial practice vectors of agents.

The practice map contains all the practices of all agents. It visualizes practice distribution of one agent at a time. Agent's practices are marked with circles on the map. The bigger the circle the more time agent uses for the practice. As simulation goes on, new circles appear and old ones grow and shrink. The practice map is trained with the property vectors of practices.

Figure 1 illustrates the user interface of the simulation.

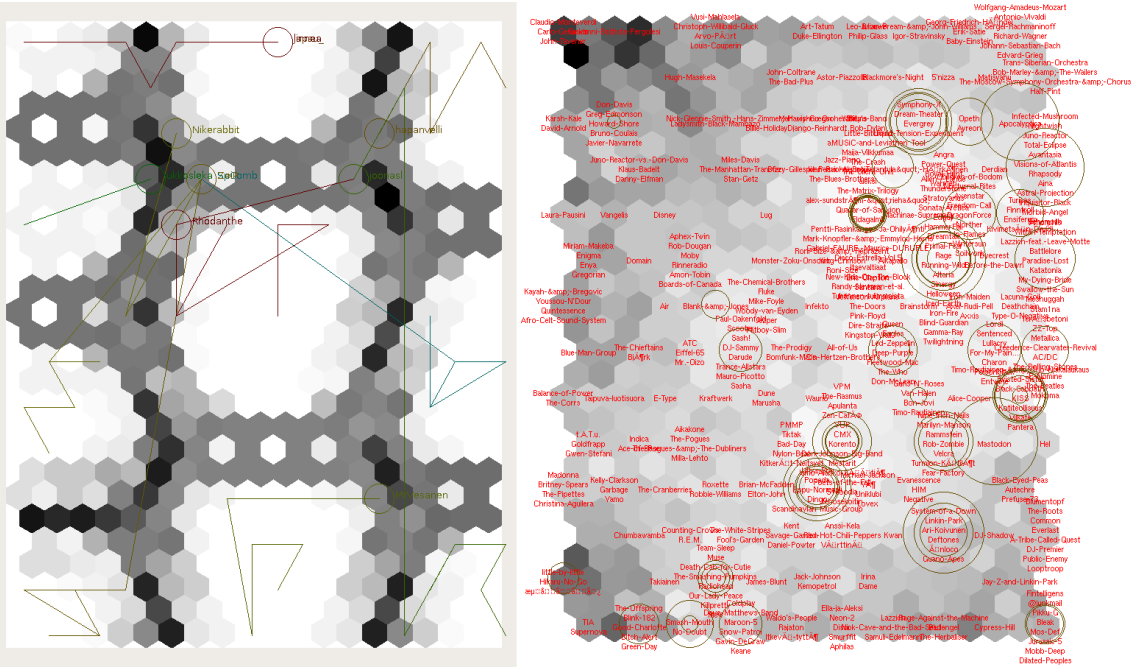

Figure 1: Visualization of the simulation. On the left there are agents marked with circles. Trajectories show how agent's practices have changed during the simulation. All the practices are shown on the right. Circles visualize practice distribution of an agent. The bigger the circle the more time agent uses on the corresponding practice.

2.8 Software Architecture

The self-organizing maps are trained with the SOM Toolbox [12] which is a Matlab implementation of the SOM. The other parts of the simulation are implemented in Java programming language [13]. Java suits well for a multiagent simulation for it is an object-oriented programming language. Objects, like agents, have state and behavior [14] and therefore it feels natural to handle agents as objects. Java program can also be turned into a web application rather easily which is useful if we want to, for example, create a recommender system that is available in the Web.

The source code is divided into four packages (see figure 2): `read` (contains classes for reading and parsing input data from files and from the Internet), `simulation` (main structure of the simulation model), `som` (self-organizing map specific classes), and `ui` (user interface classes).

The most important classes in the `simulation` package are: `Agent`, `Practice`, `Recommender`, and `SimulationRoom`. `Agent` and `Practice` model the agent and practice entities. `SimulationRoom` is sort of a pool for the agents and it takes care of running the simulation. `Recommender` class contains the functionality for checking which new practices agents have adopted.

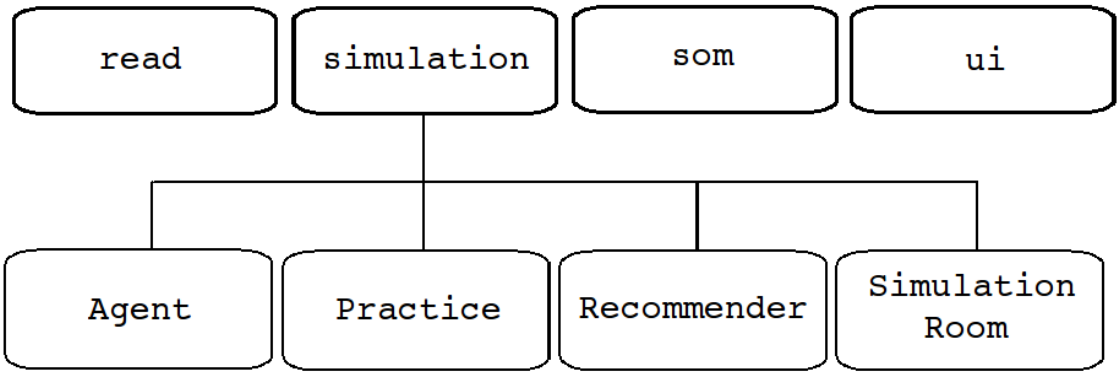

Figure 2: On the top: the package structure of the program. On the bottom: the most important classes in the `simulation` package.

3 Recommender System

3.1 Overview

The task of a recommender system is to help a person to decide what products to buy, which movie to see, etc. Because of the huge amount of choices, it is often impossible to evaluate all the possibilities and then make a sound decision. Recommender systems help by offering a small subset of possibilities a person might be interested in. [2][8]

3.2 Collaborative and Content-based Recommendations

Collaborative filtering (CF), content-based and hybrid systems are the three main categories for recommender systems. CF systems are based on the idea that people usually consult other people with similar tastes when deciding, e.g., what music to listen to. CF methods build a profile for all persons so that the similarity can be measured. The profile can be built based on the ratings the person has given or, for example, the sites he or she has visited. [2]

There are two common approaches for CF: memory-based methods and model-based methods. Memory-based methods look for people with similar tastes and then recommend the items they have liked. Model-based methods build, e.g., statistical models or neural networks which are trained with the profiles of the people and the ratings they have given to items. After these models have been trained they can be used to give recommendations for all kinds of profiles. [8]

Content-based systems use information about the properties of the items [2]. Items that match the person's own preferences are recommended to the person. For example, if a person has watched Titanic and Casablanca and rated them well, a content-based system might recognize that he or she likes romance movies and recommend Shakespeare In Love.

