Diversity Measurement of Recommender Systems Under Different User Choice Models

Z. Szlávik and **W. Kowalczyk** and **M.C. Schut**
Department of Computer Science
VU University Amsterdam (NL)
{z.szlavik, w.j.kowalczyk, m.c.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 paper, 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 the agent's choice model), run very extensive simulations, analyse various measurements regarding experimental validation and diversity, and report on selected findings. The choice models are an essential part of our work, since these can be influenced by the owner of the recommender once deployed.

Introduction

There is no denying that recommender engines have made an overwhelming entrance into the world of digital information. For this, 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 start to realise that recommenders may benefit the users in finding items of interest. Whether recommendations are generated by algorithms (Amazon, Netflix) or in 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 to know how users react to their engine, and, following from that, how they or the engine can react to these user's 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 paper, we look at the interaction effects between recommendation and user behaviour. We test a movie recommender in combination with a variety of choice models (i.e., the way that users react to predictions 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 paper, 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., (Garfinkel et al. 2006; Dias et al. 2008; Fleder and Hosanagar 2009), 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.

The objective of the study reported on in this paper 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 two 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.

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 (Andrain 1994; Derbyshire and Derbyshire 1996; Forsyth 1998). 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). We come back to this generalisation of our work later in this paper.

Related Work

The history of recommender systems is relatively short: the first recommender system, Tapestry, was developed about 20 years ago, (Goldberg et al. 1992). Since then this field has been developing very quickly, strongly stimulated by rapid development of the Internet and e-commerce (Adomavicius and Tuzhilin 2005). The main role of a recommender system is to provide its users with recommendations concerning possible products (such as 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, et cetera.

There are two main strategies for building recommender systems. The first strategy, called *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 similar way and offer them similar products.

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. 2001).

In 2006, a DVD rental company, Netflix, announced a big competition (with a $1 million prize) that was aimed at improving the accuracy of existing recommender systems (Bennett and Lanning 2007). This competition attracted the attention of thousands of researchers from all over the world, leading to the discovery of several new approaches. They, eventually, led to the improvement of the accuracy of the original Netflix recommendation engine, Cinematch, by more than 10%. These 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 retrival (Deerwester et al. 1990), but it was Simon Funk (real name Brandyn Webb) who described, on his blog (Funk 2006), 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, what resulted in numerous improvements of the original method, and finally, in reaching the main objective of the Netflix Challenge (improvement by 10%). Several improvements of Funk's method are described in (Koren, Bell, and Volinsky 2009; Takács et al. 2009).

In addition to papers on algorithmic aspects of recommender systems, there are several papers that address some psychological or economical issues of using recommender systems. In particular, the paper (Fleder and Hosanagar 2009) focuses on analysing the impact of recommender systems on sales diversity. The authors provide an analytical model of recommenders and, additionally, run some simulations with actual recommender systems on synthetically generated data.

The field of recommender systems is very broad. For further references the reader should consult survey articles, such as (Adomavicius and Tuzhilin 2005; Su and Khoshgoftaar 2009). Additionally, (Herlocker et al. 2004) provides an extensive overview of various methods for evaluating collaborative systems.

Netflix Data and Funk's Algorithm

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 (Bennett and Lanning 2007) 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) removed. The quality of recommender algorithms was measured with help of the Root-Mean-Squared-Error:

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

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.

In our experiments we decided to use the algorithm of Simon Funk, (Funk 2006). The algorithm is very simple, computationally cheap, and predicts ratings with very high accuracy. 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, called *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 the dot product of both vectors:

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

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:

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

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 and its minimum can be found with the help of the gradient descent algorithm. In our experiments we set the training parameters to values that were suggested in (Funk 2006): 10 features, all initialised to 0.1, 200 iterations per feature, learning rate set to 0.001, and the regularisation term set to 0.015.

We will refer to the collection of user and movie features as the *rating model*: it can be used to predict, for any user and any movie that occured in the training set, the rating that the user could give to the movie.

