The Impact of Recommender Systems on Item-, User-, and Rating-Diversity

W. Kowalczyk, Z. Szlávik, and M.C. Schut

Department of Computer Science
VU University Amsterdam (NL)
{w.j.kowalczyk,z.szlavik,mc.schut}@vu.nl

Abstract. Recommender systems are increasingly used for personalised navigation through large amounts of information, especially in the e-commerce domain for product purchase advice. Whilst much research effort is spent on developing recommenders further, there is little to no effort spent on analysing the *impact* of them – neither on the supply (company) nor demand (consumer) side. In this article, we investigate the diversity impact of a movie recommender. We define diversity for different parts of the domain and measure it in different ways. The novelty of our work is the usage of real rating data (from Netflix) and a recommender system for investigating the (hypothetical) effects of various configurations of the system and users' choice models. We consider a number of different scenarios (which differ in agents' choice models), run extensive simulations, analyse various measurements regarding experimental validation and diversity, and report on selected findings. The scenarios are an essential part of our work, since these can be influenced by the owner of the recommender once deployed. This article contains an overview of related work on data-mining, multi-agent systems, movie recommendation and measurement of diversity; we explain different agents' choice models (which are used to simulate how users react to recommenders); and we report on selected findings from an extensive series of simulation experiments that we ran with real usage data (from Netflix).

1 Introduction

Recommender engines have made an overwhelming entrance in the world of digital information. Here, electronic commerce is a front-runner, hoping to sell more products with effective recommenders. Still, not only commerce, but many owners of other information systems also start to realise that recommenders may benefit the users in finding items of interest. Whether recommendations are generated by algorithms (e.g., Amazon, Netflix) or in a more social way (e.g., Facebook, Twitter), owners ultimately want to know if their users can better and faster navigate through the available information. On commercial sites, this means whether users buy more items.

Besides tackling technical questions, e.g., on deployment, up-time and responsiveness guarantees and algorithmic optimisation, owners of a recommender want

L. Cao et al. (Eds.): ADMI 2011, LNAI 7103, pp. 261–287, 2012.

to know how users react to their engine, and, following from that, how they or the engine can react to these user reactions. This will enable them to dynamically adapt to the customer within the purchasing process – similar to how a human salesperson pitches in order to sell a product.

In this article, we look at the interaction effects between recommendation and user behaviour. We test a movie recommender in combination with a variety of scenarios (i.e., the way that users react to suggestions of the recommender engine) and report on the observed interaction effects. In particular, we look at *diversity* effects, i.e., the property of being *numerically distinct*. We calculate diversities for users, items and ratings. Later in this article, we define different diversity measurements and calculate user-, item-, and rating-diversities for our recommender.

We are the first to conduct such a study, to our best knowledge. Whilst other works have investigated the combination of diversity and recommenders, e.g., [14,10,11], we are the first ones to 1) do this with actual usage data (from Netflix), 2) perform a variety of diversity measurements for different parts of the recommender (item-, user-, and rating-based diversity), and 3) use a combination of data-mining and multi-agent simulation techniques. Also, we are not aware of other work that investigates this issue within the domain of movie recommendation, as we do.

Our work relies on two main technologies: *data-mining techniques* in the recommendation engine, and *agent-based simulation* for the user choice model in the scenarios. Also in this respect, this article can be considered to be breaking ground. Despite the fact that agents have earlier been used for recommendation purposes (as described extensively below), we use agents in a completely different way, namely in order to analyse possible user behaviours and to give engine owners information on how to manage and react to these behaviours. This approach is also novel to the field of recommenders that is traditionally more focused on design and development of optimisation algorithms. As such, our work gives rise to new approaches for simulation-based recommender design and deployment with the user-in-the-loop (by means of agents).

The objective of the study presented in this article is to systematically investigate item-, user-, and rating-diversity effects of a movie recommender system by means of a simulation. This simulation is based on real-world usage data. With this study, we aim to obtain an insight into the effects of recommenders from the owner's perspective (i.e., how can we set up the recommender such that it generates maximum amount of added value?) as well as the consumer's perspective (i.e., what are the consequences of particular ways of reacting to the predictions that recommenders give?). In particular, we aim to answer the following three questions:

1. *What happens if we force users to rate a certain number of items in a period of time (e.g., everyone rates 5 movies a week)?* Such a restriction is an example of how the owner of the recommender can 'influence' the behaviour of users.

2. *What is the effect of changing the type of information that a recommender gives to users?* For example, a recommender can show to the user either most popular movies, or movies that match best the user's preferences, or just a random selection of movies.
3. *What is the influence of mixed groups of users in terms of how they react to recommendations?* We may assume that not all users of a recommender react similarly to a recommender. We perform simulation experiments with both homogeneous (everyone reacts the same to recommendations) and heterogeneous user groups.

It is worthwhile to mention that our work is closely related to what is also researched in social and political sciences on group dynamics and political systems [4,9,12]. In these studies, it is also researched how individuals in groups react to each other, both in direct interactions as well through an intermediate entity (the government, or, in our case, the recommender system). The scenarios that we have used are based on general social models concerning group dynamics on peer influence, social networks and collective behaviour. Extensions of our study include the investigation of the particular role of intermediators in this dynamic process (e.g., recommenders/governments).

This article is organised as follows. In Section 2 we describe related work on movie recommendation and multi-agent systems to position our work; we also present background work on which we have based our work (e.g., specific recommender algorithms). Following that, Section 3 contains the description of the experimental design and setup as well as the simulation scenarios. In Section 4, we elaborate and analyse obtained results from the simulation experiments. Finally, Section 5 concludes and provides pointers for future work.

2 Related Work

In this section, we overview the literature related to the study presented in this article. In general, we focus on agent-based models in combination with (data) mining techniques; and in particular with collaborative filtering engines for movie recommendation. We review literature on these topics among two main dimensions: recommenders and agents. For the first, we describe *design and development* – which is about actually making and designing a recommendation engine algorithm, software and/or framework; *effects of engines* – about the interaction effects between recommendation engines and consumers; and *users* – describing opinions about, the evaluation of and interfaces of recommenders. For the latter, we discuss *consumer behaviour* – on how to realistically simulate consumer behaviour; *communities* – about forming agent-user communities; and *clustering and mining* – about the use of these techniques for recommendation and collaborative filtering.

Our objective is to give a concise overview of existing work at the cross section of movie recommendation and multi-agent systems. This overview consists of a number of exemplar articles at this intercross. We finish this section with an

explanation on how our work relates to the described existing work. In the following section, we go into details of the use of data mining techniques for the Netflix movie recommendation dataset.

2.1 Recommenders

Design and Development. It is not surprising that much effort in the area of recommenders is spent on design and development, since most e-commerce shops have incorporated such technology on their websites. Later in this article, we look specifically if the recommender technology is worth investing in (intuition says it is). First, we give a brief overview of recommender design and development that we consider relevant within the context of this article, i.e., we concentrate on *multi-agent* recommender design specifically for *movies* or television.

In [8], Dehuri *et al.* propose a multi-agent system for the *multiple recommendation problem* (MRP). In an MRP, several personalised recommendations are conducted simultaneously, which can lead to *churning*: users being given many uninteresting recommendations. The proposed multi-agent system is a bio-inspired algorithm that mimics the behaviour of a wasp colony.

Another approach for sharing recommendations is proposed by Weng *et al.* [29], where multiple engines are connected to each other. Although this is not based on multi-agent technology, the developed system (the Ecommerce-oriented Distributed Recommender System, EDRS) has several agent-like properties. The EDRS consists of multiple recommenders from different organisations, in order to improve the service to users (i.e., compared to recommending in a single organisational basis). Among others, the EDRS tries to tackle the *cold-start problem*: instead of a new recommender starting from nothing, it can use collected information from other recommenders in order to form initial recommendations. The distributed variant of recommending can also be seen as an alternative for centralised recommenders that come with privacy and scalability issues, i.e., to use distributed ones to avoid central server failure and protect user privacy. The most significant contribution of the paper is a novel peer profiling and selection strategy.

