Proposal and Evaluation of Serendipitous Recommendation Method Using General Unexpectedness

Takayuki Akiyama
Hitachi, Ltd., Central Research Laboratory
1-280, Higashi-Koigakubo,
Kokubunji-shi, Tokyo
185-8601 Japan
Tel: +81-42-323-1111 ext. 4302
takayuki.akiyama.hv@hitachi.com

Kiyohiro Obara
Hitachi, Ltd., Central Research Laboratory
1-280, Higashi-Koigakubo
Kokubunji-shi, Tokyo
185-8601 Japan
Tel: +81-42-323-1111 ext. 3612
kiyohiro.obara.pc@hitachi.com

Masaaki Tanizaki
Hitachi, Ltd., Central Research Laboratory
1-280, Higashi-Koigakubo
Kokubunji-shi, Tokyo
185-8601 Japan
Tel: +81-42-323-1111 ext. 4068
masaaki.tanizaki.tj@hitachi.com

ABSTRACT

Recommender systems support users in selecting items and services in an information-rich environment. Although recommender systems have been improved in terms of accuracy, such systems are still insufficient in terms of novelty and serendipity, giving unsatisfactory results to users. Two methods of "serendipitous recommendation" are therefore proposed. However, a method for recommending serendipitous items accurately to users does not yet exist, because what kinds of items are serendipitous is not clearly defined. Accordingly, a human preference model of serendipitous items based on actual data concerning a user's impression collected by questionnaires was devised. Two serendipitous recommendation methods based on the model were devised and evaluated according to a user's actual impression. The evaluation results show that one of these recommendation methods, the one using general unexpectedness independent of user profiles, can recommend the serendipitous items accurately.

Categories and Subject Descriptors

H.1.2 [**Models and Principles**]: User/Machine Interface – *Human factors, Human information processing.*

General Terms

Human Factors

Keywords

Recommender systems, user preference, content-based, serendipity, unexpected.

1. INTRODUCTION

In recent years, the amount of information accessible to users is increasing and becoming more diversified because of the growth of information technology and the expansion of commercial use of IT. Under this circumstance, although users can select various items (such as information, TV programs, and books) they cannot select the best of those items from a vast amount of items including many useless items.

To solve this problem, so-called "recommender systems"—for recommending suitable items to users by monitoring a user's action and extracting information concerning a user's preferences—are becoming necessary for "item-providing services" such as internet shopping sites and department stores. In the future, recommender systems will recommend items by monitoring all a user's preferences. Users will get information suitable for their needs, and they will have an opportunity to discover new items. Moreover, service providers will be able to provide services continuously because users will use their systems more frequently.

Recommendation technology is one way to retrieve information that suits a user's preferences. In information-retrieval theory, useful information is categorized as two types: that which users recognize as useful, and that which users do not recognize as useful but is actually useful [1]. We suppose that the items users like are categorized as the same two types; accordingly, in this paper, the second type of items is defined as "serendipitous items."

In general, typical recommender systems use either of two strategies: a content-based approach or collaborative filtering [2]. The content-based approach recommends items similar to users selected items by calculating the similarity between items by using feature vectors generated by extraction of a user's selection record. Collaborative filtering recommends items selected by multiple users whose selection histories are similar to the relevant user by calculating similarity between users' records.

These two methods recommend items similar to the ones that the user selected before. These items belong to the first type stated above because they are recognized as interesting items by users. For example, a typical recommendation recommends TV programs featuring actor A to users who frequently watch TV programs featuring actor A. Consequently, a user might get bored with typical recommendation because it always recommends similar items that a user already knows are interesting [2]. For that reason, recommending items belonging to the second type–namely, serendipitous ones—become necessary. For example, serendipitous recommendation recommends educational programs featuring performer A to users who do not usually watch educational programs but frequently watch performer A.

Nevertheless, typical recommendation methods cannot recommend such serendipitous items preferentially.

