A Fast Learning Recommender Estimating Preferred Ranges of Features

Takuya Watanabe[1], Yuji Nakazato[1], Hiroaki Muroi[1], Takuya Hashimoto[2], Toru Shimogaki[3], Takeshi Nakano[4], and Tsutomu Kumazawa[5]

[1] Edirium K.K.
1-8-21 Ginza, Chuo, Tokyo, Japan
{sodium, nakazato, muroi}@edirium.co.jp
[2] Acroquest Technology Co., Ltd.
3-17-2 Shinyokohama, Kohoku, Yokohama, Kanagawa, Japan
hashimoto@acroquest.co.jp
[3] NTT DATA Corporation
3-3-9 Toyosu, Koto, Tokyo, Japan
shimogakit@nttdata.co.jp
[4] Recruit Technologies Co., Ltd.
1-9-2 Marunouchi, Chiyoda, Tokyo, Japan
tf0054@r.recruit.co.jp
[5] Software Research Associates, Inc.
2-32-8 Minami-Ikebukuro, Toshima, Tokyo, Japan
tkumazawa@acm.org

Abstract. We propose a recommender system which is based on a semisupervised classification algorithm designed for estimating users' preferred ranges of features. The system is targeted for new users, and it infers likings incrementally by presenting two alternatives to users in each step. To create the learning model, multidimensional scaling is employed to reduce the original feature space to a low-dimensional space. Then, the proposed classification algorithm, called geometrical exclusion, effectively finds a region which is not preferable for users and is to be excluded from the model in the reduced space. The algorithm consists of simple geometrical operations that are based on preference information obtained from users. An experiment by simulation is conducted to measure the performance of the system, and the result indicates that it can produce good recommendations with a practical number of user interaction steps. We also report statistics collected from our system deployed in a commercial web service.

Keywords: Recommender System, Interactive Preference Learning, Multidimensional Scaling, Geometrical Operations

1 Introduction

Recommender systems have been intensively studied and attracted industry's interest, and now many web sites are equipped with recommender systems. They

are usually classified into two categories: *content-based recommenders*, whose estimation process depends on similarity between items, and *collaborative recommenders*, which harness information gathered from people with similar preferences [1, 17]. Collaborative recommendation has been demonstrated to be a powerful tool to estimate users' preferences, but to infer new users' preferences is intrinsically difficult. This is called *the cold start problem* [15].

In this paper, we propose a content-based recommender system for estimating new users' preferences to tackle the cold start problem. Preferences can be estimated by feature-wise interaction with the user as in critiquing systems [18], but it requires a high degree of user effort [13]. Considering the widespread use of mobile smart devices, such as smartphones and tablets, a complex user interface and/or an interface that needs substantial number of operations would be unacceptable. To invent a new sophisticated user interface could solve the problem [3], but to master a new interface would be a burden to casual users. To promote users' instant understanding, we opt to simply present two items to a user each turn and ask her to choose preferred one. Additionally, the proposed system can produce recommendations from item sets of various sizes.

The recommendation problem is commonly formulated as estimating *ratings* of items that have not been seen by a user in the literature[1]. We formulate it as an online semisupervised classification problem instead. The problem is intuitively decomposed as follows: 1) creating an initial model from the whole dataset whose objects are all unlabelled,2) presenting a set of objects selected from the dataset to a user, where cardinality of the set could possibly be one but is two in our case, 3) receiving response from the user with labellings of objects, i.e., either *preferred* or *not preferred*, 4) updating the model based on the labellings, and 5) predicting labels of unlabelled objects by referencing the model. Precision of the prediction would increase through iterating the steps 2-4.