Lee and Yang [19] propose another multi-agent framework, specifically suited for TV programme recommendations. The framework consists of parts for information gathering, user modelling, and system adaptation; it incorporates a decision tree-based approach (combined with an overfitting-reduction technique) to learn users' preferences. The proposed system contains different types of agents for different programme types (films and news), as well as for learning, collaboration and adaptation. The agents collect information through the set-top-box of the user, on-line (movie and news) sites, and agents of other users, in order to profile a user and offer recommendations. The collaboration with other agents is to overcome the problem of modelling users with only sparse samples.

Finally, Ono *et al.* [20] presents a recommender for constructing movie preference models. Here, data is explicitly acquired through a WWW questionnaire survey instead of collecting data implicitly (e.g., through weblog analysis).

In the work, a novel way is presented of constructing context-aware and multi-applicable preference models using Bayesian networks. Context-awareness relates to a specific situation at hand (for example in a mobile internet scenario); multi-applicability refers to item *promotion* in addition to recommendation. First experimental results show that the approach is promising, despite that the improvement of prediction accuracy is only slight (according to the authors because of the sparsity of the data used for model construction).

Algorithms. The main role of a recommender system is to provide its users with recommendations concerning possible products (e.g., movies, books or services) that would match their preferences or needs. This can be achieved with the help of various algorithms that analyse available data about products and users, such as their (product or user) characteristics, purchase history, explicit feedback provided by users in the form of ratings, product reviews. Within the last two decades many algorithms for this task have been invented. We will provide here a brief overview of the most significant types of these algorithms. A more extensive overview of recommendation algorithms can be found in survey articles of Adomavicius and Tuzhilin [1] or Su and Khoshgoftaar [27].

There are two main strategies for building recommendation engines. The first strategy, *content filtering*, is used in situations when some information about individual users or products is available. This information – attributes that describe various properties of users or products – is used to construct their profiles. These profiles are then used to assign products to users. Another, more general approach which does not rely on external information, *collaborative filtering*, uses information about the past user behaviour (such as users' purchase history) to group users that behave in a similar way and offer them similar products. In some cases both approaches can be combined, leading to *hybrid recommender systems*. There are several ways of combining content-based and collaborative methods; they are outlined in [1].

In some situations users evaluate products by assigning them a *rating* – usually an integer between 1 and 5 that expresses a user's opinion about the given product. When rating information is available, one can measure similarity between products or between users by comparing the row or column vectors of the 'user by product' matrix of ratings. In this way, two similarity measures can be defined: *item-to-item* and *user-to-user*, which can then be used for generating recommendations, Sarwar *et al.* [24].

In 2006 the DVD rental company, Netflix, announced a big competition (with a $1 million prize) that was aimed at improving the accuracy of their movie recommender system, *Cinematch* [5]. This competition attracted the attention of thousands of researchers from all over the world, leading to the development of several new approaches. Eventually, these efforts led to the improvement of the accuracy of the original Netflix recommendation engine by more than 10%. The new approaches are based on the concept of matrix factorisation of a huge matrix that contains ratings given to items by users. The concept of matrix factorisation, also called low rank matrix approximation, Singular Value Decomposition, or

latent factor model, had already been known and used earlier, for example, in the context of information retrieval [7], but it was Simon Funk (real name Brandyn Webb) who described, on his blog [13], a very simple and efficient modification of this method in the context of the Netflix competition. His algorithm (just a few lines of code!) was good enough to beat the Cinematch baseline ratings by 5%. Not surprisingly, Funk's approach attracted the attention of other competitors, which resulted in numerous improvements of the original method, and finally, in reaching the main objective of the Netflix Challenge (improvement by 10%). Several modifications of Funk's method are described in [18,28].

Effects. In this article, we concentrate on the *effects* of recommenders, rather than design and development. We are particularly interested in the impact of recommenders on item-, user- and rating-diversity, but others have looked at other effects, mainly regarding added business value. We consider three such other works here.

Firstly, Garfinkel *et al.* [14] have developed an empirical method to evaluate the relationship between recommendations and sales. It incorporates 1) indirect impact of recommendations on sales through retailer pricing, 2) potential simultaneity between sales and recommendations, and 3) a comprehensive measure of the strength of recommendations. It was found that the recommendation strengths have a positive impact on sales; sales affect the recommendations; and recommendations have a positive impact on the prices. The authors emphasise and confirm that the research on the business value of recommenders to retailers is only nascent. In the article, a research model is constructed that consists of a set of equations with sales, prices, and recommendation strength as dependent variables. Based on an empirical investigation with panel data, the aforementioned conclusions are derived.

In [10], Benjamin Dias *et al.* look at the business value of a personal recommender in the area of consumer packaged goods. The study was conducted in collaboration with a Swiss e-grocer. The analysis involved shopper penetration, and the (in)direct extra generated revenue. From the analysis it could be concluded that the effect of a recommender is larger than only the direct extra generated revenue (e.g., introducing new categories of products to shoppers). It was also found that accurately updating the model was key to generating extra revenue.

Finally, the work of Fleder and Hosanagar [11] is very closely related to the topic of our paper: it concerns the effect of recommender systems on the diversity of sales. In their work, they investigate (by means of modelling and simulation) whether recommenders help consumers discover new products, or if they reinforce the popularity of already-popular products. The study involves two stages: firstly, path-dependent effects are explored by means of analytical modelling; secondly, by means of simulation, the results are validated with a combination of specific choice models and particular recommender implementations. Results of the study are threefold: firstly, recommenders can indeed create a rich-get-richer effect and reduce sales diversity; secondly, there is a difference between

individual (consumer) and aggregate (overall) diversity: while the former decreases, the latter can actually increase; finally, different recommender design choices affect sales diversity differently.

Users. Another type of recommender effects concerns the way that users interact with the system, e.g., how do users evaluate a particular recommender, does the interface affect a consumer in particular choices, does it influence the opinions of users? Cosley *et al.* [6] investigate how users' opinions are influenced by recommenders. Two features of interfaces are looked at: the rating scale and the display of predictions at the time that users rate items. The study shows that users do not react much to different rating scales, but that they can be manipulated to tend to rate toward the system's recommendations. But users know when the systems tries to manipulate them. An experimental study with 536 users was conducted to answer questions on the way that people respond to recommendations and to see if they noticed when being manipulated. For the first, it was found that users respond to the way the predictions are shown, but it is unknown how long this change in ratings last; for the second, although users would not consciously know they were manipulated, user satisfaction suffered in case the predictions were tinkered with. The study is a first step towards knowing how to deliver accurate predictions to users in a way that creates the best experience for them.

2.2 Agents

Simulation of Consumer Behaviour. Multi-agent models can be used to model consumer behaviour in an intuitive, straightforward way. In our work reported here, we employ a multi-agent system to model choices that users make for watching movies and how they respond to recommended predictions. In the literature, more simulations on consumer behaviour can be found, of which we describe 3 articles here.

Ahn [2] presents a way to use agent-based modelling to imitate user behaviour in internet stores regarding browsing and collecting information. Through this, the rational behaviour of each user is found, which is used to simulate shopping and evaluate different *target customer aid functions*, e.g., personalised pages. Results indicate that personalised pages might not be the best aid function compared to simpler ones.

Ben Said *et al.* [21,23,22] present the CUBES project (CUstomer BEhaviour Simulator) with which one can simulate a consumer population and investigate effects of marketing strategies in a competing market. The project is on the intersection of sociology, marketing and psychology. In CUBES, one can represent observed behaviours as well as simulate large consumer populations. It is possible to simulate behavioural attitudes, impacts of consumption acts, retroactive effects of these acts, and brand reactions. It is shown in [23] through a series of experiments that macro-emergent market phenomena can be explained by behavioural attitudes such as imitation and innovation.