The purpose of this study is to realize serendipitous recommendation. Accordingly, actual data that users recognized as “serendipitous” was collected, and a user-preference model was established first. Serendipitous recommendation methods based on that model were devised and evaluated with actual data. The results of this evaluation verified the effectiveness of a serendipitous recommendation method using “general unexpectedness” that is independent from a user’s profile.

2. RELATED WORKS AND MOTIVATION

In the early stage of developing recommendation systems, the accuracy of recommendation of the first-type items was improved. It was thought that this improved accuracy was enough to enhance user satisfaction. However, it is recognized that novelty and serendipity are important factors in satisfying a user, aside from simply suitability to a user’s preference [2, 3, 4, 5].

There are several related works on serendipitous recommendation. Ziegler et al. supposed that serendipitous items exist in recommendation lists of different items in different categories more than in the lists of similar items, and they proposed a recommendation method to increase diversity of recommendation lists [6, 7]. They defined “intra-list similarity” as the similarity between all items in a recommendation list by calculating similarity between two items. Moreover, they increased diversity by inserting low-similarity items.

Approaches that recommend serendipitous items directly have also been proposed. Hijikata et al. proposed a method for improving novelty and serendipity by calculating the probability of known items by using the information about knowns or unknowns given explicitly by user [8]. Another method calculates the probability of “degree of interest” by using an evaluation of items selected by a user (namely, “interested” or “not interested”). The items whose degree-of-interest probabilities are nearly equal are taken as serendipitous and recommended [9].

Another proposed method considers the items that are different from the ones users use habitually as serendipitous and recommends those [10]. This method uses a preference model to predict items that users like and a habit model to predict items that users use habitually. It then recommends a recommendation list including serendipitous items by predicting the unexpectedness of items by calculating differences between the results of the preference model and the habit model.

As mentioned above, the only serendipitous recommendation methods proposed until now are based on researchers’ own assumptions; no methods based on actual data regarding a user’s actual impression of selected items have been devised. Moreover, many works suppose that serendipitous items mean unexpected items, and they do not treat items that are unexpected and interesting.

In this study, the authors clarified what kinds of items are actually serendipitous by collecting data concerning a user’s actual impressions, made assumptions based on that actual data, and devised two serendipitous recommendation methods based on those assumptions.

3. MODELING SERENDIPITOUS ITEMS ACCORDING TO ANALYSIS OF ACTUAL DATA

3.1 User-preference model

The assumption of user preference was established first, and what kinds of items are serendipitous for users was verified by analyzing a user’s actual impressions collected by questionnaires based on this assumption. The user-preference model established before the questionnaires were given is explained in the following. Figure 1 shows the concept of the model. In this model, items are arranged in feature vector space generated by features of items. Although this feature vector space is highly dimensional, for simplicity, two-dimensional space is introduced in Figure 1. Items that a user selected before exist in the area near the feature vector that the user recognizes and knows are interesting (so-called “recognized items” below because the user recognizes them as interesting and not surprising if recommended). In a distant area from that area, serendipitous items (namely, surprising and interesting items) are supposed to exist. In an area far from the recognized area, not-interesting items are supposed to exist. Broadly speaking, it is supposed that each user has several recognized areas in the feature vector space, because there may be several reasons that the user selected certain interesting items; for example, the reasons for selecting a drama and a documentary program may be different.

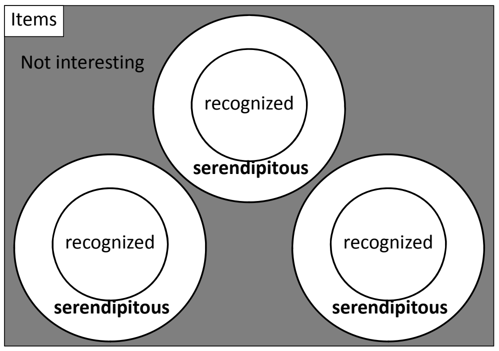

Fig. 1: Concept of user-preference model

3.2 Questionnaire

To collect users’ actual impressions, a questionnaire was given to thirty users. The method is mentioned below. First, users read the information concerning a TV program selected randomly from TV programs over three months (31,433 programs), and then they classify these TV programs as recommended items into three categories: “recognized program” (first-type item), “serendipitous program” (second type) and “not-interesting program.” An electric program guide (EPG) is used to provide the information concerning TV programs, which includes title, performer, and the other contents of programs.