The advantage of CF is that nothing needs to be known about the properties of the items that are recommended, only how other people have liked those. This allows CF to be applied to various items if only user data is available. The advantage, and limitation as well, of content-based systems is that they do not use information about other persons. Therefore, they will never recommend items that are totally different from the items a person has previously rated or purchased, even though the person might like them [2]. On the other hand, content-based systems can deal with the cold-start problem, i.e. they are able to recommend new items that have not yet been given enough ratings by other persons [2].

3.3 Multiagent Simulation Approach

Our method is a hybrid approach since it takes into account both other agents and similarities of practices. Simulation takes a group of people and uses data about the practices of the group and the properties the practices. The system then simulates what new practices persons would adopt and proposes those practices to them. When a new practice is established, the system recommends it. We assume that a practice is established when the time used for it has exceeded half of the average time used for the initial practices.

Normally, recommender systems use rating data but our approach is suitable for more general data such as listening times. It can also utilize ratings if we assume that a rating positively correlates to the time used for the practice. That is, the better the rating, the more time the person will use for the practice.

4 Simulation Case: Last.fm

We first introduce the Last.fm service. Then, to demonstrate how the simulation and the recommender system work, we analyze two example runs which use the same Last.fm data but different adaption methods and visualization.

4.1 Overview

Last.fm is a free Internet radio service that tracks listenings of the registered users [15]. It has its own system for giving recommendations to the users and streaming music they might like. Over 15 million different people use the service at least once in a month [16].

Users can tag artists. Tags are not chosen from a predefined list but filled in a text form. A user can also add another user as friend and a mutual friendship is formed if the other user accepts the request. The friendships can be only mutual, i.e. if the other user does not accept, or ignores the request, no friendship is formed.

Audioscrobbler [4] is a data service which provides access to the Last.fm data. It can give, for example, the most listened artists by a user or the friends of the user. Data is given in XML format, which makes it easy to parse.

4.2 Example Run 1

4.2.1 Setup

We take a user group selected from Audioscrobbler database, their 50 most listened artists and the tags of the artists as data. Since Audioscrobbler does not have random access to the users, we pick one of my friends, *Japsu_*, and take the nine first users in his friend list to form the user group (the author is not among the nine himself). Japsu_ is chosen because he has several friends in Last.fm so we get a big enough sample.

We analyze the example run in six parts: 0, 10, 200, 1500, 3200 and 6100 steps. We show the visualization after each part and focus especially on the user *Rhodanthe's* listenings. Therefore, all the maps on the right (see appendix A) show the artists Rhodanthe listens to. The agent maps on the left have very clear clusters. This is because the map size is too big for the training data which consists of only 10 input vectors - the initial practice vectors of the agents. However, using oversized maps, we are able to see even small changes in agents' practice vectors. Except for the size, the maps are trained with the default parameters in the SOM Toolbox.

In this run we use adaption method 2. The other simulation parameters are: $\alpha = 0.2$, $w_d = 1$, $w_f = 0$, $w_{d2} = w_t = 0.5$. As w_f is zero, all influence values are equal.

4.2.2 Analysis of Results

In the initial state all agents are located uniformly around the map. After 10 steps each of them have stirred a little bit (see figure 4).

After 200 steps Rhodanthe has approached *hapanvelli* (see figure 5). He has also adopted several new artists and the top three of them are The Prodigy, with 92 plays, John Coltrane, with 69 plays, and ATC with also 69 plays. All of these exceed the half of the average number of plays, 58, and thus they are recommended to Rhodanthe. hapanvelli listens to The Prodigy and John Coltrane so it's likely that Rhodanthe has adopted those from him. ATC is adopted from *Nikerabbit* since he is the only one in the group who listens to it.

After 1500 steps (figure 5) we can see that the agents have assembled into two groups except for *joonasl* who is still hanging in his initial cluster in the down-right corner.

Finally, after 3200 steps (figure 6) all agents have gathered into the same cluster and the variety of artists has clearly decreased. In 6100 steps Linkin Park grows so big that it drops all the other artists. Agents are still not in the exact same cell on the map because they do not listen to Linkin Park equally much.

Obviously, the outcome of the simulation, that everybody would listen to only Linkin Park in the end, is absurd. Therefore, there should be a mechanism to prevent the big artists from knocking off the small ones. The other adaption method takes care of this problem.

4.3 Example Run 2

4.3.1 Setup

We use the same data set as in the first run. However, now we have different influence values for the agents. An influence value is calculated as the ratio of the number of friends an agent has and the maximum number of friends the agents have. The influence values can be observed from the circle sizes of the agents.

In this example, we use a smaller map size in order to get a better agent map but in consequence there is less movement on the map. Therefore, the run is analyzed in only four parts. The practice map also looks a little bit different since it has more cells. It also normalizes the distances logarithmically so that even the small distances are distinguishable (see appendix B).

In this run we use adaption method 1. The other simulation parameters are: $\alpha = 0.2$, $w_d = w_f = w_{d2} = w_t = 0.5$.

4.3.2 Analysis of Results

The visualization in the initial state is shown on the top of the figure 7. First movement, shown on the bottom of the same figure, appears only after 200 steps.

Rhodanthe has not yet moved but he has already adopted new practices so much that they are recommended to him: Boards of Canada with 103 plays, Freedom Call with 96 plays and Björk with 93 plays. None of these artists is same than those that were recommended to Rhodanthe in the first run. This indicates that the selection of the adaption method can have a significant impact on the outcome.

After 1500 steps Rhodanthe has still not moved but other agents have. After 3000 steps agents have settled down to their final practices shown in the figure 8. Agents have the same artists but they are still not in the same cell because they have different total listening times. In the top three of the artists are: HIM, Linkin Park and Machinae Supremacy.

4.4 Stability of Predictions

We can not calibrate the simulation parameters and validate the predictions since we are lacking history data. Simulation is stochastic and therefore we examine how well it converges to the same outcome states, i.e. how stable it is, when given the same initial parameters and run several times. We use the coefficient of variation as the measure.

To examine the stability, we use user *Rj* and his 84 friends as the agent group and the ten most listened artists of each as the practice group. We test ten different setups where we change the number of simulation steps and the adaption method (sec. 2.4.3) which agents use to increase their practice times. The results are shown in the table 4.