Finally, Ishigaki [17] models consumer behaviour with the help of real-world datasets. The underlying model is a Bayesian network (thus not multi-agent, but because it still concerns simulation of consumer behaviour, we included it here). The model assumes a large-scale dataset (e.g., weblog data, questionnaire data) and tries to explain certain consumer-related factors (e.g., satisfaction level and product valuation). The paper distinguishes different types of models which are increasingly elaborative: process models (e.g., advertisement models), utility models (including data mining) and psychological models (e.g., Stimulus-Response models, information processing models). The proposed computational model fills gaps that these other models have: it is quantative, it is constructed on actual data, and it deals with uncertainty of consumer behaviour. The model is applied in a retail service application and a content providing service (i.e., movie recommendation). For the latter, details are described in [20].

Communities. Agent communities are a big area of research in multi-agent systems. An agent community is *a stable, adaptive group of self-interested agents that share common resources and must coordinate their efforts to effectively develop, utilize and nurture group resources and organization* [26]. Sen *et al.* [26] have looked at various issues in such communities, e.g., trust-based computing, negotiation, and learning. In recent work [25,15,16], they introduce the Social Network-based Item Recommendation (SNIR) system that consists of agents that are situated in a social network to find items for their peers. The system was used to search and recommend photos on the Flickr photo-sharing social network based on the use of tag lists as search queries. In an experimental study, SNIR outperforms a content-based approach and other recommendation schemes.

Search, Clustering and Mining. The last category of related literature concerns clustering and mining techniques for web data (aka *web usage mining*) as well as search in social networks. This is a wide area of research, and we focus particularly on the combination of such techniques and multi-agent systems.

Alam *et al.* [3] present a web clustering technique for session data based on swarm intelligence, specifically particle swarm optimisation (PSO). The technique clusters on *time* and *browsing sequence* dimensions of the web usage data set. Sessions are modelled as particles for the PSO algorithm; the swarm of agents represents the complete clustering solution. It is demonstrated by means of a series of simulation experiments that the algorithm performs better than k-means clustering.

Following up on the previous category of work (agent communities), Gursel *et al.* [15] have developed an agent-based system (closely related to the beforementioned SNIR) to mine social networks of users to improve search results. In this system, agents observe users' activities as well as the ratings and comments provided by the user to items retrieved from the social network. The process refines search results based on preferences (categories $\times$ contacts), the learning of preferences with mining, category determination, and recommendation (ranking

the items). An experimental study investigates this system to see whether a user has different preferences for the same friend based on item topics. The study shows that the mechanism 1) filters search results by contacts, and 2) ranks results according to user preferences for the users who posted those items.

2.3 Summary

We overviewed related work here. Of these related works, we consider the work of Fleder *et al.* [11] most relevant. However, our conducted analysis is different from theirs in an important aspect: whilst their findings are based on theoretical conduct (still, with *actual* recommendation engines), we based ourselves on real usage data. Also, our scenarios are related to the choice models of Fleder *et al.* but are broader and more flexible: whilst the definition of Fleder *et al.* considers user behaviour (i.e., the way users react to recommendation), we also allow for the owner of the recommendation engine to change its configuration. Therefore, the 'recommender part' of a scenario concerns the *information* that is shown to the user (i.e., most popular items, random items, most relevant items). If we know how users react to specific information (e.g., buy more items if most popular items are shown), then the engine owner can anticipate on this.

Our extended notion of 'choice model' then makes the work of Cosley *et al.* [6] (described above) very relevant. Without going into the details of that study again, remember that it shows that there are specific effects of how users react to recommendations. This demonstrates that an engine owner has to put significant efforts into deciding what an engine should show to users in order to give them the best experience. For 'creating such a best experience', the specific type of *collaborative filtering* (explained below) that we use here (i.e., by means of ratings) is a complicated one. Ratings are a much-used way to elicit and show users' *preferences* for items. But these preferences are complicated – people do not necessarily always desire to maximise these in the context of recommendation. A rating expresses how good a user finds a particular item – for example, a complicated art-house movie would, in general, be considered better than an average blockbuster. However, this does not mean that users always want to watch art-house movies, e.g., after a week of hard work, one might prefer to settle for a 3-stars blockbuster. Hence, users do not always maximise rating when choosing movies. In comparison with other types of collaborative filtering (e.g., those that show what other items users have seen or bought), this aspect of preferences must always be kept in mind.

3 Experiments

In this section, we describe the data used as the basis of our experiments (Section 3.1), the recommender algorithm we used (Section 3.2), the various simulation scenarios considered (Section 3.3), the setup of our simulations (Section 3.4), and finally, the various aspects of diversity we measured (Section 3.5).

3.1 The Netflix Data

In our experiments we used the original data from Netflix and the basic variant of Funk's algorithm. Now we will describe the data and the algorithm in more detail.

The dataset used in the Netflix Challenge [5] consists of about 100 million records which represent ratings given by about 500.000 users to 17.700 movies over a period of about 6 years. Each record consists of four items: `user_id`, `movie_id`, `date`, and `rating`. Ratings are integers between 1 and 5 that reflect users' satisfaction levels (the higher the better). This dataset served as a *training set*: a recommender algorithm was supposed to use this set to learn how to predict ratings that could be given by users to movies in the future. To test the accuracy of various algorithms Netflix also provided a *test set*: around 3 million records with actual ratings (known to Netflix, but not disclosed to the public) replaced by question marks. The quality of recommender algorithms was measured using Root-Mean-Squared-Error:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - t_i)^2} \tag{1}$$

where the sum ranges over all N test records, p_i denotes the predicted rating (which is produced by the tested recommender algorithm) and t_i is the true rating that was given by the user to the movie.

3.2 The SVD Algorithm

The algorithm that we decided to use in our experiments is described in [13]. It is based on an assumption that the taste of a user can be captured by a vector of a few numbers called *user features*, while the corresponding characteristics of a movie can be expressed by another vector of the same length, *movie features*. More formally, we assume that for every user u and movie m, there are two vectors of length d: user features, f_u, and movie features, f_m. Additionally, we assume (and this is a very strong assumption!) that the preference (possible rating) of user u with respect to movie m, $p_{u,m}$, can be expressed by a dot product of both vectors:

$$p_{u,m} = \sum_{i=1}^{d} f_u(i) f_m(i) \tag{2}$$

The values of feature vectors are unknown, but they can be estimated from the training data. Let $r_{u,m}$ denote the actual rating given by user u to movie m (as recorded in the training set). Then the error made by our recommender on the training set is a function of feature vectors:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,m}(\sum_{i=1}^{d} f_u(i)f_m(i) - t_{u,m})^2} \quad (3)$$

This function, although it involves a huge number of unknowns (d times the number of users plus d times the number of movies), is relatively simple (after ignoring the sqrt symbol it becomes a polynomial), so its minimum can be found, for example, with the help of the gradient descent algorithm.

When optimal values of feature vectors of movies and users are found, Equation 2 can be used for calculating, for any user and movie, the expected rating the user could give to the movie.

In our simulations we will often have to generate predictions for users or movies that are new and do not have corresponding feature vectors. In such situations we use the following rules:

- if both the user and the movie are new then use the average rating of all available ratings,
- if only the user is new then use the average rating given to the movie by others,
- if only the movie is new then use the average rating given by the user to other movies.

3.3 Scenarios

In order to study how diversity changes under various conditions, it is important to understand the circular nature of a recommender. On one hand, the recommendations a system provides to its users shape the way users act (otherwise the system cannot be considered of good quality). On the other hand, the way users select and, in this case, rate items also affects the recommendations that the system gives in a later stage (assuming that the system is retrained regularly or adapts on-line).

This circular process, illustrated in Figure 1, can be broken down into 'rounds' where one round is defined, for our purpose, as a list of user-movie-rating tuples. A tuple indicates that a certain user gave a certain rating to a certain movie, and one round is made up by several people rating several movies. One round has a direct effect on the next round, and so on. Namely, after every round, new ratings are processed by the recommendation engine to update the rating model that will be used in the next round. In other words, the fact that the recommender system iteratively updates its model after every round has a cumulative effect on rounds over time, which is affected by user behaviour. As also shown in Figure 1, both recommender system designers and users can make several choices. For example, a system might offer the most popular movies to users, and users might either behave in an accepting manner, or watch and rate other movies.