In the questionnaire, three categories are available for choice by users. “Recognized program” means programs that users can expect from their own preference, for example, programs that users frequently watch. “Serendipitous program” means programs that users feel are interesting and surprising when recommended,

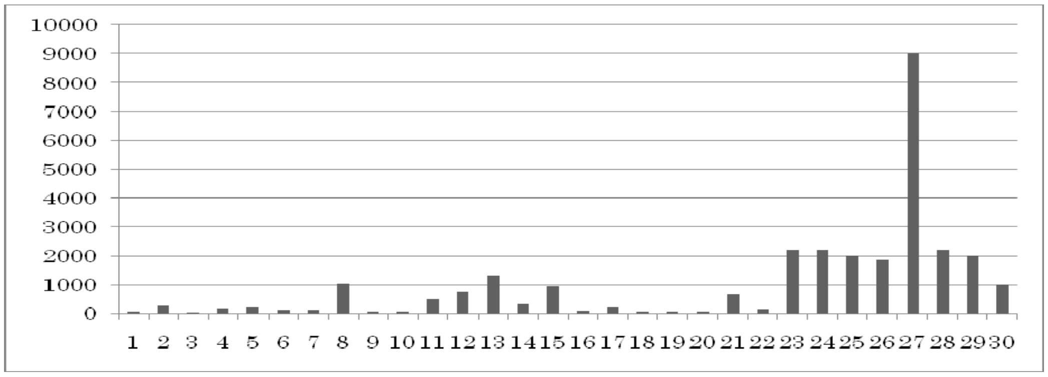
Fig. 2: Number of evaluated programs by each user

Vertical axis: Number of evaluated programs, Horizontal axis: User ID

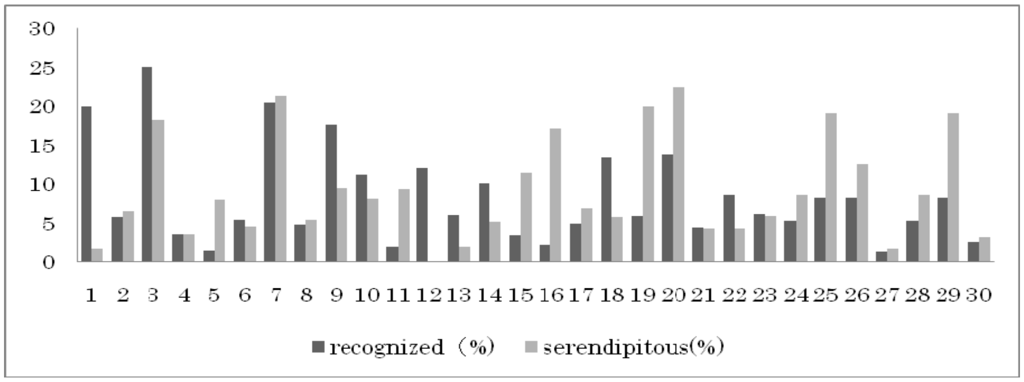

Fig. 3: Ratio of recognized programs and serendipitous programs in all programs for users

Vertical axis: Rate of each program in all programs, Horizontal axis: User ID

for example, programs that users do not expect from their own preferences but are interested in. "Not-interesting program" means programs that users are not interested in even though recommended.

It takes much time to answer this questionnaire (about one minute per program evaluation), so each user answered the questionnaire over one month, from ten to one hundred answers per day. We supposed that a user's preference does not change much over one month, because a series of TV programs lasts about three months.