Our system uses a model embedded in a two-dimensional space as the initial model and employs multidimensional scaling (MDS) [5] to create it. MDS is a technique to construct a low-dimensional representation while preserving the distances between objects as well as possible. It has been mainly used as a visualisation technique to convert high-dimensional dataset to two- or three-dimensional image, as in [2]. Some recommender systems employ MDS or related visualisation techniques to create a two-dimensional map to be presented to a user, and user navigation is done in that map directly [6, 10, 9]. Giamattei and Sholtz employed correspondence analysis to visualise product recommendations and catalogues in a two-dimensional space [7]. In [16], MDS is used to embed both labelled and unlabelled objects in a reduced space, and linear discriminant analysis (LDA) is taken in the reduced space to classify objects. Khoshneshin and Street proposed a collaborative filtering algorithm that simultaneously embeds users and items in a reduced space by using MDS [11]. Le and Lauw proposed an embedding algorithm of both users and items based on a Bayesian approach, given preference data of items evaluated by users [12]. In [14], the problem of localising new users and items into an existing embedding is formulated as a quadratic programming. Grad-Gyenge et al. proposed a recommendation method

based on a graph embedding [8]. To the best of our knowledge, there has been no attempt to propose an online classification algorithm which directly updates a low-dimensional model.

We present the proposed algorithm, which we call *geometrical exclusion*, used in our system in Section 2. Our algorithm has several number of unique features, such as fast learning inspired by binary search and regularisation of the learning process by referring to the original feature space to mitigate the degraded precision caused by restricting to learn in a two-dimensional space. In Section 3, we report the result of a simulation based experiment to measure the performance of our system with varying sizes of item sets. The experimental result shows that our algorithm is able to recommend favourable items with relatively steady and practical number of steps. Further, we deployed our system as a commercial web service as reported in Section 4. The result also indicates that our algorithm performs well in a production setting. We conclude our discussion in Section 5 with some notes on future work.

2 Geometrical Exclusion

In this section, we explain how to select objects to be labelled by a user and how the model is updated geometrically by using the information obtained from labelled objects and by excluding a region of the model.

2.1 Basic Idea

In our setting, the system presents a pair of objects to be labelled to a user. Assuming that there is no prior knowledge about user's liking, an efficient way is to select objects in a space along the line of binary search. We illustrate this by an example. Suppose we have a feature f_1 that ranges from 0 to 100. We want to know user's liking as a subrange, so we divide the range at the midpoint and present representative objects, which we call candidates, of each divided subrange to the user. Here we take candidates c_1 and c_2 from around 20 and 80 respectively since they are clearly distinguishable and not too extreme. If the user chooses c_1, we retain $[0, 50]$ and discard the rest. In the next turn, we divide the remaining range, present candidates around 10 and 40 to the user and discard a half according to the user's choice. Estimation proceeds in the same way until the size of the remaining range turns to be less than some threshold.

Unfortunately, the above estimation process of user's liking cannot be sraightforwardly extended to two features in a two-dimensional space. The easiest way is to select on-axis candidates and to estimate the range feature-wise, but we consider a more general case where candidates are selected from an arbitrary axis pivoting at the centre. Suppose that there are two features, and we present two candidates to the user at each turn. In this case, however, we can see that a single turn is not sufficient to determine the right direction to proceed, because we cannot determine which of the features the user takes more seriously when she chooses one candidate. In other words, one feature of a candidate chosen by

the user may not be preferable, but she chooses the candidate in favour of the other feature. So we have to take another axis that is orthogonal to the previous one and present another pair of candidates, augmenting the information about user's liking. If we extend the argument to n-dimensional case this way, the situation gets worse and more complicated since we need an exponential number of candidate pairs, i.e. 2^{n-1}, to infer user's liking.

Our solution to treat a high-dimensional feature space is to reduce its dimensionality. We employ classical MDS using Euclidean distance as the distance measure [5]. We focus our argument on reduction to a two-dimensional space in this paper, but the algorithms themselves could be applied to other cases.

To estimate original feature values from coordinates in the reduced space, we can employ linear regression. By calculating ranges of the original features from a survived region in the reduced space, we can predict whether a new object is *preferred* by a user. Regression is also applied to check whether a selected pair of candidates is appropriate to be labelled by a user because magnitude relation of a feature of the candidate pair does not necessarily coincide with the slope of a regression plane. Candidate pairs that have a feature whose magnitude relation does not coincide with a regression plane are not presented, and this regularises the learning process of user's liking as ranges in the original space.