In this paper, we select several such system-user choice pairs and investigate their impact of various aspects of the whole process. System-user choice pairs

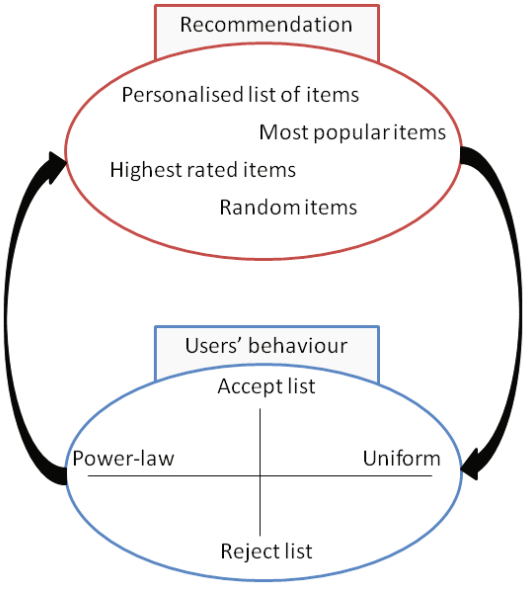

Fig. 1. The lifecycle of a recommender, and choices by system designers and users

define scenarios, and we run a simulation for each scenario. We have identified the following 10 scenarios.

1a. The first scenario is made up by the recommender providing personalised lists of items (i.e., the items that are supposed to receive the highest rating) and all users rate those, and only those, items that are recommended to them. The number of items that are recommended to a user is the same as the number of items the user rated in the true data. As the user is supposed to rate all the items offered by the recommender, we call this scenario a **Yes-men** scenario. It is well-known that in the Netflix data the number of items that are rated by individual users follows a power-law. Therefore, sometimes we will refer to this scheme as a power-law or a natural distribution.

1b. A very similar scenario is made up by, as above, providing users with personalised recommendations which are then accepted, but this time the number of movies rated in a round is distributed evenly among users. This change allows us to investigate the effect of 'uniformisation' in a certain sense. This scenario will be identified in this paper as **Uniform Yes-men**.

2a. As the first representative of another group of scenarios, we also define a scenario in which users are offered the most popular movies so far. Popularity is defined as the number of times a movie has been rated since the very beginning of the simulation. Note that this is sensitive to initialisation, but we make an attempt to make this less of an issue, as described in

the next subsection. With this scenario, as before, users accept what they are offered, and the number of movies users rate follow a power-law. This scenario is the **Popularists**.

2b. The counterpart of the above scenario is called **Uniform Popularists**, in which the power-law distribution of the previous scenario is replaced by a uniform distribution.

3a. We also define a scenario in which users accept what they are given, the number of movies rated per user is distributed according to a power-law, but this time it is the so far highest rated movies that get recommended to users. We identify this scenario as **Trend-followers**.

3b. The counterpart of Trend-followers are, obviously, the **Uniform Trend-followers**, where the power-law is replaced by a forced uniformity.

4. The next scenario is significantly different from those discussed so far. In this scenario, movies that are presented to users are selected at random. The number of items that are presented to a user is determined in the same way as in scenario 1a and follows a power-law. This scenario is called the **Randomisers**.

5-7. To also investigate mixtures of (power-law based) scenarios, we have made up three scenarios that present a transition from Yes-men to Randomisers. In these three scenarios, we define a probability with which a user either accepts what they are personally offered, or rates a random movie instead. Thus, between the 100% pure Yes-men and pure Randomisers (= 0% Yes-men), we define scenarios called **75% Yes-Randomisers**, **50% Yes-Randomisers**, and **25% Yes-Randomisers**, respectively. These scenarios are to allow us to investigate various mixtures of the two scenarios.

The following subsection describes the simulations in general, which is followed by the description of diversity measures used for the analysis of simulation results.

3.4 Simulations

The outline of a simulation is depicted in Figure 2: firstly, we initialise the recommender, then recommend movies to users, who select movies, then they rate them. Once movies in a round are rated, the recommender (user and movie profiles) is updated, and the next round of simulation starts.

A round, shown as a loop in Figure 2, represents a time period of one month, i.e. we simulate recommendations and movie ratings of one month before updating the recommender and moving on to the next one-month period. Given that we use the Netflix data for calibration (see the following paragraphs), we have 60 rounds available per scenario. The choice of one month was made so that it provided us with a sufficient number of rounds, a feasible time frame for running the simulations, and an intuitive unit of simulation rounds.

In order to keep control over the simulations and to keep them realistic, we 'calibrated' our simulations with real data. The data we used for calibration was the Netflix dataset (Section 3.1), to which we refer as the *'true dataset'* in the remainder of this article. The calibration consisted of several components.

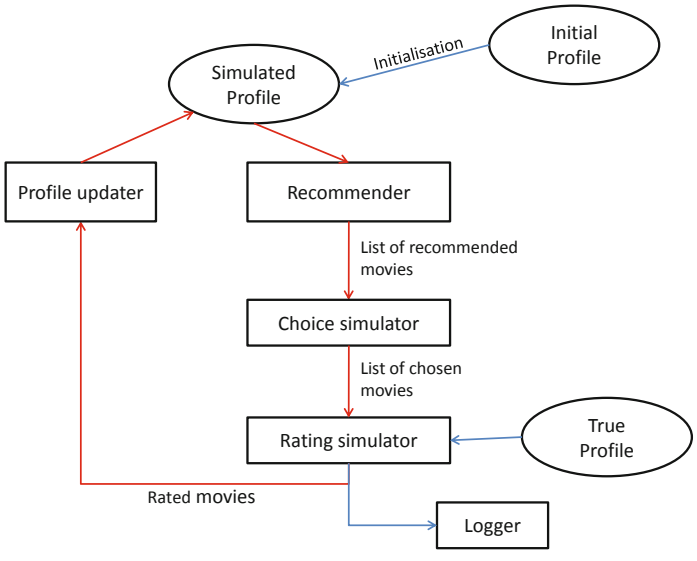

Fig. 2. Simulation outline

Firstly, in each round of the simulation, the system was allowed to recommend only those movies that appeared up to the end of that round in the true dataset. With this method, we aimed at controlling available movies, so the system would not recommend a movie that was only going to be released the following month.

Secondly, to avoid large variations caused by random initialisation, we initialised the recommender of the simulation using user-movie-rating triplets from the first one year of the Netflix data. This allowed us to make comparisons between our simulations and what happened in reality from year two onwards.

Thirdly, we only simulated movie recommendation and selection, but not movie rating. In other words, the scenario determined which movies a user got recommended and chose, but how users actually rated a movie was determined by their True Profiles (TP-s), that were built for them on the whole true dataset. The reason for this choice was, similar to the choices described above, to keep more control over the simulation.

The latter two design choices resulted in three kinds of profiles for a simulation: an Initial Profile (IP) was built using one year of data from the true dataset, a True Profile (TP) was built on the whole true dataset, and a Simulated Profile (SP) was updated in each round of the simulation. A Simulated Profile can be viewed as a profile that evolves from an Initial Profile over time, as an interaction between users and the recommender system.

The selection of users in a given round was also controlled. In each round, we selected the same users that rated movies in the true dataset in the corresponding period. For example, if there were ratings from 5000 users in the dataset for February 2002, then our simulation also used those 5000 users in the corresponding round. As we used the same ID-s in our simulation as those in the true dataset, we achieved a control over users' decisions to rate movies in a round (or, for that matter, to join and start rating movies at all).

Throughout the simulations, we assumed that no user should rate a movie more than once, which was also the case in the true dataset. Allowing for a movie to be rated several times would not have provided us with valid simulation data. It had been observed in test simulations that users would get the same recommendations in successive rounds if an item would be allowed to be re-recommended once already rated.

The number of movies that were recommended to a user in a round was determined by the scenario in use (see power-law and uniform versions of some scenarios). To allow for comparison with the real situation reflected by the true dataset, the overall number of movies rated in a simulation round was always the same as in the true dataset.