Experimental Setup

In order to study how a recommender system affects diversity, it is important to understand the circular nature of the 'life' of a recommender. On one hand, the recommendations a system provides to its users shape the way the 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'. First, the Recommender uses the Current Rating Model (CRM) to produce recommendations for a specific user. Then the User selects some of the recommended movies and rates them. The ratings are sent back to the Recommender and the CRM is updated. In this way, a user's behaviour affects the rating model and vice versa: the rating model has an impact on a user's ratings. To investigate how diversity changes over time, we run simulations in which we control user behaviour. We control user behaviour by introducing several choice models that simulate how users make selections from a list of recommended items. We run a number of simulated rounds one after another. The simulation output for each choice model is then analysed and compared, focusing on various senses of diversity.

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

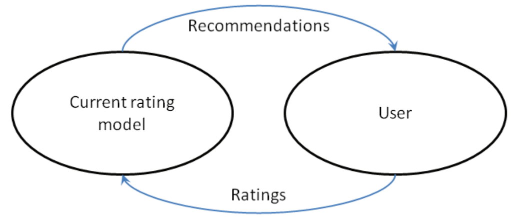

Figure 1: Movie- and user profiles, and movie selection and ratings affect each other.

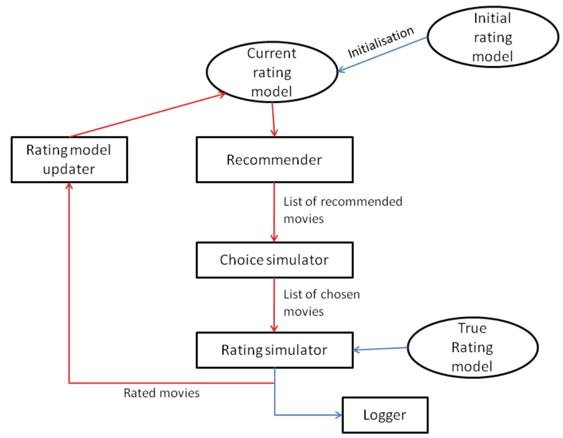

Figure 2: Simulation outline.

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 (i.e., CRM) 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 rating model and moving on to the next period. The Netflix dataset covers the period of 6 years; we use the data from the first year to initialize the system, and use the remaining data for simulating 60 rounds for each choice model. The choice of the one month interval was made as it provided us with a sufficient number of rounds, suitably smooth diversity measurements, and a feasible time frame for running the simulations on a single computer.

To model the actual rating behaviour of users we used the complete Netflix dataset, also refered to as the 'true dataset', to develop the True Rating Model (TRM). We assumed here that the model that is trained on the complete dataset will correctly approximate the ratings given by users to movies.

The Netflix data was also used for keeping control over the simulations and making them realistic. 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 year covered by 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 selection, with the use of choice models, but not movie rating. In other words, the choice model active for the current simulation determined which movies a user chose from their recommendation list, but how users actually rated a movie was determined by the True Rating Model. 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 rating models for a simulation: an Initial Rating Model (IRM) was built using the first year of data, a True Rating Model that was built on the whole true dataset, and a Current Rating Model was updated in each round of the simulation. A CRM can be viewed as model that evolves from the IRM over time, as an interaction between users and the recommender system. We could notice here that as the simulations can go differently than in real life, the last CRM might be different than the TRM.

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 2000 users in the dataset for January 2002, then our simulation also used those 2000 users in the corresponding round. As we used the same IDs 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 rates a movie twice, which was also the case in the true dataset. Allowing for a movie to be rated several times would not provide us with valid simulation data, as it was observed in test simulations that user profiles do not change so significantly as to not recommend users the same movies in several, often all, successive rounds.

The number of movies that were recommended to a user in a round is determined by the choice model in use. Depending on the choice model, users selected either the same number of movies as they indeed did according to the true data, or the number of movies per user was distributed evenly among users of the round. 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.

Choice models

We used six choice models, which resulted in six runs of simulation. Choice models were not mixed, i.e. each user behaved according to the choice model of the simulation being executed. This way we could observe how the use of particular choice models affected diversity. The used choice models were the following:

1. The first choice model assumes that each user of the recommender system completely accepts the recommendations. Users rate those, and only those, movies that are recommended to them. The number of movies recommended to users is the same as in the true dataset. Throughout this paper, we refer to this choice model as **Yes-men**.
2. The second choice model, similarly to the first one, assumes that users rate those and only those movies that are recommended to them. However, in this case, the number of ratings per user follows a uniform distribution, i.e. every user rates the same number of movies. These users are called **Uniform Yes-men**.
3. The third choice model assumes that everyone watches and rates movies that are highest rated on average. In other words, everyone rates the globally top movies. The number of movies rated per user is the same as in the true dataset (i.e. not uniformly distributed). This group of users is referred to as **Trend-followers**.
4. Choice model four is the same as model three, only this time the number of movies rated is distributed evenly among users. These users are the **Uniform Trend-followers**.
5. The fifth choice model assumes that users are completely indifferent to recommendations. These users were simulated as if they rated a random selection of movies from the movies available for the round. The number of movies selected per user is the same as in the true dataset. We call these users **Randomisers**.
6. The last, sixth, choice model is a mixture of choice models one and two: Users accept the movies recommended to them, but only with a certain probability. For each movie in the recommendation, if the user does not rate the movie as recommended, they rate a random movie that is not in the list of recommended movies. The number of movies per user conforms to that of the real dataset. After some test simulations, we set the probability that a recommended movie is actually rated by a user to 0.25. With this value, we achieved mean ratings fairly comparable to those in the true dataset. This choice model is called **25% Yes-Randomisers**.

Measures of diversity

For each of the six simulations determined by the six choice models introduced above, we measured diversity in various ways. An overview of the used diversity measures is presented in Table 1.

In addition to the diversity measures described below, we also calculated the *average ratings per simulation round*. It is considered to be positively related to user satisfaction and recommender system performance. The following list describes the diversity measures we used.

- 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. Entropy is calculated as follows:

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

  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 with $log(n)$.
- **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, $m \in 1 : count(available\_movies)$. 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 define Cosine diversity as one minus the Cosine coefficient.

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

Results and Discussion

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 choice model, and Figures 4 to 8 show various diversity values that we measured throughout the simulations.

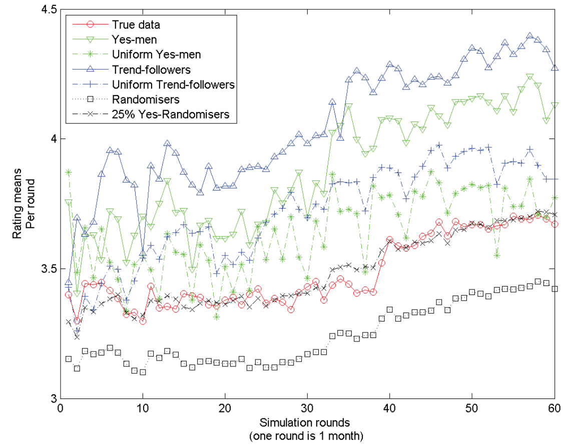

Figure 3: Mean ratings.

There are a number of aspects from which results of the simulations can be investigated, of which we consider the following three. Firstly, we investigate diversity values (as well as mean rating values) of our six simulations in comparison to corresponding values extracted from the true dataset, thus also providing some validation for our simulation results. Secondly, we investigate the effects of forcing users to behave in a certain uniform way, i.e. when every user rates the same number of movies per round. Thirdly, we compare three groups of simulations to each other: 1) where users rate movies the recommender system suggests, 2) when everyone rates movies that are highest rated overall, and 3) when movies are randomly selected for rating.

True data vs. Simulation results

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

As Figure 3 shows, our deterministic models resulted in mean ratings consistently higher than values in the true dataset. We can also infer from Figure 3 that the introduction of randomness decreases average ratings. It needs to be noted that the parameter value of 0.25 for the 25% Yes-Randomisers has been determined (after some small-scale simulations) so that simulation mean ratings do not deviate considerably from true mean ratings.

In terms of variance (Figures 5 and 6), we can observe that the true dataset's variances are considerably higher than those in our simulations. There might be several reasons for this, but we think that two important reasons stand out. First, our system is pure in the sense that, in a particular simulation, everyone selects movies in a determined way, but also, intra-user variability is not considered (i.e., users would always give the same rating to a movie if they were allowed to rate it multiple times). Second, the SVD algorithm has been proven to be better than Netflix's own algorithm used

	Unique Movies	Global variance	User-based variance	Entropy	Cosine
Overall measurements	x	x			
Per movie measurements				x	
Per user measurements			x		x
Source: Binary lists/vectors	x			x	
Source: 1-5 values lists/vectors		x	x		x

Table 1: Overview of diversity measures we used.

at the time of recording the data. Further to variance values, this quality difference between recommender algorithm could also explain why the mean ratings of our models are higher than those in the true dataset.