All users live in Japan, twenty six work at Hitachi, Ltd., Central Research Laboratory and four are university students. Twenty five are male, and five are female. Fifteen are from twenty to thirty years old, eleven from thirty to forty, and the other four from forty to fifty. Each user evaluated about one thousand to five thousand programs.

3.3 Analysis method

The programs collected by questionnaire are first converted into term vectors extraction by morphological analysis of text information in the EPG. Each vector component contains two values, whether the EPG text includes the term or not. The recognized programs are then clustered to estimate the recognized area. For clustering, the distance between program Pi and program Pj is defined as

$$\mathrm{distance}(P_i, P_j) = \sum_{n=1}^{N} w_n \left| P_i(n) - P_j(n) \right| \quad (1)$$

where Pi(n) means the vector component of the *n*th term in program Pi, whether program Pi includes the *n*th term or not (1 or 0), w_n means the user's weight (a metric of user's preference) of the *n*th term. The user's own distance between programs is determined by introducing user's weight w_n.

Weight w_n of *n*th term *v* is calculated by TFIDF (product of term frequency and inverse document frequency) [11]. TFIDF is a metric of weighting characteristic terms occurring in observed documents by frequency in observed groups and in all groups. This metric is introduced to weight a user's preference as follows.

$$w_n = \mathrm{tfidf}(v \mid D) = \mathrm{tf}(v \mid D) \times \log\left(\frac{N_{all}}{N(v)}\right) \quad (2)$$

Here, D represents observed program, which means recognized programs here, tf(v|D) means the frequency that term *v* occurs in D, N_{all} means the total number of programs, and N(*v*) means the number of occurrences of term *v* in all programs.

3.4 Results

Figure 2 shows the number of programs evaluated by each user, and Figure 3 shows the ratio of recognized programs and serendipitous programs in all programs. Although each user has various ratios, it is clear the rates of recognized programs are very low and there are a lot of inefficient programs. It is also clear that users who frequently watch TV programs evaluate more programs as recognized rather than serendipitous. On the other hand, the users who rarely watch TV programs evaluate more programs as serendipitous rather than recognized.

In regard to the questionnaire, most users said they feel serendipitous concerning the programs that they do not know before but are interesting (for example, interesting educational programs for users who do not watch educational programs) and the programs including an unexpected combination of interesting features (for example, educational programs featuring a comedian). However, surprising programs are not always unexpected programs, so the meanings of surprising would include other factors. Moreover, some users evaluated no programs as serendipitous, and some users cannot classify programs into the three types; consequently, it is difficult to evaluate their subjective impression quantitatively.

A clustering result of recognized programs is shown as dendrogram in Figure 4. The clustering method used is hierarchical clustering. The height of the cluster means average distance between programs belonging to the cluster and the cluster center calculated from Equation (1). The number of recognized areas is determined by cutting at a certain height of a cluster.

Figure 5 shows the ratio of average distance of recognized programs (radius of recognized area) and average distance of not-interesting programs (radius of not-interesting area) from the nearest center of the cluster with height of clusters. When the number of clusters increases, not-interesting programs are distributed outside of recognized area. On the other hand, when the number of clusters decreases, not-interesting programs are distributed inside the recognized area because the number of clusters is fewer than the true number of recognized areas.

Figure 6 shows the ratio of average distance of serendipitous programs (radius of serendipitous area) and the radius of a recognized area from the nearest center of the cluster with height of clusters. As the number of clusters increases, serendipitous programs are distributed outside of the recognized area.

Figure 7 plots the results from Figures 5 and 6. It is indicated that not-interesting programs are distributed outside the recognized area, and serendipitous programs are distributed far outside the recognized area.