3.5 Measures of Diversity

For each of the simulations determined by the ten scenarios described in Section 3.3, we measured diversity in various ways. An overview of the used diversity measures is presented in Table 1.

Table 1. Overview of diversity measures used in our experiments

	Unique Movies	Global variance	User-based variance	Entropy	Cosine
Overall	x	x			
Per movie				x	
Per user			x		x
Source data: Binary	x			x	
Source data: 1-5		x	x		x

In addition to the diversity measures described below in more detail, we also calculated the *average rating over all users per simulation round*. It is considered to be positively related to user satisfaction and recommender system performance. We calculated the following diversity values in each round of a simulation:

- The **number of Unique Movies** rated in the given round. This measure allows us to see whether all the movies available in a simulation round were rated by at least one person. It shows diversity in the sense of breadth of rated movies. Note that, by definition, the number of unique movies in a simulation round can not be higher than that of the true dataset for the same period.

- **Global variance of ratings** in a simulation round, and that of the corresponding period in the true dataset. This measures diversity that describes the breadth of values of ratings given in a certain round. Both this and the previous measure are overall measures, i.e., they consider a round's data without using detailed information about individual users or movies.
- **Mean User-based variance of ratings**. We take the variance of ratings for each user of the current round, and then report the mean of these variances. Note that this measure is not concerned with the number of movies a user rates.
- **Normalised Shannon Entropy of rated movies** in a round. The entropy H is defined as follows:

$$H = -\sum_{i=1}^{n} p_i \cdot log(p_i); p_i = \frac{occ(movie_i)}{count(ratings)} \tag{4}$$

  where n is the number of unique movies rated in a round in question, and $occ(movie_i)$ denotes the number occurrences of the ith movie. As the maximum value of H depends on the number of movies n, we normalise H by dividing it by $log(n)$.
- **Cosine diversity**. The Cosine coefficient is a commonly used similarity measure, which we base another diversity measure upon. For each pair of selected users, the vectors U_i and U_j are considered, where $U_{i,m}$ denotes the rating given by user i to movie m, where m varies over all available movies. Then the Cosine coefficient is calculated for each possible pair of users. Because the maximal value of the Cosine coefficient – one – indicates perfect similarity, and we are rather interested in *diversity*, we consider a *Cosine diversity* measure which is defined as shown in Equation 5. For computational efficiency reasons, we calculate the coefficients on a randomly selected sample of $n = 1000$ users. Our tests show that this sample size is sufficiently representative of the whole population of users in a given round.

$$\hat{C} = 1 - \frac{1}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{U_i \cdot U_j}{\|U_i\| \cdot \|U_j\|} \tag{5}$$

Having described scenarios, the simulation setup, and diversity measures in this section, the next section presents and discusses results of the simulations that we carried out.

4 Results and Analysis

In this section, we present and discuss the results of our simulations. We do this using six figures: Figure 3 shows the mean ratings per round per scenario, and Figures 4 to 8 show various diversity values that we measured throughout the simulations.

There are a number of aspects from which results of the simulations can be investigated. After comparing our simulation results to the true dataset, we move on to discuss further three aspects as identified in the Introduction:

1. *What is the effect of forcing users to rate the same number of items in each round?*
2. *What is the effect of changing the type of information that a recommender gives to users?*
3. *What is the effect of mixing groups of users of different type?*

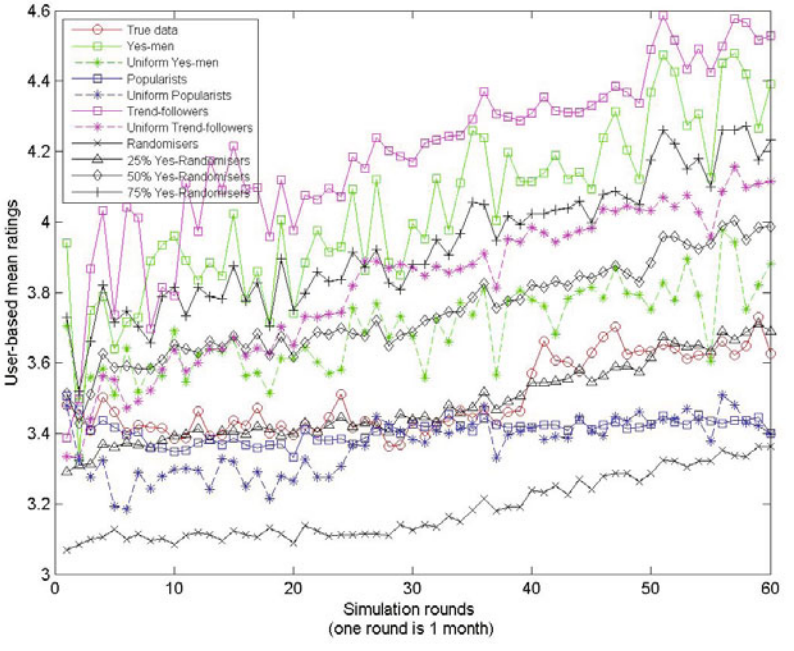

Fig. 3. Mean ratings over users

4.1 True Data and Simulation Results

As stated above, we first compare our simulation results to corresponding measurements on the true dataset.

Figure 3 shows that several of our scenarios resulted in mean ratings consistently higher than values in the true dataset, while some of our simulated values are lower. One of our scenarios with a mixed model (25% Yes-Randomisers) matches values of the true dataset quite closely. In other words, from the mean ratings perspective, when people accept personalised recommendations in 25% of the time, otherwise rate random movies, we get a similar situation to reality.

In terms of variance (Figures 5 and 6), we can observe that the true dataset's variances are consistently and considerably higher than those in our simulations. This observation can be explained as follows. In our simulations, everyone rates

movies according to the same "true rating model" that was built from the Netflix data. Therefore, user ratings are fully consistent with this model. However, it is well-known that humans are inconsistent when they are allowed to rate the same movie several times. Cosley *et al.* [6] measured the consistency of user ratings by asking 212 users to rate some movies and re-rate them again after 6 weeks, collecting in this way 1892 ratings on the 5-stars scale. It turned out that although the mean of new ratings was almost the same as for old ones (the difference in means was about 0.01 stars), the new ratings were quite different from the original ones: 20% of them were lower and 20% were higher. This means that the variance that could be attributed to a non-deterministic rating behaviour of humans is about 0.4. However, to avoid problems with extreme ratings (1 and 5 stars) the experiment considered only ratings that originally had values of 2, 3 or 4 starts. Therefore, we could speculate that when using the full scale of 5 stars, there would be additional 20% of the original '5 stars' ratings re-rated as '4 stars' and another 20% of '1 star' ratings that would be re-rated as '2 stars'. In the Netflix dataset 27.65% of ratings had value 1 or 5 what leads to a more realistic estimate of the variance that could be attributed to user inconsistency: $0.3447 = 0.2 * 0.2765 + 0.4 * 0.7235$. Here we assume that a lower or higher rating differs from the original rating by exactly one star. Let us notice that after correcting the true dataset's variances by 0.3447 (Figures 5 and 6), they will still be slightly higher than the variances measured on simulated data.

According to results displayed in Figure 7, the entropy of rated movies per round is higher than in the true data in several simulations, while lower in some others, suggesting that perhaps some combination of our scenarios would be able to model the true data, at least in the entropy sense.

Values of the Cosine diversity (Figure 8) reveal that the selection and ratings of movies in reality was close to a 1:3 mixture of Yes-men and Randomisers (= 25% Yes-Randomisers). Apart from the pure random scenario, other scenarios resulted in lower Cosine diversity than those already mentioned in this paragraph. In short, in reality, the differences in selections of movies between users are quite high.