Table 4: Convergence results. The metric is the coefficient of variation. n is the number of simulation steps and methods refer to the two adaption methods which agents use to adapt their practice times.

	n=10	n=100	n=1000	n=3000	n=6000
Method 1	14.0%	31.7%	31.8%	16.2%	6.3%
Method 2	31.1%	6.8%	4.7%	0.2%	0.1%

The results show us that with method 2 simulation converges faster. This is because with method 2 simulation ends up in a situation where one practice has superseded all the others, as shown in the example run 1 (sec. 4.2). Because of the randomization, the one practice differs between the simulation runs and the outcomes become different. However, since the CV is calculated as the mean of each element's CV, it is close to zero even if the one practice varies.

4.5 Visualization of Artist Clusters

As a by-product, our visualization framework proves to be able to produce interesting artist maps representing the mutual relationships of the artists. Figure 3 shows a SOM which uses Finland's top artists as data. In the picture the clusters are defined and drawn by hand. They could be given, e.g., the following tags: 1. Finnish/Swedish metal; 2. Heavy metal; 3. Finnish rock; 4. Rock; 5. Classic rock; 6. Alternative rock.

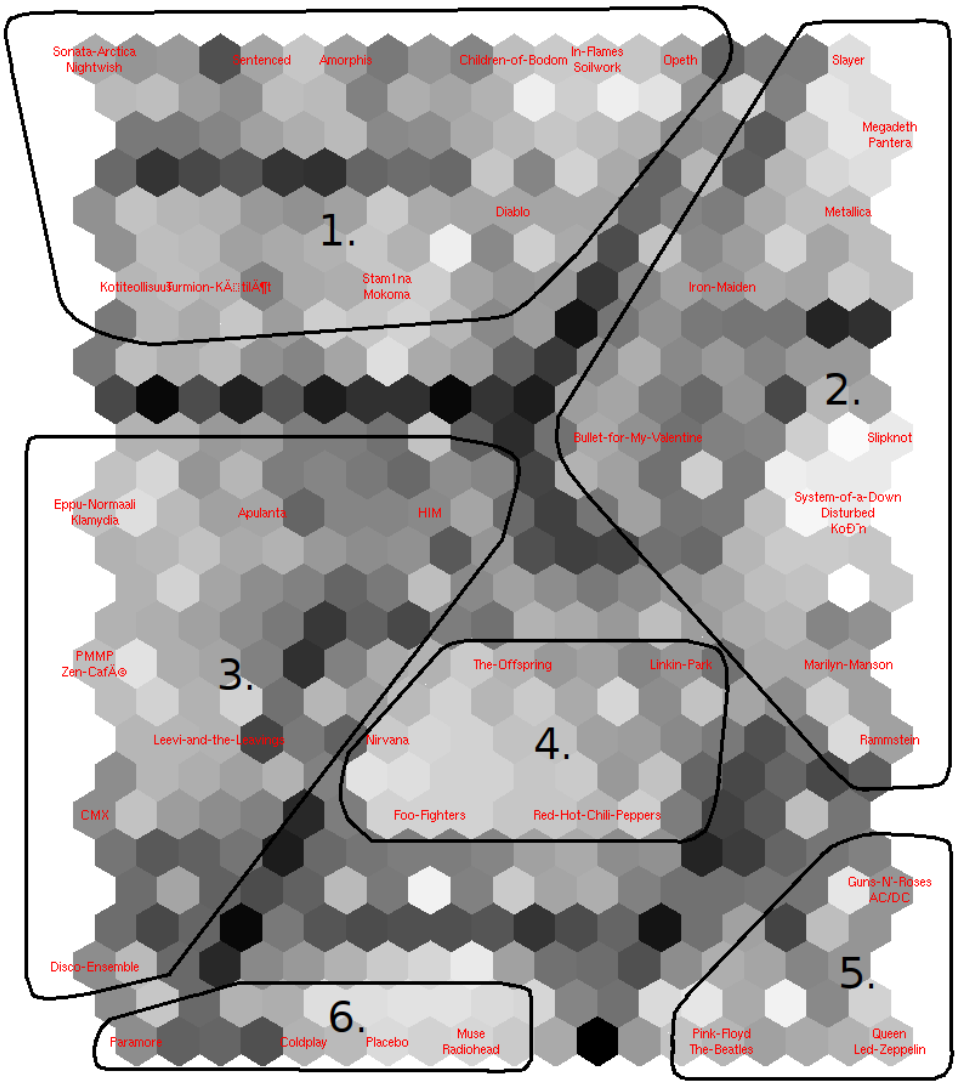

Figure 3: Self-organizing map of Finland's most listened artists for a period of one week.

5 Conclusions

5.1 Strengths and Weaknesses

Our framework has the advantage of being applicable to several different simulation cases. We first started with hobbies as practices and took the data from Wikipedia articles. Later, we decided to move from Wikipedia to Last.fm since Last.fm would offer us proper agent data as well. This change required us to implement only a new parser to get the data in the right format from Last.fm.

On the other hand, excess generality has the risk of making simulation so vague that it cannot be applied to any real life problems. To prevent this, our framework is built so that new interaction models are easy to define.

In our experiments with Last.fm data, the interaction models are quite artificial and not based on any sociological research. For example, they do not take into account that new artists, nobody in a user group yet listens to, occur. Therefore, even if we had the history data for calibrating our model, the calibration would not be successful. This is because, in a long term, new hit artists would rise up in real life but our model could not predict them.

Our agent model is also oversimplified since it takes into account only the practice times and the influences between agents. This diminishes diversity and leads to outcomes where all agents have the same practices.

Since the predictions of our simulation are not quite reliable, the recommender system cannot be considered reliable either. Nevertheless, given proper models it could prove useful, especially because it has the visualization supporting the recommendations. Herlocker et al. [17] argue that current recommender systems are black boxes, and if added some transparency, users would accept the recommendations better. Our visualization should help the users to understand the process behind simulation and recommendations, thus adding to the credibility of the system.

Our original idea was not to create a reliable predictor but a functional simulation platform. In this sense we have succeeded as our example runs show. Building a recommender system was not in the original plan either but it came almost as a by-product with the simulation framework. I got the idea for it when I started to use Last.fm data and with brief pre-study it turned out to be an intriguing research topic.

In general, multiagent simulation feels a natural approach to the problem since we do not have to make any generalized assumptions about the whole societies. We only define the interaction models and give the data about agents. Then we can observe if the simulation follows the reality and if not, recalibrate the parameters in the models.

5.2 Future Work

To obtain reliable simulation results, we would need to calibrate our models and this requires history data. We could set up a crawler that downloads the current state of an agent group between certain intervals. However, the interaction models and agent models should first be rebuilt so that they would base on some existing research results.