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

Values of the mean Cosine coefficient (Figure 8) reveal that the selection and ratings of movies in reality was close to a 1:3 mixture of Yes-men and Randomisers. As also observed above already, values calculated from the true dataset tend to be between values generated by deterministic and random choice models, respectively.

Based on the findings above, we can conclude that the true dataset's diversity could be simulated as a mixture of Yes-men and Randomisers, but only in some sense. This means that in case a recommender owner wants to control diversity, they might try to influence user behaviour such that a desired high/low level of diversity is reached. One possible way of control is requiring users to rate a certain number of movies per month (not many more, and not many less). The effect of such a manipulation is investigated in the next subsection.

'Normal' vs. Uniform simulation results

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 four of our six simulations. We analyse the 'transition' from Yes-men to Uniform Yes-men, and from Trend-followers to Uniform Trend-followers, respectively. We attempt to find effects that are common in these two 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 case, 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 choice models, the global variance of ratings tends to become lower (Figure 5), however, user-based variance of ratings tends to increase (Figure 6). This is a kind of change also observed in (Fleder and Hosanagar 2009), 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, our results, in which we simulate a forced way of acting uniformly while using a recommender, are in line with their findings.

The entropy of rated movies (Figure 7) indicates that the distribution of movies when using the uniform versions of choice models is closer to being uniform themselves. This is particularly evident when we consider the differences between Trend-followers and Uniform Trend-followers, and not so much in case of the Yes-men models. When users can only choose from the same list of movies and they need to choose the same number of movies, the distribution of movies are even (i.e., entropy is high). 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 mean Cosine coefficients in Figure 8 show that uniform versions of our models tend to have higher Cosine values. This means that, although users are forced to act uniformly in a way, the average difference between movie selections of two users does not reflect uniformity. This effect is likely to be the result of the presence of a particular type of users in 'unforced' circumstances, who rate a lot more movies than the average population. Rating a lot of movies allows these users to have a greater chance of non-zero dot product with others' rating vectors, which contributes to an overall lower mean Cosine coefficient value. As the distribution of how many movies a user rates follow a power law, the dominant users of the distribution cause the mean Cosine coefficient to be lower than in the uniform case.

Also, according to our simulations, forced uniformity makes the mean global ratings decrease (Figure 3). This is probably caused by the fact that many users (in the long tail) need to rate more movies than they actually like (or know).

In conclusion, forced uniformity has several negative effects on user behaviour. First, the total number of selected items, as well as the mean rating, are decreasing, and the average "distance" between users is increasing. We believe that the only reason to apply such uniform models in practice is the need for collecting more data that is needed for training recommender systems.

Yes-men vs. Trend-followers vs. Randomness

In this subsection, we compare three groups of simulation results in order to find out how various – fundamentally different – movie selection methods result in different diversity values. The first group contains the Yes-men type results, the second is a group of Trend-followers, and the third is results by choice models involving random selection of movies.

As it was expected, a random selection of movies results in the lowest mean ratings, and we believe this also means lowest satisfaction with the selection of movies users got to rate. The highest mean ratings are associated with Trend-follower type choice models[1].

In terms of number of unique movies rated (Figure 4), a random selection of movies covers the whole spectrum of available movies, while Yes-men and Trend-followers are either not much different from random selection ('normal' versions) or the spectrum is considerably tighter (Uniform versions).

The global variance of ratings (Figure 5) is high and stable in case of choice models involving randomness, while Yes-men have higher variance then Trend-followers. User-based variance (Figure 6) also shows a similar pattern – random models result in highest variance, Yes-men are in the middle, and Trend-followers are lowest.

With respect to entropy (Figure 7), total randomness clearly is on top, as, by definition, entropy is highest when samples are taken from a uniform distribution. As there is a large entropy difference between various versions of Yes-men and Trend-followers (discussed in the previous subsection), we do not feel that there is enough evidence to draw conclusions about entropy results with respect to these two types of choice models.

The mean Cosine diversities (Figure 8) also show the highest values for randomness dominated choice models, indicating that the pairwise differences between users is highest when movies are selected randomly. This is naturally the case as the large number of available movies and the low number of ratings per user per round make it very unlikely that rating vectors of two users will overlap. Yes-men have the second lowest mean Cosine values and Trend-followers are associated with the lowest of these values, which reflects that – due to the higher chance of rating the same movies[2] – the overlap between rating vectors of users will be higher.