Based on the findings above, we can conclude that the true dataset's diversity can be simulated as a mixture of Yes-men and Randomisers, but only in a limited sense. In other senses of diversity, other mixtures might model reality better. Looking at this from another perspective, if a recommender system owner wants to influence diversity of ratings, e.g., they want higher diversity in some sense, they might want to make different recommendations to different people (e.g., personalised or popularity-based recommendations), or they might also want to influence user behaviour (e.g., by trying to achieve that people act uniformly, so perhaps they might be easier to 'handle'). In the next subsection, we investigate the effect of forcing users to act uniformly, which is followed by the comparison of what a recommender can offer to users, and that, by the comparison of some mixture scenarios.

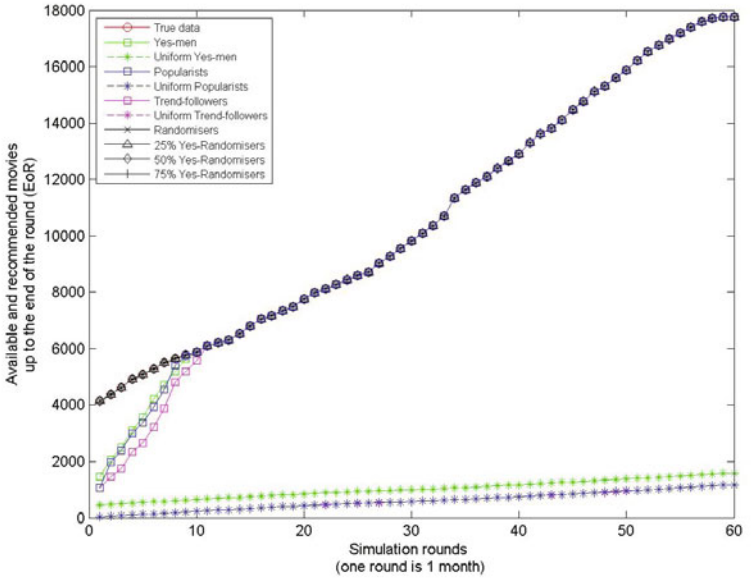

Fig. 4. Number of unique movies rated

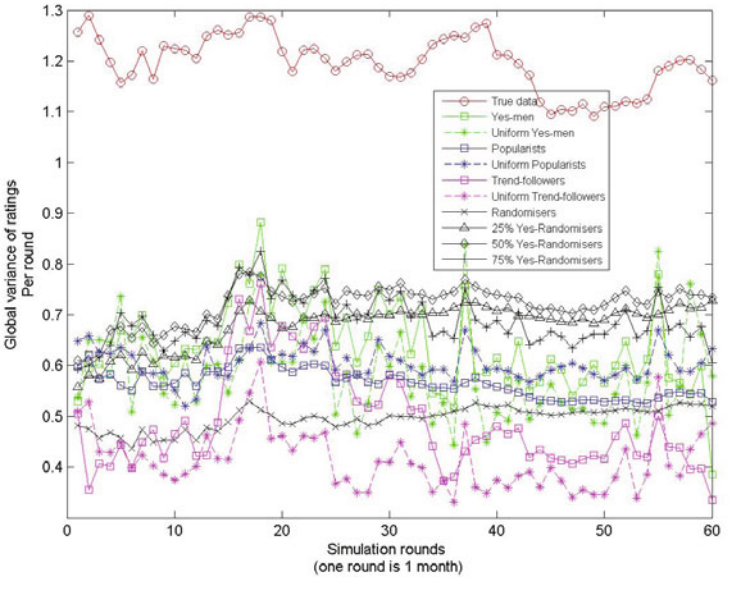

Fig. 5. Global variance of ratings

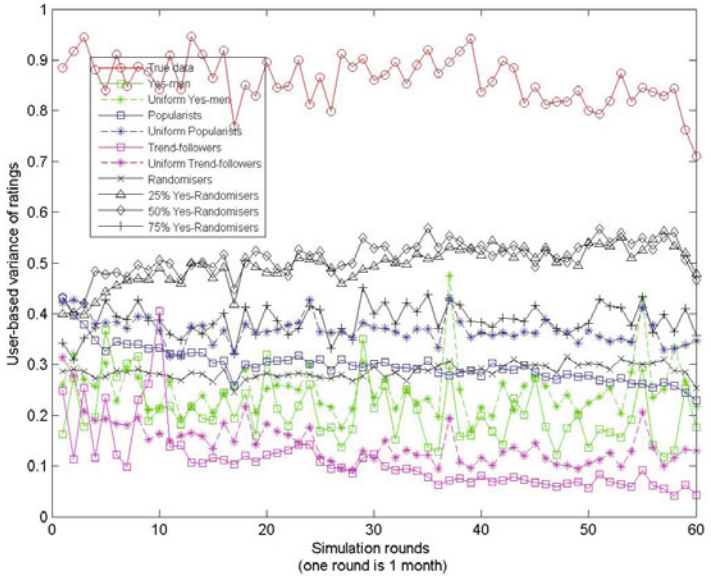

Fig. 6. User-based average variance of ratings

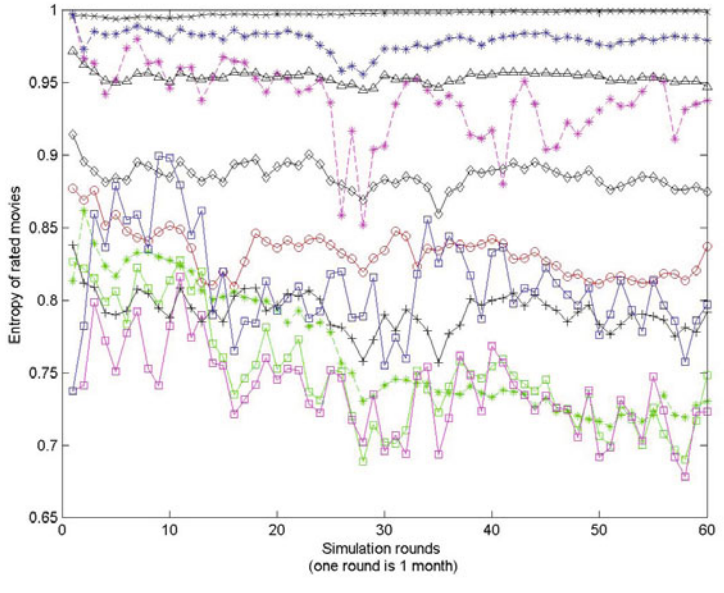

Fig. 7. Entropy of rated movies

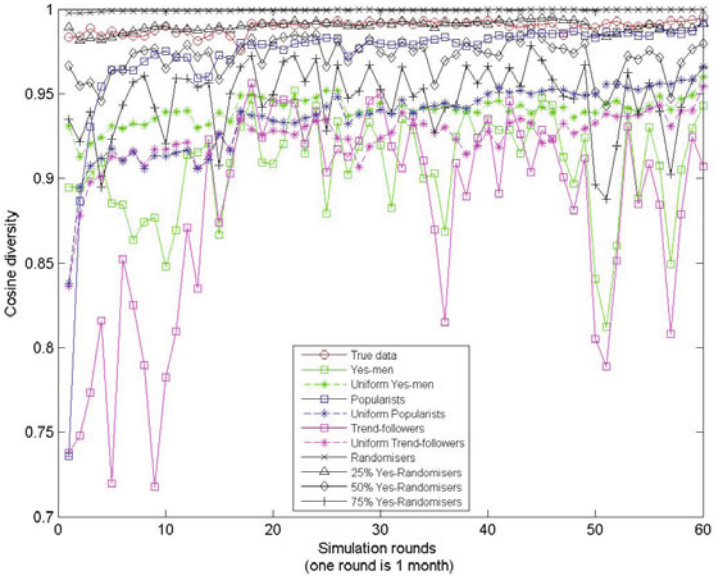

Fig. 8. Mean Cosine diversity values

4.2 'Normal' and 'Uniform' Scenarios

To study the effect of every active user rating the same number of movies in a round (i.e., a kind of forced behaviour), we consider six of our ten simulations. We analyse the 'transition' from Yes-men to Uniform Yes-men, from Popularists to Uniform Popularists, and from Trend-followers to Uniform Trend-followers, respectively. We attempt to find effects that are common in these three kinds of transitions.