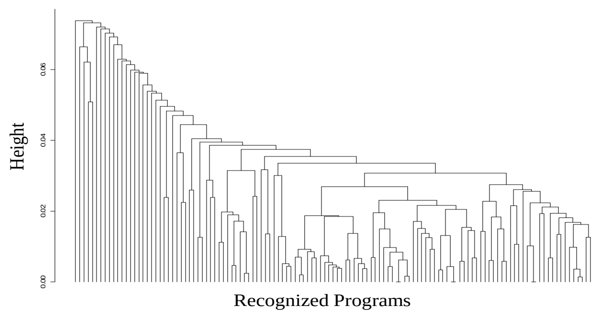

Fig. 4: Clustering result of recognized programs

(leaf nodes: recognized programs; vertical axis: height of cluster)

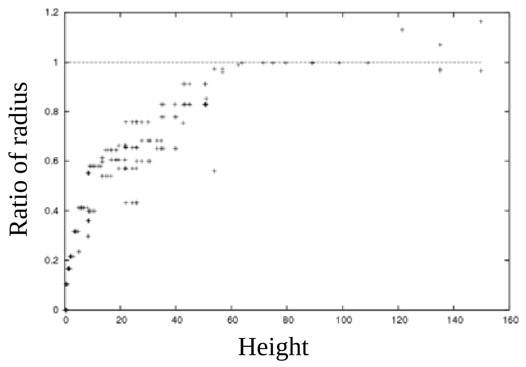

Fig. 5: Ratio of radiuses of recognized area and not-interesting area with height of cluster

(Denomination: radius of not-interesting area)

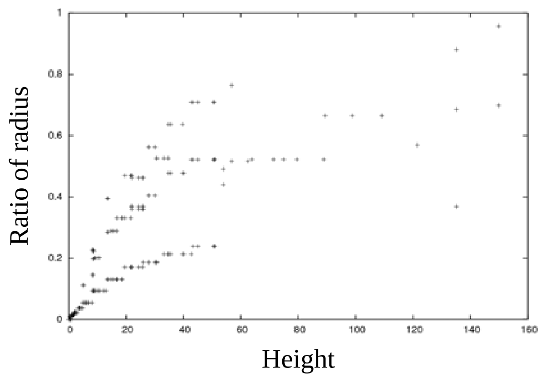

Fig. 6: Ratio of radiuses of recognized area and serendipitous area with height of cluster

(Denomination: radius of recognized area)

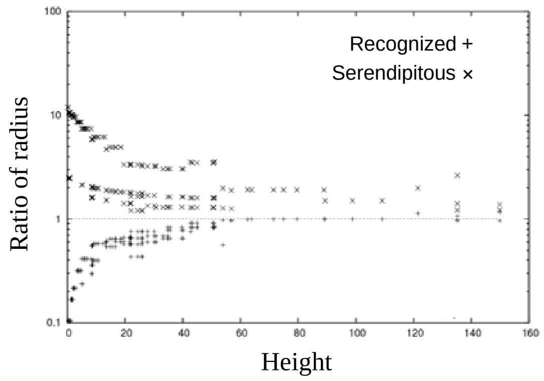

Fig. 7: Ratio of radiuses of not-interesting area, recognized area, and serendipitous area with height of cluster

(Denomination: not-interesting area)

3.5 Model based on analysis results

To summarize the results presented in this section, in the feature vector space generated by EPG texts, not-interesting programs are distributed outside the recognized area and serendipitous programs are distributed far outside the recognized area. This result does not support the assumption in Figure 1. We therefore suggest the structure of user preference as shown in Figure 8 instead of that shown in Figure 1.

Distance from the center of the recognized area means the number of terms in the program vector but not in the center because the program-vector components are described by only two values, whether each term in the contents of programs is included or not. In addition, the weight of terms is calculated as a user's preference by TFIDF. Therefore, even though the item includes many low-weight terms and is rarely watched, the distance from the recognized area is not far. And if the program includes high-weight terms belonging to the other recognized area, the distance from recognized area becomes far. Consequently, programs including many high-weight terms belonging to the other recognized area and not similar to the ones in the nearest recognized area are distributed in the intermediate region of recognized areas, and users treat them as serendipitous programs. This assumption expresses that "the contents makes users feel serendipity concerning an unexpected combination of program contents," which some users commented in the questionnaire.