One way to improve the agent models could be to utilize the diffusion of innovations theory. This theory classifies agents to early adopters, secondary adopters, tertiary adopters, etc., based on how fast they adopt new innovations [18]. However, we do not have the data required for classification but possibly the history data could be used for that too. In general, more detailed data about the agents would be needed. We could do this by looking for new data sources or maybe setting up a survey of our own, which we would use for collecting all the necessary information about a group of people.

References

[1] Wooldridge, M. Lecture Slides for An Introduction to Multiagent Systems. Available at: http://www.csc.liv.ac.uk/~mjw/pubs/imas/distrib/pdf-index.html, accessed 25 August 2008.

[2] Adomavicius, G., Tuzhilin, A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734-749, Jun., 2005

[3] Pantzar, M. Shove, E. The Choreography of Everyday Life: Towards an integratice Theory of Practice. 2008.

[4] Audioscrobbler. The Social Music Technology Playground. Available at: http://www.audioscrobbler.net/data/webservices/, accessed 4 June 2008.

[5] Lesser, V. R. Cooperative multiagent systems: A personal view of the state of the art. IEEE Trans. Knowledge Data Eng., vol. 11, pp. 133-142, Jan. 1999.

[6] Klügl, F. Multi-Agent Simulation. Lecture slides. 2004. Available at: http://ki.informatik.uni-wuerzburg.de/~kluegl/teach/EASSS2004%20Multi%20Agent%20Simulation.pdf, accessed 25 August 2008.

[7] Kohonen, T. Self-Organizing Maps. Springer Series in Information Sciences, 30, Springer, 2001.

[8] Polčicová, G. Topographic Organization of User Preference Patterns in Collaborative Filtering. Doctoral thesis, Slovak University of Technology in Bratislava Faculty of Informatics and Information Technologies, 2004.

[9] Weisstein, Eric W. Least Squares Fitting. From MathWorld–A Wolfram Web Resource. Available at: http://mathworld.wolfram.com/LeastSquaresFitting.html, accessed 21 August 2008.

[10] Dogra, Shaillay K., Coefficient of Variation. From QSARWorld–A Strand Life Sciences Web Resource. Available at: http://www.qsarworld.com/qsar-statistics-coeff-variance.php, accessed 18 August 2008.

[11] Spector, L., and Klein, J. 2002. Evolutionary dynamics discovered via visualization in the breve simulation environment. In Proceedings of the Workshop 'Beyond Fitness: Visualising Evolution' at The 8th International Conference on the Simulation and Synthesis of Living Systems, Artificial Life VIII.

[12] Adaptive Informatics Research Centre: SOM Toolbox, Available at: http://www.cis.hut.fi/projects/somtoolbox/, accessed 28 October 2008.

[13] Java, 2007. Java Platform, Standard Edition 5.0. Sun Microsystems, Inc. Available at: http://java.sun.com/javase/6/docs/, accessed 21 August 2008.

[14] Sun Microsystems Inc. What Is an Object? Available at: `http://java.sun.com/docs/books/tutorial/java/concepts/object.html`, accessed 21 August 2008.

[15] About Last.fm Available at: `http://www.last.fm/about`, accessed 21 August 2008.

[16] Aukia, J. Bayesian clustering of huge friendship networks. Master's thesis, Helsinki University of Technology, Department of Computer Science and Engineering, Espoo, 2007.

[17] Herlocker, J., Konstan, J.A., Riedl, J. Explaining Collaborative Filtering Recommendations. ACM 2000 Conference on CSCW. 2000

[18] Rogers, Everett M. Diffusion of Innovations, Fifth Edition. New York, 2003.

Appendix A

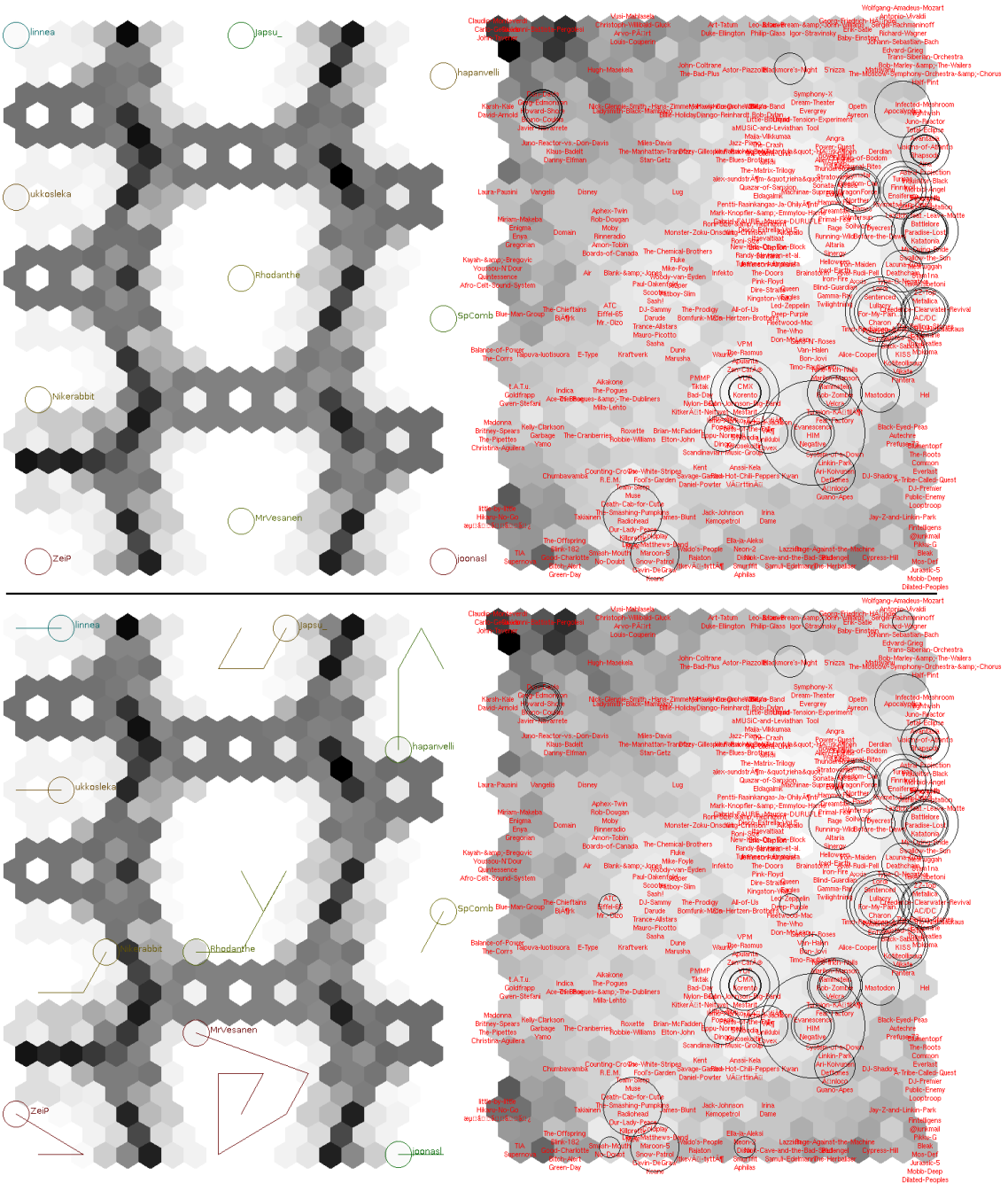

Figure 4: On the top: simulation in its initial state. On the bottom: simulation after 10 steps. Maps on the right visualize the listening distribution of Rhodanthe.