It is an important observation that uniformity, in the sense we described it in the paragraph above, drastically decreases the number of movies ever watched. Figure 4 shows that, by the end of the simulation, in the unforced cases, almost 18.000 movies have been rated by at least one person, while in case of uniformly distributed numbers of ratings per movie, only around 1.000 movies have been rated.

When using the uniform versions of scenarios, the global variance of ratings tends to become lower in case of Yes-men and Trend-followers (Figure 5), however, user-based variance of ratings tends to increase in over three pairs of scenarios (Figure 6). Considering the two inconsistent pairs of scenarios, this is a kind of change also observed in [11], where it was found that the use of a recommender system decreases global diversity but increases individual diversity. As having to use a recommender system can be considered as a forced way of acting in a uniform way, some of our scenarios, which also simulate a forced way

of acting uniformly while using a recommender, are in line with their findings. Popularist scenarios do not confirm this, however.

The entropy of rated movies (Figure 7) is almost uniform, for uniform versions of choice models. This is particularly evident when we consider the differences between Trend-followers and Uniform Trend-followers as well as between Popularists and Uniform Popularists, and it seems to hold partially in case of the Yes-men scenarios. However, if one of these uniformities is not true, i.e., they can either get personalised recommendations or choose as many movies as they want, entropy stays lower, meaning that there will be movies increasingly more/less popular over time.

The Cosine diversity values in Figure 8 show that uniform versions of our models tend to show higher diversity in case of Yes-men and Trend-followers, and not in Popularists. The latter case can be explained by the fact that, almost by definition, the popular movies become even more popular, so Popularists rate a very limited set of movies (just the most popular ones) what automatically reduces the diversity of their ratings. In the former two cases, the number of movies that users have to rate is increasing with the number of required ratings. This leads to the increase of rating diversity.

On a different note, according to our simulations, forced uniformity makes the mean global ratings decrease (Figure 3), unless the mean ratings are very low already. This is probably caused by the fact that many users (in the long tail of the 'normal' power-law distribution) need to rate more movies than they actually like (or know).

Summary Forced uniformity has some negative effects on user behaviour, such as that the total number of selected items, and that the mean ratings tend to decrease. However, in several diversity measurements we do not get a very clear tendency of change. Looking at the results from another angle, if a recommender owner wants to achieve certain results (higher/lower diversity in some sense, higher mean rating which might be associated with user satisfaction), they need to know which recommendation types match which user behaviours. Then, they might either target groups of users with certain types of recommendations (e.g., highest rated movies or most popular ones), or try to change user behaviour some other way to match what they offer as recommendations.

4.3 Yes-Men, Popularists and Trend-Followers

In this subsection, we compare three groups of simulation results in order to find out how various – fundamentally different – movie recommendation approaches result in various diversity values. The first group contains the Yes-men type results (personalised recommendation), the second is a group of Popularists, and the third is results by Trend-followers.

In terms of number of unique movies rated (Figure 4), all three approaches produce the same results. In other words, it does not seem to matter which of the three approaches is used, the movie coverage will be very similar (though we are limited here to the movies rated in the real data).

The variance of ratings is lowest in case of Trend-followers (Figures 5 and 6), which shows that when highest rated movies are offered to users, the ratings will be very consistent. The global variance values of Yes-men tend to be higher than those of Popularists (Figure 5), while Popularists are clearly associated with higher user-based variance levels than Yes-men (Figure 6). So although the Popularist approach results in globally more consistent ratings, on a user level, users get more consistent quality of recommendations in case of a personalised approach. Quality is more consistent despite the greater fluctuation in variance for Yes-men (Figure 6), as variance values still tend to stay below those of Popularists.

With respect to the entropy of rated movies (Figure 7), Yes-men and Trend-followers have the same level of entropy[1], while Popularist movies are more evenly distributed in terms of how many users watch and rate individual movies. From these and other results presented in previous paragraphs it is becoming evident that recommending popular movies results in very different simulation behaviours from the other two approaches.

The difference of Popularists from Yes-men and Trend-followers is further backed up by Cosine diversity values in Figure 8. Selections and ratings of movies by the Popularist approach are considerably more diverse per user, while the other two approaches produce comparable Cosine diversity values. The difference is also shown by the Cosine values being more stable in case of Popularists.

If we take the mean ratings per user (Figure 3), we can see that the ratings of Popularists are the lowest among the three approaches studied in this subsection. Also, mean ratings corresponding to Popularists do not increase over time as with other approaches. The popularity-based approach is clearly different from the other two. As Figure 3 also shows, offering the so far highest rated movies (Trend-followers) results in the highest mean rating values, which is surprising given that we would have expected a personalised approach to produce the highest mean ratings (which we also associate with highest user satisfaction with the recommender). We speculate that either the level of personalisation is not of high enough quality then, or that recommending what has been rated highest so far is actually better than personalisation (of this kind).

Summary. Trend-followers are associated with the highest mean values, and often with the lowest diversity values (which indicate consistent quality of recommendations). Recommending highest rated movies, hence, makes sense, and it seems that the rating scale is just good enough to let new movies also be recommended (so they get a chance of getting high ratings). Also, a strictly popularity-based approach does not perform well according to our simulations, and does not produce consistent ratings: it is highly dependent on the initial set of movies rated, so when new items enter the database, they stand little chance of being recommended to many users. Hence, we think that a popularity-based approach should always be coupled with a promotion-based one (not

[1] Now we only consider the power-law versions of scenarios as i) they are the 'normal' cases (see true dataset), and ii) there are great entropy differences between these and uniform versions, which have already been addressed in the previous subsection.

investigated in this paper), so there can be a balance between old, popular movies, and new, potentially popular movies.

A different kind of mixture is investigated in the next section, which studies the gradual transition from a Yes-men approach to a totally random selection of movies.

4.4 From Yes-Men to Random Selection

In this section we investigate how diversities and main ratings change as we introduce noise into personalised recommendations. Whether the source of noise, i.e. randomness, comes from the user rejecting recommendations and rating other movies, or from the recommender introducing random movies (for example, to facilitate exploration of movies), it does not make a difference for our purpose. One of the questions we aim to answer is whether we can model the true dataset as a combination of Yes-men and noise. If this can be done, further research into recommenders and user behaviour could be made easier. For our investigation, we consider five scenarios in which randomness gradually ranges from 100% to 0%.

As Figure 3 shows, as we introduce more randomness into the personalised recommender approach, average ratings will gradually decrease. This is natural, as noise can not really be considered 'quality'. What is noticeable though, is that if users take only 25% of the personalised recommendations, we end up with mean ratings that closely match reality. So we might say that there is a lot to improve on the recommender producing the true dataset, but we also need to keep in mind that there are considerable differences between reality and our simulations (different recommender engine, various user behaviours in reality, etc.).

In terms of both types of rating variances measured (Figures 5 and 6), there do not seem to be a mixture that would model reality (even if reality is corrected by intra-user variability mentioned already). Also, there is no monotonic function that would show a clear transition from Yes-men to Randomisers. As the figures show, a 50% noise level (an even mixture) seems to give the highest variance in ratings. This is due to the fact that the mean ratings of Yes-men and Randomisers are relatively far apart, and low in variance, and, as there are both low and high ratings in mixture scenarios, the variance of mixtures should indeed be higher, reaching their peak at around a 50-50% mixture.

Regarding entropy (Figure 7), it seems possible to match reality relatively closely, although this is not achieved the same way as for the mean (i.e. with 25% Yes-Randomisers). The mixture that models reality would roughly be a 65% Yes-Randomisers scenario. Comparing entropy values of various levels of randomness, by definition, a totally random approach results in the highest entropy, while less noise results in movies less evenly distributed.

As for entropy, Cosine diversity values (Figure 8) also show the highest possible value for Randomisers, and that a mixture can model reality quite closely. However, now it is again the 25% Yes-Randomisers that fit the true data well (just as for the mean ratings). With respect to the overall values of Cosine diversity, total randomness results in users' selections and ratings being the most

different from one another, and less noise makes people select and rate movies more similarly.