Figure 9 shows the distribution of each type of program plotted against distance from one center of a recognized area. The solid line represents the distribution of not-interesting programs, the dotted line represents the distribution of serendipitous programs, and the dashed line represents distribution of recognized programs. The nearest peak of recognized programs to the origin represents the peak of the distribution of the recognized area, and the next-nearest peak represents several recognized areas. As shown in Fig. 7, not-interesting programs are distributed broadly both in the recognized area and the serendipitous area; consequently, it is difficult to distinguish only serendipitous programs accurately by distance between programs given by Equation (1).

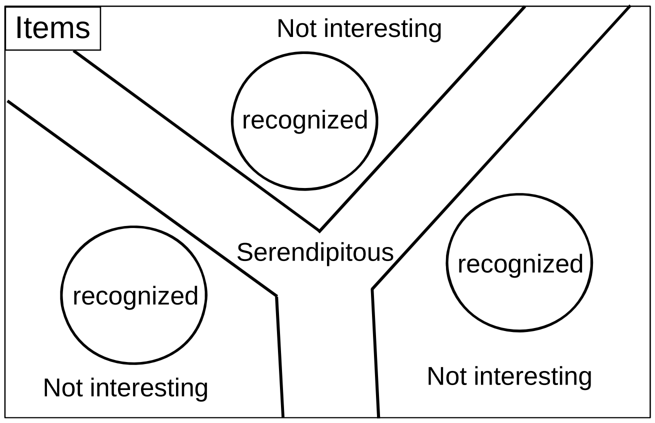

Fig. 8: User-preference model based on analysis results

4. PROPOSAL AND EVALUATION OF RECOMMENDATION METHODS

4.1 Proposed methods

4.1.1 Using distance between items

The distance between items used in this method is calculated from Equation (1) reflecting a user's preference. First, the proposed

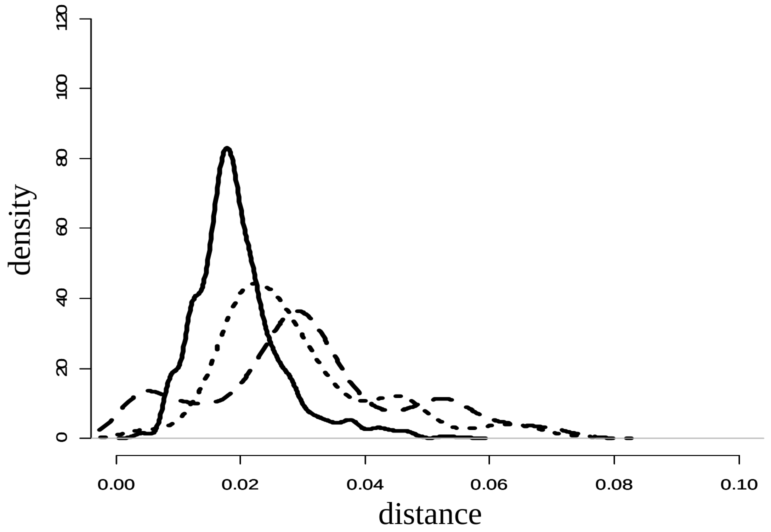

Fig. 9: Density of programs in each area with distance from center of recognized area

(Vertical axis: density of programs, horizontal axis: distance from center of recognized area,

solid line: not interesting programs, dotted line: serendipitous programs, dashed line: recognized programs) recommender system learns features of programs according to the user's viewing history. In the same way as described in section 3, program vectors are defined by a term vector, whose component has two values. Second, the system splits watched programs (i.e., recognized programs) into several clusters by hierarchical clustering and finds the centers of recognized areas. The number of recognized areas is defined as 7 to 10 according to the results from the questionnaire. The system then calculates the distance of each not-watched program from the nearest center, and recommends the 10 longest programs. In short, the system recommends 10 highest score programs calculated according to

$$\text{Score}(P_i) = \text{distance}(P_i, C_{nearest}) \qquad (3)$$

Here, C_{nearest} means the center of the nearest recognized area with program Pi.