These results show what designers or recommender systems might expect from a user: if they offer a personalised list of movies, then diversity is expected to be higher but overall ratings will suffer; if they only offer popular movies, then ratings will be higher, but diversity will drop; and finally, if they offer random movies (we can call this an 'exploration-facilitating' recommender), ratings are likely to become low, but a wide range of movies will be offered at least to some people.

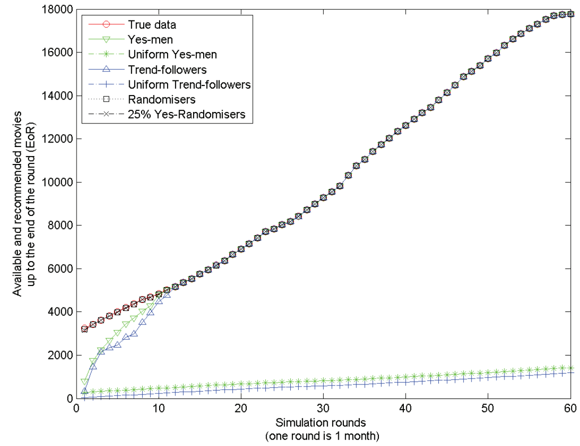

Figure 4: Number of unique movies rated.

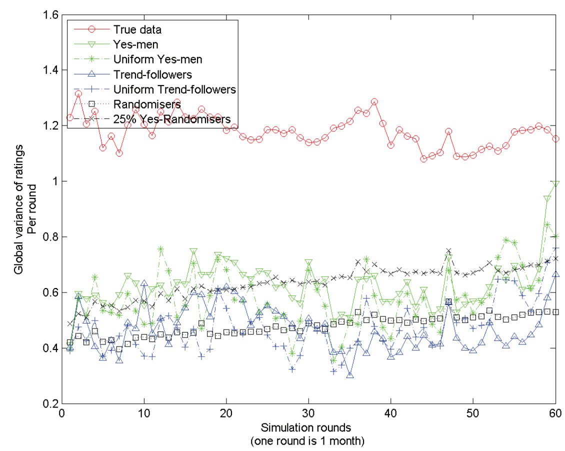

Figure 5: Global variance of ratings.

Conclusions

In this paper, we have investigated how diversity changes under different models of user behaviour when using a recommender system. Our simulations have practical implications, both for recommender system owners and designers. In particular, we have found that forced uniformity in terms of number of items rated does not necessarily result in users becoming more uniform, and the mean ratings they give to items will decrease, indicating lower system performance as perceived by the user (which answers Question 1 from the Introduction). Also, we have identified how three kinds of choice models, i.e., Yes-men, Trend-followers and Randomisers, result in different diversity and mean rating values (answering Question 2). This is a particularly important result as these behaviours can directly be encouraged by recommender system owners, e.g., we might decide to offer more trending items to one particular kind of users.

A limitation of our work is that, apart from the 25% Yes-Randomisers choice model that implicitly assumes a somewhat heterogeneous population of users, no mixtures of behaviours have been investigated. Further simulations need to be carried out to see how more heterogeneous populations

[1]Note that the comparison between Yes-men and Trend-followers is carried out in a coupled manner, e.g., we say that the mean for Trend-followers is higher because it is higher for Trend-followers than for Yes-men, but also higher for Uniform Trend-followers than for Uniform Yes-men.

[2]Note that the overlap is not very high in case of Trend-followers as users are not allowed to rate the same movie multiple times, and many users join the system, and rate again, in different rounds.

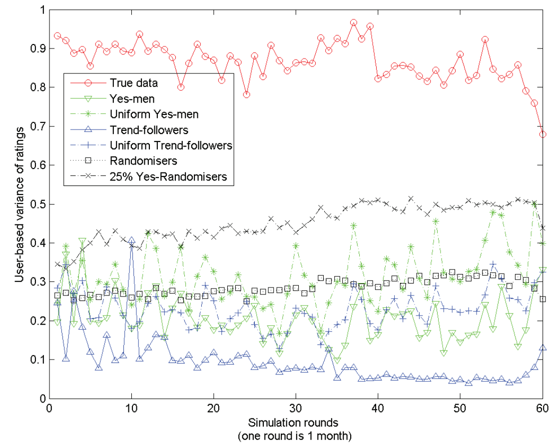

Figure 6: User-based average variance of ratings.