Summary. Introducing noise to personalised movie recommendations can model reality, although not consistently across diversity measures. This shows that, naturally, reality is more complicated than a single recommendation algorithm plus a somewhat random user behaviour. However, for specific purposes, we believe that such mixtures can be useful for future studies of recommenders.

5 Conclusions

In this paper, we have investigated how diversity changes under different scenarios determined by 1) the type of information a recommender offers, and 2) the behaviour of users of the recommender. We have considered four main types of recommender-user interaction styles: Yes-men, Trend-followers, Popularists and Randomisers. Depending on the number of items that users rate we have identified two variants of each type (except Randomisers): 'power law', where users rate the same number of movies as in Netflix data, and 'uniform', where each user rates the same number of movies. Additionally, we have analysed mixtures of users of Yes-men and Randomiser type, using proportions of 25%, 50% and 75%.

For each of the 10 scenarios we have performed extensive simulations of 60 rounds (covering the period of 5 years) collecting, for each round, the following diversity measures:

- the mean and the variance of ratings,
- the mean of user-based variances of ratings,
- the number of unique movies rated in a given round,
- the normalised Shannon entropy of rated movies,
- Cosine diversity of users' ratings.

The analysis of results demonstrates a big variability of the simulated scenarios. In particular, it turned out that for non-uniform scenarios:

- the mean rating is highest for Trend-followers, then Yes-men, followed by Popularists and Randomisers,
- the diversity of rated movies (entropy) is biggest for Randomisers, then Popularists, followed by both Yes-men and Trend-followers,
- the diversity of user ratings (Cosine diversity) is biggest for Randomisers, then Popularists, followed by Yes-men and Trend-followers.

None of the simulated scenario mimics the behaviour of Netflix clients. This may have been caused by the fact, that these clients do not form a uniform group that falls under one of our scenarios. Finding mixtures of scenarios that reflect real-life user behaviour is beyond the scope of this paper, but is certainly worth further investigation.

Our results have some practical consequences for designers of recommender systems. We have found that imposing the uniform selection mechanism has some negative effects. For example, the average ratings of Yes-men and Trend-followers significantly drop, suggesting lower user satisfaction. Additionally, to achieve certain targets (e.g., high ratings which could be attributed to the high performance of the recommender system), user groups that behave according to certain patterns need to be identified. Once it is known how groups of users behave, they can be offered appropriate kinds of recommendations (e.g., personalised lists of items, most popular items, or certain mixtures) in order to achieve the previously identified targets.

References

1. 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 17(6), 734–749 (2005)
2. Ahn, H.J.: Evaluating customer aid functions of online stores with agent-based models of customer behavior and evolution strategy. Information Sciences 180, 1555–1570 (2010)
3. Alam, S., Dobbie, G., Riddle, P.: Exploiting swarm behaviour of simple agents for clustering web users' session data. In: Data Mining and Multiagent Integration, ch. 4. Springer, Heidelberg (2009)
4. Andrain, C.F.: Comparative political systems: policy performance and social change. M.E. Sharpe (1994)
5. Bennett, J., Lanning, S.: The netflix prize. In: KDD Cup and Workshop in Conjunction with KDD, pp. 3–6 (2007)
6. Cosley, D., Lam, S.K., Albert, I., Konstan, J.A., Riedl, J.: Is seeing believing? how recommender interfaces affect users' opinions. In: Cockton, G., Korhonen, P. (eds.) Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (2003)
7. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society of Information Science 41(6), 391–407 (1990), `citeseer.ist.psu.edu/deerwester90indexing.html`
8. Dehuri, S., Cho, S.B., Ghosh, A.: Wasp: A multi-agent system for multiple recommendations problem. In: Proc. of 4th International Conference on Next Generation Web Services Practices. IEEE (2008)
9. Derbyshire, J.D., Derbyshire, I.: Political Systems of the World. Palgrave Macmillan (1996)
10. Dias, M.B., Locher, D., Li, M., El-Deredy, W., Lisboa, P.J.G.: The value of personalised recommender systems to e-business: A case study. In: Pu, P., Bridge, D.G., Mobasher, B., Ricci, F. (eds.) Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 291–294 (2008)
11. Fleder, D., Hosanagar, K.: Blockbuster culture's next rise or fall: The impact of recommender systems on and sales diversity. Management Science 55(5), 697–712 (2009)
12. Forsyth, D.: Group Dynamics, 3rd edn. Wadsworth Publishing (1998)
13. Funk, S.: Netflix Update: Try This at Home (2006), `http://sifter.org/~simon/journal/20061211.html`

14. Garfinkel, R., Gopal, R., Pathak, B., Venkatesan, R., Yin, F.: Empirical analysis of the business value of recommender systems. Tech. Rep. Working Paper 958770, Social Science Research Network (2006)
15. Gursel, A., Sen, S.: Improving search in social networks by agent based mining. In: Kitano, H. (ed.) Proceedings of the International Joint Conference on Artificial Intelligence (2009)
16. Gursel, A., Sen, S.: Producing timely recommendations from social networks through targeted search. In: Decker, K., Sichman, J., Sierra, C., Castelfranchi, C. (eds.) Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems, AAMAS 2009 (2009)
17. Ishigaki, T., Motomura, Y.: Toward computational modeling of the consumer based on a large-scale dataset observed in a real service. In: Proc. of International Conference of Soft Computing and Pattern Recognition. IEEE Computer Society (2009)
18. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42, 30–37 (2009)
19. Lee, W.P., Yang, T.H.: Personalizing information appliances: a multi-agent framework for tv programme recommendations. Expert Systems with Applications 25, 331–341 (2003)
20. Ono, C., Kurokawa, M., Motomura, Y., Asoh, H.: A context-Aware Movie Preference Model using a Bayesian Network for Recommendation and Promotion. In: Conati, C., McCoy, K., Paliouras, G. (eds.) UM 2007. LNCS (LNAI), vol. 4511, pp. 247–257. Springer, Heidelberg (2007)
21. Said, L.B., Drogoul, A., Bouron, T.: Multi-agent based simulation of consumer behaviour: Towards a new marketing approach. In: International Congress on Modelling and Simulation Proceedings (2001)
22. Said, L.B., Bouron, T.: Multi-agent simulation of virtual consumer populations in a competitive market. In: Proceedings of the Seventh Scandinavian Conference on Artificial Intelligence. IOS Press (2001)
23. Said, L.B., Bouron, T., Drogoul, A.: Agent-based interaction analysis of consumer behavior. In: Proceedings of First International Joint Conference on Autonomous Agents and Multi-Agent Systems. ACM (2002)
24. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, WWW 2001, pp. 285–295. ACM, New York (2001), http://doi.acm.org/10.1145/371920.372071
25. Sen, S.: Finding Useful Items and Links in Social and Agent Networks. In: Cao, L., Bazzan, A.L.C., Gorodetsky, V., Mitkas, P.A., Weiss, G., Yu, P.S. (eds.) ADMI 2010. LNCS, vol. 5980, pp. 3–3. Springer, Heidelberg (2010)
26. Sen, S., Saha, S., Airiau, S., Candale, T., Banerjee, D., Chakraborty, D., Mukherjee, P., Gursel, A.: Robust Agent Communities. In: Gorodetsky, V., Zhang, C., Skormin, V.A., Cao, L. (eds.) AIS-ADM 2007. LNCS (LNAI), vol. 4476, pp. 28–45. Springer, Heidelberg (2007)
27. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in Artificial Intelligence, 1–19 (2009), http://dx.doi.org/10.1155/2009/421425
28. Takács, G., Pilászy, I., Németh, B., Tikk, D.: Scalable Collaborative Filtering Approaches for Large Recommender Systems. J. Mach. Learn. Res. 10, 623–656 (2009)
29. Weng, L.T., Xu, Y., Li, Y., Nayak, R.: Towards information enrichment through recommendation sharing. In: Cao, L. (ed.) Data Mining and Multiagent Integration. Springer, Heidelberg (2009)