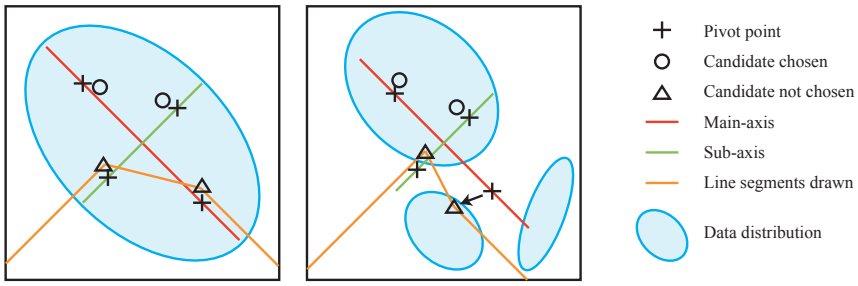
2.2 Algorithm of Geometrical Exclusion

After reducing the dimensionality to two, we partition the search space so as to make it filled with tiles. Hereafter we can employ various kinds of operations applicable to a pixelated plane by treating a tile as a pixel, including elementary geometrical operations like drawing a line. The learning process is therefore easily and directly visualisable. We believe that this property of our algorithm makes learning process fairly tractable and parameter tuning relatively straightforward.

The algorithm of geometrical exclusion is shown in Fig. 1. We first calculate the mean m of the distribution of remaining data in the reduced space. Next, we find a main-axis a_1 so that the divergence along a_1 is the largest, by applying principal component analysis after setting m as the origin of the space, and then find a sub-axis a_2 which is orthogonal to a_1 and goes through m (line 5). For each axis a_i (in a_1 and a_2), we calculate the distribution of remaining data along a_i, then select pivot points p_1 and p_2 on a_i at 20 and 80 percentile respectively (line 7). We find candidates c_1 and c_2 so that they are the nearest to p_1 and p_2 respectively and their magnitude relations of all features coincide with the ones estimated from coordinates in the reduced space and regression planes for each feature (lines 8-10). We then display c_1 and c_2 to the user and ask her which one she prefers (line 11). Let r_i^p and r_i^n be the candidate points that are preferred and not preferred respectively (line 12). Then we determine a region to be excluded based on r_1^n and r_2^n, collect tiles in the region and exclude data on the tiles as follows. We set a boundary consisting of one line segment l_1 from r_1^n to r_2^n and two half-lines l_2 and l_3 (line 14). One half-line starts from r_1^n, taking the same direction as the direction from m to a pivot point from which r_1^n is selected. The other half-line is drawn likewise. We divide the space by the

1: Initialise search space S
2: **repeat**
3: Fill S by colour k_1
4: Analyse S by principal component analysis
5: $a_{1,2} \leftarrow$ axes corresponding to two eigenvalues
6: **for** $i \leftarrow 1, 2$ **do**
7: Select pivot points $p_{1,2}$ from axis a_i
8: **repeat**
9: Select candidates $c_{1,2}$ around $p_{1,2}$ respectively
10: **until** *coincide*?$(S, c_{1,2})$
11: Display candidates $c_{1,2}$ and wait response
12: $r_i^p \leftarrow$ candidate chosen, $r_i^n \leftarrow$ candidate not chosen
13: **end for**
14: Place line segments $l_{1,2,3}$ based on the position of $r_{1,2}^n$
15: Draw $l_{1,2,3}$ on S by colour k_3
16: Select an arbitrary tile t not belonging to $l_{1,2,3}$
17: Flood-fill S from t by colour k_2
18: $\boldsymbol{k} \leftarrow$ colours of $r_{1,2}^p$ and the mean
19: $u \leftarrow$ subscript of the majority of colours in $\boldsymbol{k}$
20: Exclude k_{3-u} and k_3 coloured tiles from S
21: **until** *converged*?(S)

Fig. 1. Geometrical exclusion algorithm.



(a) Convex case. (b) Concave case.

Fig. 2. Schematic diagrams of geometrical exclusion.

boundary (lines 15-18). Then we decide which region is *not* to be excluded based on the majority rule (line 19). The region which contains more points within r_1^p, r_2^p and m than the other is to be remained. Complement of the majority region (i.e. the boundary and the minority region) is to be excluded (line 20).

Our strategy to determine the excluded region is to collect tiles in the region by colouring them like drawing a picture. The shape of the excluded region may be concave (see Fig. 2), so its formal analysis is difficult because it needs careful treatment of exceptions. Instead, we simply draw a boundary and fill a region isolated by the boundary. We employ Bresenham's algorithm [4] for the former, and a classic flood fill algorithm for the latter respectively.