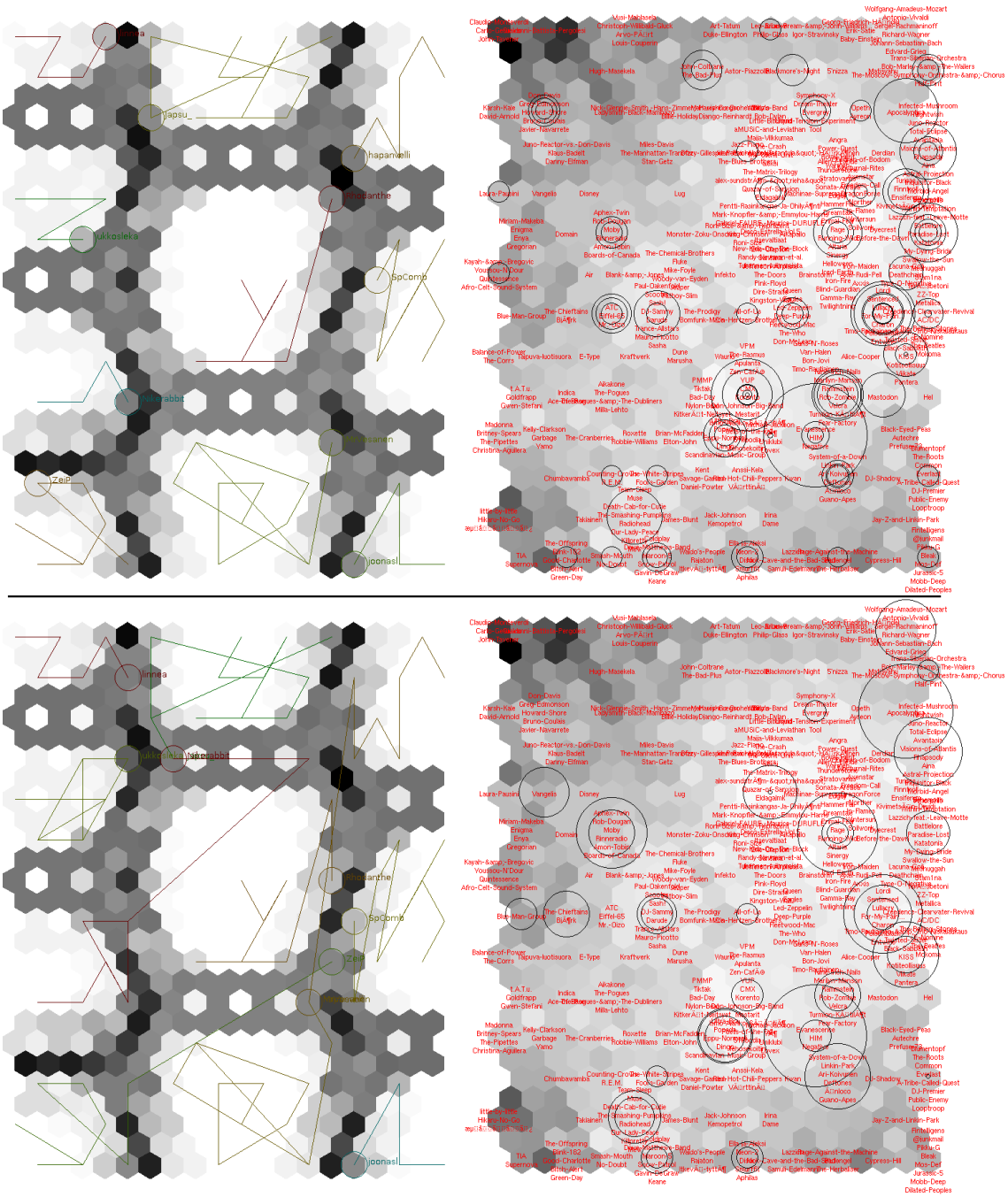

Figure 5: On the top: simulation after 200 steps. On the bottom: simulation after 1500 steps.