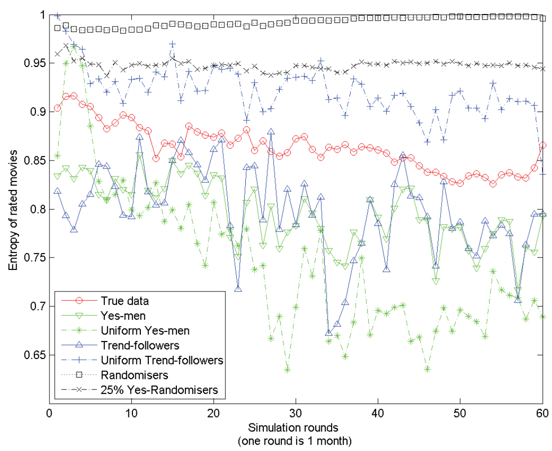

Figure 7: Entropy of rated movies.

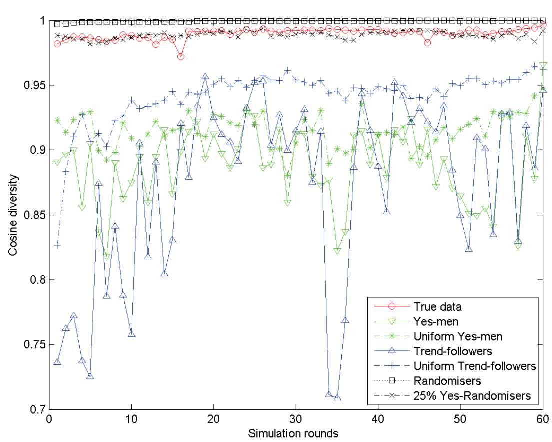

Figure 8: Cosine diversity values.

(and their interactions) affect diversity levels.

Our work is related to the areas of group dynamics and political systems, as mentioned in the introduction. The choice models 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), which we intend to investigate further in future work.

References

Adomavicius, G., and Tuzhilin, A. 2005. 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.

Andrain, C. F. 1994. *Comparative political systems: policy performance and social change*. M.E. Sharpe.

Bennett, J., and Lanning, S. 2007. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 3–6.

Deerwester, S. C.; Dumais, S. T.; Landauer, T. K.; Furnas, G. W.; and Harshman, R. A. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science* 41(6):391–407.

Derbyshire, J. D., and Derbyshire, I. 1996. *Political Systems of the World*. Palgrave Macmillan.

Dias, M. B.; Locher, D.; Li, M.; El-Deredy, W.; and Lisboa, P. J. G. 2008. The value of personalised recommender systems to e-business: A case study. In Pu, P.; Bridge, D. G.; Mobasher, B.; and Ricci, F., eds., *Proceedings of the 2008 ACM Conference on Recommender Systems*, 291–294.

Fleder, D., and Hosanagar, K. 2009. Blockbuster culture's next rise or fall: The impact of recommender systems on and sales diversity. *Management Science* 55(5):697–712.

Forsyth, D. 1998. *Group Dynamics*. Wadsworth Publishing, 3 edition.

Funk, S. 2006. Netflix Update: Try This at Home. `http://sifter.org/~simon/journal/20061211.html`.

Garfinkel, R.; Gopal, R.; Pathak, B.; Venkatesan, R.; and Yin, F. 2006. Empirical analysis of the business value of recommender systems. Technical Report Working Paper 958770, Social Science Research Network.

Goldberg, D.; Nichols, D.; Oki, B. M.; and Terry, D. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35:61–70.

Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; and Riedl, J. T. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22:5–53.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42:30–37.

Sarwar, B.; Karypis, G.; Konstan, J.; and Reidl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, 285–295. New York, NY, USA: ACM.

Su, X., and Khoshgoftaar, T. M. 2009. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence* 2009:1–19.

Takács, G.; Pilászy, I.; Németh, B.; and Tikk, D. 2009. Scalable Collaborative Filtering Approaches for Large Recommender Systems. *J. Mach. Learn. Res.* 10:623–656.