Table 1. Correlation matrices used in the experiment.

(1) Rental residences.

F.1	F.2	F.3	F.4
1.0	-	-	-
0.8	1.0	-	-
-0.2	0	1.0	-
-0.1	0.3	0.0	1.0

(2) Three separated groups.

F.1	F.2	F.3	F.4
1.0	-	-	-
0.9	1.0	-	-
0	0	1.0	-
0	0	0	1.0

(3) Three groups.

F.1	F.2	F.3	F.4
1.0	-	-	-
0.9	1.0	-	-
-0.1	0.3	1.0	-
0.2	-0.2	0.1	1.0

(4) Two separated groups.

F.1	F.2	F.3	F.4
1.0	-	-	-
0.85	1.0	-	-
0	0	1.0	-
0	0	-0.7	1.0

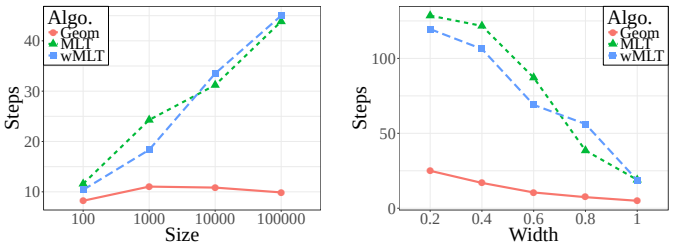
3 Experimental Evaluation by Simulation

We conducted an experiment to evaluate the performance of our algorithm, with artificially generated datasets and a user behaviour simulator. The reason to rely on a simulation is that we need to compare the resulting performance to those of other systems, measured under the same experimental setting and user inputs.

The datasets consists of randomly generated objects, each of whom has four features. Each feature is real-valued and sampled from the standard normal distribution. We prepared several number of datasets with differing sizes and correlations of features. Correlation between each feature is one of the key factors of our algorithm's performance, so we tested against four configurations of feature correlations, as shown in Table 1. *Rental residences* configuration is reproduced from the correlation matrix of the actual rental residences dataset, and the other three configurations are artificially chosen for comparison.

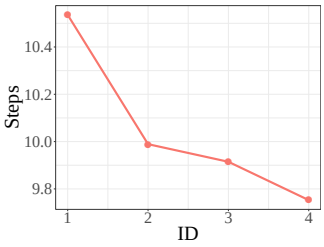
The design of the simulator roughly follows a way employed in [13]. In each trial, the simulator randomly selects an object from the dataset at first, and it is used as a *target object* during the trial. Additionally, a set consisting of objects similar to the target object, including the target object itself, found within the dataset is constructed, and we call it the *target set* of the trial. Members of the target set are objects whose every feature value falls within a certain range, centred at values of the target object. We tested several number of width configurations of the range, relative to the standard deviation σ of the feature values, from $\pm 0.2\sigma$ to $\pm 1.0\sigma$. Next, the simulator runs an algorithm to be tested and get two objects recommended by the algorithm. If at least one of the recommended objects matched an object in the target set, the trial finishes successfully. When both objects failed to match, the simulator calculates the Euclidean distance between the target object and each recommended object and takes the closer one to the target object. The object chosen above is passed to the algorithm as if it were a user input. Then the algorithm produces next recommendation, and the simulation iterates this way automatically.

To make an automatic simulation possible, without relying upon actual users' behavioural data, we employed an experimental setting that reflects reality in some degree, though significantly simplified. A user can compare two items based on Euclidean distance to the target item, and she chooses a closer one as her preferred item. This is based on assumptions that users have clear figures of ideal

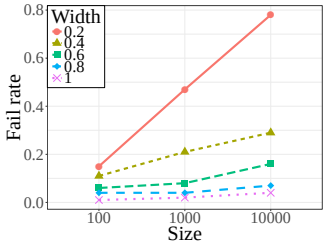


(a) Number of steps with varying data sizes.

(b) Number of steps with varying width.



(c) Number of steps with varying correlation coefficients.



(d) Fail rate with varying data sizes and width.