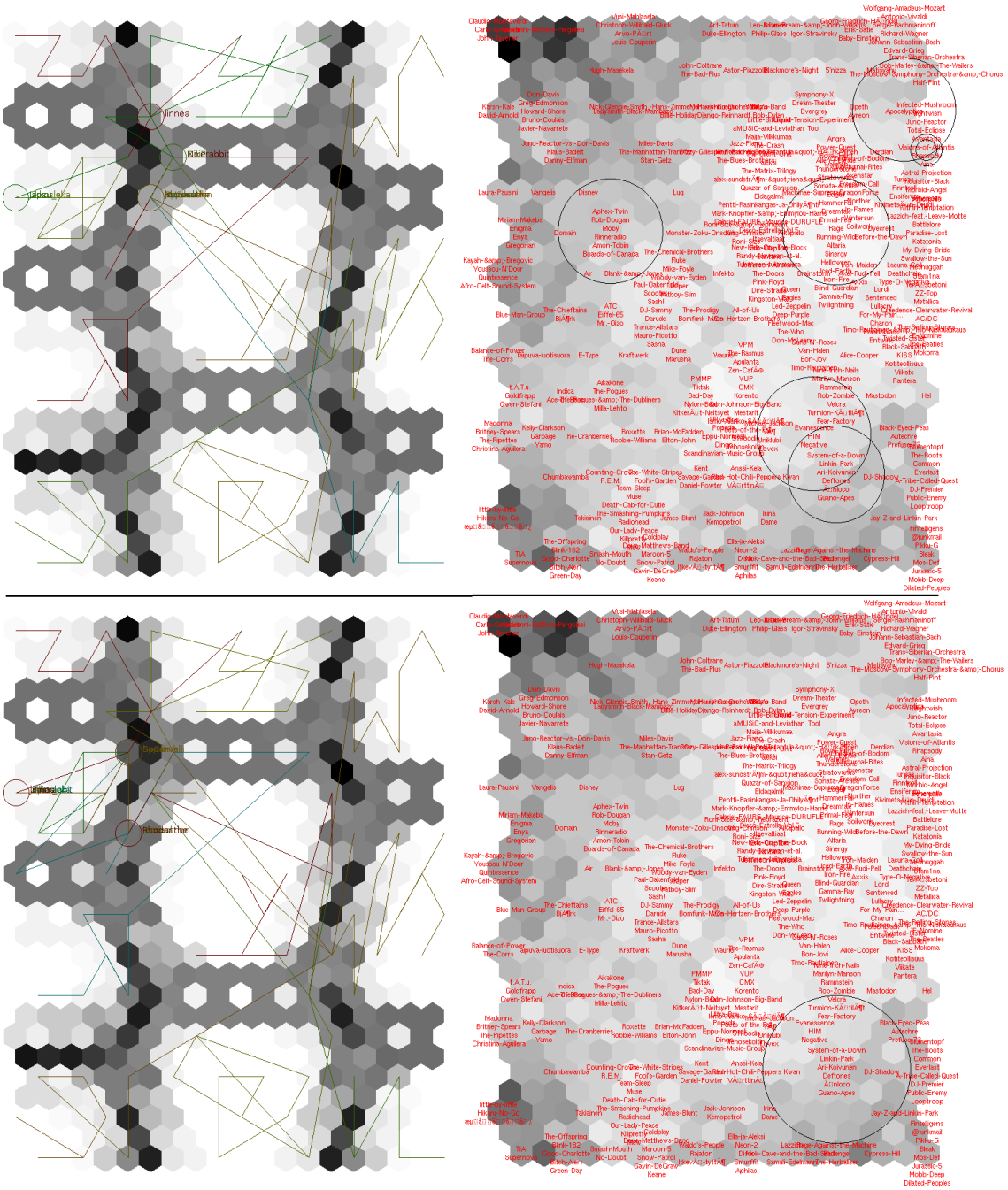

Figure 6: On the top: simulation after 3200 steps. On the bottom: simulation after 6100 steps.

Appendix B

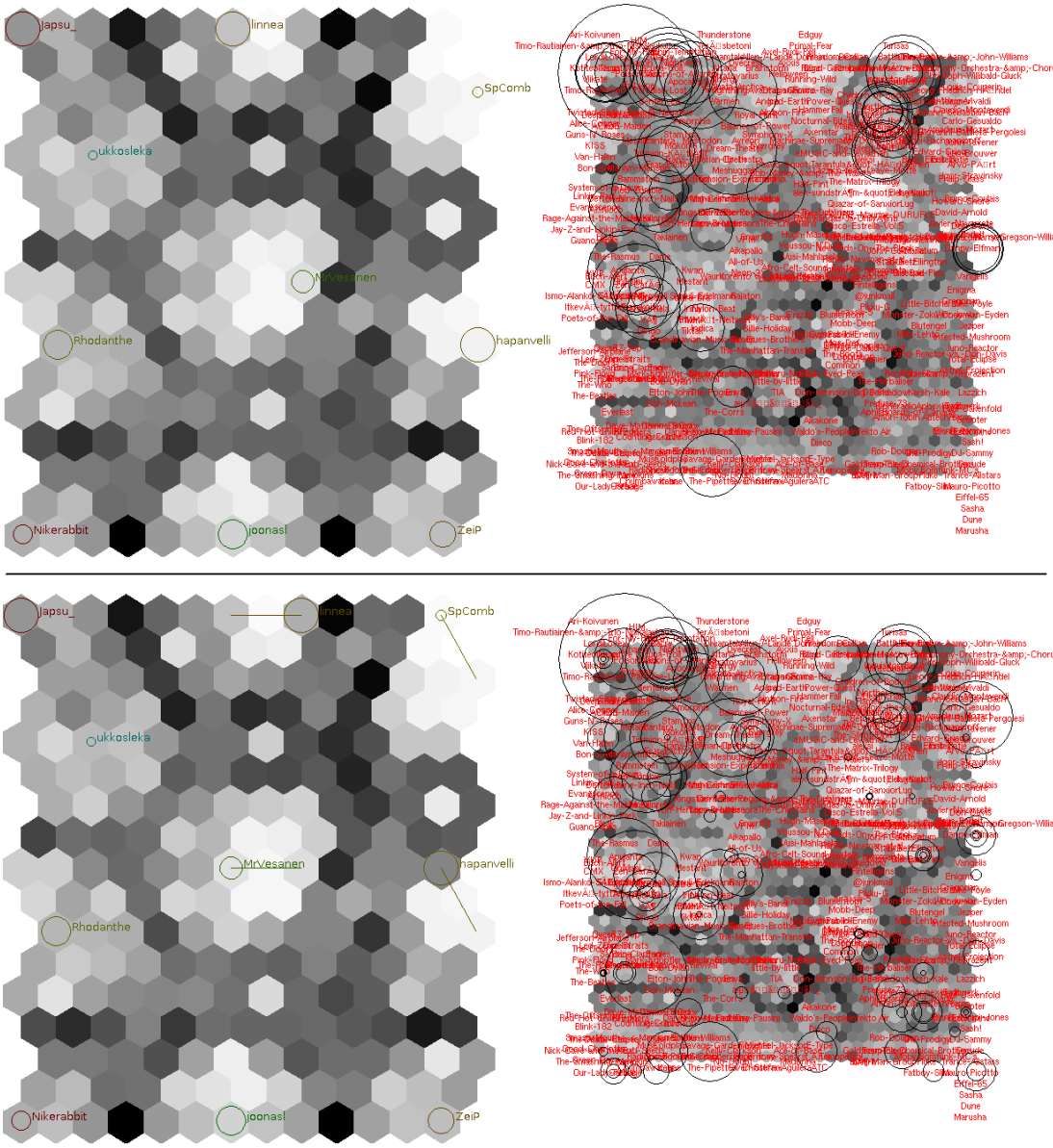

Figure 7: On the top: simulation in its initial state. On the bottom: simulation after 200 steps. Maps on the right visualize the listening distribution of Rhodanthe.

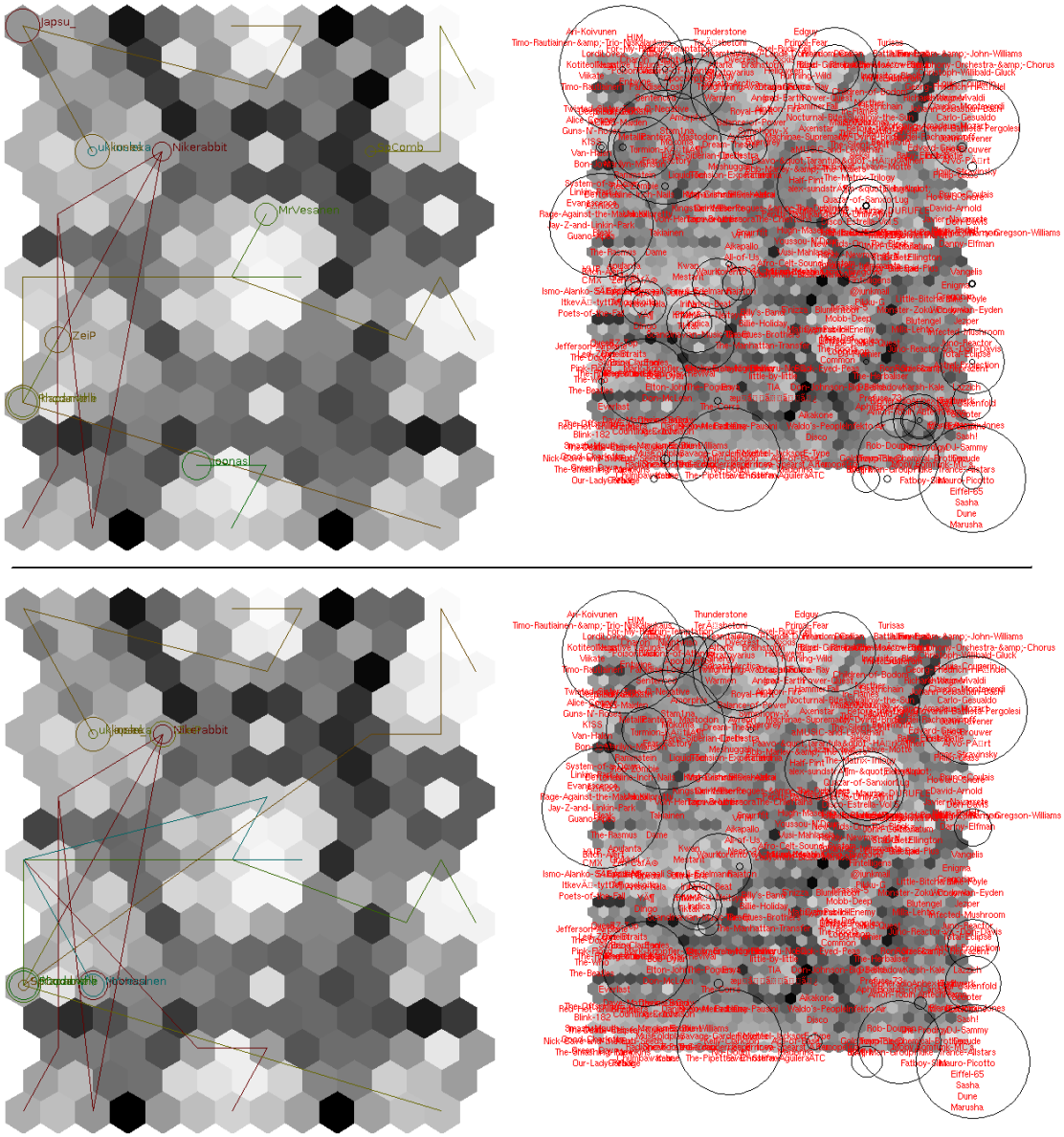

Figure 8: On the top: simulation after 1500 steps. On the bottom: simulation after 3000 steps.

Eric Malmi

Kandidaatintyön yhteenveto

Monet tieteenalat, kuten sosiologia ja kuluttajatutkimus, pyrkivät mallintamaan ihmisen käyttäytymistä. Nykyään tietokoneet tarjoavat hyödyllisen apuvälineen mallintamiseen, sillä tietokoneille voidaan rakentaa simulaatioita, joiden avulla pystytään tutkimaan ihmisen käyttäytymistä ilman työläitä ja kalliita ihmiskokeita. Simulaatioita varten tarvitaan sosiaalista aineistoa ihmisten käyttäytymisestä, ja tätä on saatavilla hyvin esimerkiksi internetin verkostoitumissivustojen kautta.

Tätä työtä varten on rakennettu stokastiseen moniagenttisimulointiin perustuva simulaatiokehys, jolla voidaan simuloida sosiaalisia vuorovaikutussuhteita. Kehys käsittelee agentteja, eli tässä yhteydessä ihmisiä, sekä agenteilla olevia käytäntöjä, joilla tarkoitetaan esimerkiksi ihmisten kuuntelutottumuksia tai muita harrastuksia. Simulaatioon syötetään agenttien alkutila, minkä jälkeen se alkaa simuloida sitä, kuinka agentit omaksuvat uusia käytäntöjä toisiltaan ja luopuvat vanhoista. Lopputuloksena saadaan ennustus esimerkiksi siitä, minkälaista musiikkia agentit kuuntelevat tulevaisuudessa.

Kehyksen on tarkoitus soveltua useaan erilaiseen simulointitarpeeseen, joten agentteja ja käytäntöjä koskeva data esitetään yleisesti vektoreina. Jokaista agenttia varten tarvitaan käytäntövektori, jonka alkiot kuvaavat eri käytäntöihin varattua aikaa. Käytäntövektorilla voidaan kuvata siten esimerkiksi eri harrastusten viikoittaisia harrastusaikoja. Jokaista käytäntöä varten tarvitaan puolestaan ominaisuusvektori, joka kuvaa käytäntöön liittyviä ominaisuuksia ja sitä, kuinka oleellisia eri ominaisuudet ovat käytännölle.

Agenttien väliset vuorovaikutusmallit voidaan myös määritellä vapaasti. Kehykseen on toteutettu valmiiksi yksinkertainen stokastinen malli, joka perustuu ajatukseen, että henkilöt vuorovaikuttavat todennäköisimmin samanlaisten henkilöiden kanssa. Algoritmi toimii siten, että agentit valitsevat ensin kukin itselleen vaikuttajan, jolta alkavat omaksua jotain käytäntöä. Vaikuttaja valitaan satunnaisesti muista agenteista, mutta kuitenkin niin, että painotetaan niitä, joiden käytäntövektori on lähellä agentin omaa käytäntövektoria. Agenteille voidaan määrittää lisäksi sosiaalista statusta kuvaava lukuarvo, jota käytetään myös painotuksessa.