Fig. 3. Average number of steps and fail rate. Default data size, width and correlation configuration are 10,000, 0.6 and *rental residences* resp.

Table 2. Average sizes of target sets. Data size is 10,000 and correlation configuration is *rental residences*.

Width	$\pm 0.2\sigma$	$\pm 0.4\sigma$	$\pm 0.6\sigma$	$\pm 0.8\sigma$	$\pm 1.0\sigma$
Target set size	4.9	52.5	218.1	559.6	1099.7

items they want to buy, and also they can precisely measure the distance between an ideal item and a presented one, comparing each feature independently. This may seem too unrealistic, but we want to show basic characteristics and relative advantage of our algorithm here, in a clear setting involving few confounding factors, ensured by such strong assumptions.

We compared the performance of three algorithms with slight modifications: *Geometrical Exclusion*, *More Like This (MLT)* [13], *Weighted More Like This (wMLT)* [13]. In order to make wMLT work, we had to discretise values of features before comparison. We are aware of potential influences that the way of this discretisation may have, but we simply divide the value range of each feature into 0.2σ width ranges in this experiment, to keep the number of parameters we need to examine as few as possible. Another difference to McGinty's setting is that the initial pair of objects to be presented for MLT and wMLT are selected randomly in our experiment.

We ran the simulation with varying data sizes, width of the target set range and correlation configurations. Results are averaged over 100 runs for each set-

ting and are shown in Fig. 3 and Table 2. Fig. 3(a) clearly indicates that our algorithm is able to produce favourable recommendations within the range of practical number of steps under a realistic setting. The number of steps necessary to recommend a preferred item varies steadily around 10 for all size settings. This contrasts with that of the MLT and wMLT algorithms, whose number of steps increases logarithmically as the data size increases. The relative target set size for the case of 10,000 objects, $\pm 0.6\sigma$ width and *rental residences* correlation configuration is about 2 %, as shown in Table 2, and this is a moderate number for our service. Standard deviation of the steps under the above setting is 8.57 for our algorithm, whereas that of MLT and wMLT are 18.87 and 43.16 respectively. Fig. 3(b) shows that the number of steps decreases when the target set size increases for all three algorithms, and this conforms to our intuition. Our algorithm outperformed MLT and wMLT in all five width settings. Correlation configurations are arranged from the most difficult to recommend, *rental residences*, to the easiest, *two separated groups*. The number of steps necessary for our algorithm, shown in Fig. 3(c), decently reflects those relative difficulties.

Our algorithm occasionally fails to recommend a preferred item because its tile-wise exclusion mechanism may drop all the tiles that contain target objects before presenting them to the user. In Fig. 3(d), fail rate statistics with varying data sizes and width are shown. Fail rates for the range of width broader than or equal to $\pm 0.6\sigma$ are kept fairly low for all data sizes.

4 Deployment as a Web Service

We implemented the proposed algorithm as an online recommender system embedded in a commercial web service to search rental residences, and we made our service publicly accessible during five weeks. During this period, we collected log data to record user behaviour, from which we obtained statistics shown below.

The rental residence dataset of our system has four integer-valued features: monthly rent, floor area, age and distance from the nearby station. These features are selected out of several tens of available features, considering suggestions by domain experts. The dataset is grouped by nearby train/subway station. The number of groups is several thousand, and the number of residences contained in each group ranges from one to nearly ten thousand.

A user of our service is asked to choose a station first. Then the system loads the dataset and creates an initial model specific to that station. After the setup, two residences are displayed and the user is asked to choose more preferable one (see Fig. 4). The learning step progresses in accordance with the algorithm of the geometrical exclusion, however, to enhance usability, we made a modification so that the steps cannot be repeated more than four times. After the learning steps, the system computes ranges of the features, which represent user's likings for rental residences. Finally, residences classified as *preferred* are displayed.

As a result of the publicly opened deployment, the total number of sessions was 6,318, though, the number of sessions with user activity was 2,963 (46.9 %), meaning that more than half of the sessions finished with no user responses.

Fig. 4. A screenshot of the prototype displaying 1. floor plans and 2. features of two residences. When a user presses 3. one of the buttons at the bottom which is under a preferable residence, 4. number of operations left is decremented and new residences are shown. Several parts containing sensitive information are blurred.

Meanwhile, 1,970 (31.2 %) sessions were *completed*, that is, substantial amount of users finished the learning process and got final recommendations. The number of unique users who completed their sessions was 1,455, which was 38.1 % of the total number of unique users, 3,815. The average number of sessions per user was 1.83, and its standard deviation was 8.63. Average time consumed to finish the process was 144 seconds, and its standard deviation was 147. Considering the above rate of completion, this falls within a tolerable range of ordinary users.

5 Conclusion and Future Work

We presented a novel recommender system that directly utilises a two-dimensional space, reduced from the feature space, to embed a learning model for semisupervised classification. The proposed learning algorithm is based on geometrical operations on the reduced space. By a simulated experiment, we showed that our system is able to estimate preferred ranges of features effectively. The number of steps necessary to complete the estimation is kept within a practical range and stable enough with varying sizes of item sets. We believe that this property made our system amusing for many users involved in an open trial, and thus, it resulted in high completion rate.

As future work, we plan to formally analyse properties of our algorithm such as convergence of the learning process by geometrical exclusion. In addition, it is desirable to measure our system's performance in a "real" setting rather than in a simulated experiment. Ideally, recommender systems' performance should be compared in a practical production environment in an A/B test fashion.

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 Trans. on Knowl. and Data Eng. 17(6), 734–749 (2005)
2. Basalaj, W.: Incremental multidimensional scaling method for database visualization. Proc. Visual Data Exploration and Analysis VI, SPIE 3643, 149–158 (1999)
3. Baur, D., Boring, S., Butz, A.: Rush: Repeated recommendations on mobile devices. In: Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI '10). pp. 91–100. ACM (2010)
4. Bresenham, J.E.: Algorithm for computer control of a digital plotter. IBM Syst. J. 4(1), 25–30 (1965)
5. Cox, T.F., Cox, M.: Multidimensional Scaling, Second Edition. Chapman and Hall/CRC (2000)
6. Frank, J., Lidy, T., Hlavac, P., Rauber, A.: Map-based music interfaces for mobile devices. In: Proceedings of the 16th ACM International Conference on Multimedia (MM '08). pp. 981–982. ACM (2008)
7. Giamattei, M., Scholz, M.: Exploiting correspondence analysis to visualize product spaces. In: Proceedings of the 7th Conference of the Italian Chapter of AIS (2010)
8. Grad-Gyenge, L., Kiss, A., Filzmoser, P.: Graph embedding based recommendation techniques on the knowledge graph. In: Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization. pp. 354–359 (2017)
9. Kagie, M., van Wezel, M., Groenen, P.: Map Based Visualization of Product Catalogs. ERIM Report Series Research in Management ERS-2009-010-MKT, Erasmus Research Institute of Management (ERIM) (2009)
10. Kagie, M., van Wezel, M., Groenen, P.J.F.: A graphical shopping interface based on product attributes. Decis. Support Syst. 46(1), 265–276 (2008)
11. Khoshneshin, M., Street, W.N.: Collaborative filtering via euclidean embedding. In: Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10). pp. 87–94. ACM (2010)
12. Le, D.D., Lauw, H.W.: Euclidean co-embedding of ordinal data for multi-type visualization. In: Proceedings of the 2016 SIAM International Conference on Data Mining. pp. 396–404 (2016)
13. McGinty, L., Smyth, B.: Comparison-based recommendation. In: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning (ECCBR '02). pp. 575–589. Springer-Verlag (2002)
14. O'Shaughnessy, M.R., Davenport, M.A.: Localizing users and items from paired comparisons. In: 26th IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2016). pp. 1–6 (2016)
15. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02). pp. 253–260. ACM (2002)
16. Trosset, M.W., Priebe, C.E., Park, Y., Miller, M.I.: Semisupervised learning from dissimilarity data. Comput. Stat. Data Anal. 52(10), 4643–4657 (2008)
17. Yang, X., Guo, Y., Liu, Y., Steck, H.: A survey of collaborative filtering based social recommender systems. Comput. Commun. 41, 1–10 (2014)
18. Zhang, J., Pu, P.: A comparative study of compound critique generation in conversational recommender systems. In: Proceedings of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'06). pp. 234–243. Springer-Verlag (2006)