Vaikuttajan valinnan jälkeen valitaan yksi sen käytännöistä. Käytäntö voi olla mikä tahansa vaikuttajan käytännöistä, mutta valinnassa painotetaan niitä käytäntöjä, joihin vaikuttaja käyttää eniten aikaa, ja jotka ovat lähellä agentin omia käytäntöjä. Kun käytäntö on valittu, lisää agentti tälle käytännölle varaamaansa aikaa sekä vastaavasti pudottaa saman ajan pois joiltain muilta käytännöiltä. Edellä esitellyn menetelmän voidaan ajatella mallin-

tavan tilannetta, jossa henkilö tapaa esimerkiksi jonkin itselleen läheisen ystävän, ja he alkavat puhua molempia kiinnostavasta musiikinalasta. Ystävä kertoo henkilölle jostain uudesta alan artistista, minkä seurauksena henkilö päättää alkaa kuunnella tätä artistia.

Simulaation etenemistä havainnollistetaan itseorganisoivilla kartoilla. Itseorganisoiva kartta on ohjaamattomaan oppimiseen perustuva neuroverkkoalgoritmi. Se kuvaa annetun vektoriaineiston epälineaarisesti kaksiulotteiselle kartalle, ja näin ollen sitä voidaan käyttää moniulotteisen aineiston visualisointiin. Simulaatiokehys käyttää kahta itseorganisoivaa karttaa, joista toinen havainnollistaa agenttien ja toinen käytäntöjen välisiä suhteita. Kartoille syötetään käytäntö- ja ominaisuusvektorit, minkä jälkeen ne oppivat kuvaamaan vektorit kartalle. Kun agentit alkavat vaikuttaa toisiinsa ja muuttavat käytäntövektoreitaan, ne kuvautuvat uusiin soluihin kartalla, joten vuorovaikutus havaitaan agenttien liikkeenä kartalla.

Jotta simulaatio voisi antaa järkeviä ennustuksia, täytyy mallin parametrit kalibroida. Kalibrointi suoritetaan historia-aineistolla, joka kuvaa sitä, kuinka agenttien käytännöille varaamat ajat ovat muuttuneet ajan myötä. Parametreja muutellaan niin kauan, kunnes simulaation antamat ennustukset saadaan sovitettua mahdollisimman hyvin kerättyyn historia-aineistoon. Aineistoa voidaan käyttää myös ennustusten todentamiseen siten, että jaetaan aineisto kahteen joukkoon. Toisella aineistojoukolla kalibroidaan parametrit ja sen jälkeen tutkitaan, kuinka hyvin malli ennustaa toista joukkoa.

Simulaatiokehyksen yhteyteen on rakennettu myös suosittelijajärjestelmä, joka suosittelee käyttäjille niitä käytäntöjä, joista he saattaisivat olla kiinnostuneita. Esimerkiksi useat verkkokaupat, kuten Amazon.com, hyödyntävät tällaisia järjestelmiä ehdottaessaan asiakkaalle tuotteita, jotka saattaisivat kiinnostaa asiakasta. Perinteiset suosittelijajärjestelmät voidaan jakaa kahteen alakategoriaan: yhteisöllisen suodatuksen (collaborative filtering) menetelmiin sekä sisältöpohjaisiin (content-based) menetelmiin. Yhteisöllisessä suodatuksessa järjestelmä etsii käyttäjän kanssa samanlaisia käyttäjiä ja suosittelee tuotteita, joista nämä samankaltaiset käyttäjät ovat pitäneet mutta joita käyttäjä itse ei ole vielä kokeillut. Sisältöpohjaiset menetelmät etsivät puolestaan tuotteita, jotka ovat samankaltaisia niiden tuotteiden kanssa, joista käyttäjä on aiemmin pitänyt.

Työssä esitellyn suosittelijajärjestelmän ideana on se, että ensin ennustetaan simuloimalla, mitä käytäntöjä agentit omaksuvat, ja sen jälkeen suositellaan näitä käytäntöjä agenteille. Järjestelmä yhdistelee yhteisöllistä suodatusta ja sisältöpohjaisia menetelmiä, sillä se huomioi sekä samankaltaiset agentit että samankaltaiset käytännöt.

Järjestelmän toiminnallisuutta esitellään analysoimalla kaksi esimerkkiajoa, jotka käyttävät Last.fm-aineistoa. Last.fm on internetradio, joka tallettaa käyttäjien kuuntelutottumuksia. Last.fm-palvelun yhteyteen on rakennettu Audioscrobbler-palvelu, joka tarjoaa pääsyn Last.fm-aineistoon ilmaiseksi. Audioscrobblerin kautta voi ottaa selville esimerkiksi käyttäjän eniten kuuntelemat kappaleet, artisteille annetut tagit ja käyttäjien palvelussa muodostamat ystäväverkostot. Esimerkkiajoissa käytetään käytäntöinä artisteja. Käytäntöjen ominaisuuksina käytetään artisteille annettuja tageja.

Esimerkkiajoja varten valitaan yksi käyttäjä Last.fm-palvelusta ja otetaan tämän käyttäjän yhdeksän ensimmäistä ystävää. Toisessa ajoista määritetään agenteille myös sosiaaliset statukset olettamalla, että agentin sosiaalinen status on sitä suurempi, mitä enemmän agentilla on palvelussa ystäviä. Simulaatioita ajetaan niin kauan, kunnes agentit ovat saavuttaneet jonkinlaisen tasapainotilan.

Esimerkkisimulaatiot suppenevat tasapainotiloihin, joissa kaikki agentit kuuntelevat samoja artisteja. Tämä on luonnollisesti epäuskottava ennuste, joten simulaatio on osittain puutteellinen. Simulaation parantamiseksi tarvittaisiin paremmat vuorovaikutusmallit, jotka pohjautuisivat oikeisiin tutkimustuloksiin. Ongelmana on myös se, ettei ole saatavilla historia-aineistoa, jolla voitaisiin kalibroida simulaation vapaat parametrit sekä todentaa sen antamat ennusteet.

Historia-aineiston keräämiseksi voitaisiin toteuttaa pieni ohjelma, joka lataisi säännöllisin väliajoin jonkin käyttäjäjoukon sen hetkiset kuuntelutottumukset. Esimerkkiajojen perusteella voidaan kuitenkin sanoa, että alkuperäinen tavoite, eli toimivan simulointikehyksen toteuttaminen, on onnistunut. Suosittelijajärjestelmän toteuttaminen ei kuulunut myöskään alkuperäiseen suunnitelmaan, vaan se syntyi ikään kuin simulaatiokehyksen sivutuotteena.