This method may not recommend serendipitous programs accurately because not-interesting programs are distributed broadly. This method is referred to as the "first method" hereafter.

4.1.2 Using general unexpectedness

This method (hereafter, "second method") introduces "unexpectedness of programs" in addition to the distance used in the first method in order to capture a "surprising" factor. The results of the questionnaire indicate that the serendipitous programs have an unexpected aspect for users, as shown in Fig. 8. It is assumed that "unexpectedness" means something is hard to predict the program contents. Regarding a program-recommendation system, it is assumed that it is related to an unlikely combination of features. The second method treats highly unexpected and interesting programs as serendipitous programs. A general metric of difficulty of expecting programs for every user is defined by the sum of the tendencies of co-occurrence of the terms in the program.

$$\begin{aligned} &\text{Expectedness}(P_i) \\ &= \frac{1}{|P_i|} \sum_{v,w \in P_i} \text{Tendency of co-occurrence}(v, w) \qquad (4) \\ &= \frac{1}{|P_i|} \sum_{v,w \in P_i} \frac{Nvw}{Nv + Nw - Nvw} \end{aligned}$$

Tendency of co-occurrence (v, w) means tendency of co-occurrence of terms *v* and *w* in all programs. It makes it possible to evaluate quantitatively how unexpected a program is for users. N*v* means number of programs including term *v*, and N*vw* means number of programs including both term *v* and *w*. |Pi| means number of terms in program Pi and is a normalized factor. If the co-occurrence of the terms is low, expectedness will be low, and the program will be highly unexpected, so users would be unable to find it. Unexpectedness is defined as the inverse of expectedness (see Equation (6)), and 10 high-score programs (calculated according to the sum of squares of distance between programs and unexpectedness as below) are recommended.

$$\begin{aligned} \text{Score}(P_i) = {} & \alpha \times \text{distance}(P_i, C_{nearest})^2 \\ & + (1-\alpha) \times \text{Unexpectedness}(P_i)^2 \end{aligned} \qquad (5)$$

$$\text{Unexpectedness}(P_i) = \frac{1}{\text{Expectedness}(P_i)} \qquad (6)$$

Parameter α controls the degree of combination of a user's preference and unexpectedness of programs. Simply put, equation (5) is a very simple linear combination of squares of distance and unexpectedness.

4.2 Evaluation method

4.2.1 Dataset

The results of the questionnaire implemented in the third section were used. Data of fourteen users who classified more than 100 programs into recognized or serendipitous programs were selected, because it was supposed that serendipitous recommendation becomes necessary after watching TV programs for about one month. (It was assumed that users get bored with typical recommendation after about one month and most users watch fifty TV programs per month). Each user evaluated from 1000 to 5000 programs, and the ratio of serendipitous programs in all evaluated programs is 7 to 8%.

4.2.2 Procedure

The three proposed methods are applied to each user. The procedure is mentioned below. First, the system learns recognized areas from fifty recognized programs. In this evaluation experiment, 50 recognized programs were prepared randomly as a training set from evaluated programs as recognized. Next, the system recommends ten high-score programs by using the proposed methods, random recommendation, and a method using only unexpectedness for each user from the remaining evaluated programs by using the recognized areas learned first. Random recommendation means recommending ten programs randomly, and the method using only unexpectedness calculates a program score according to unexpectedness only (α =0 in Equation (5)). This experiment was performed ten times, and each time different recognized programs were used and the accuracy of each method was compared.

4.2.3 Evaluation metrics

Our purpose is to recommend serendipitous programs. So we use detection rate and precision as evaluation metrics for the purpose of evaluating accuracy of the proposed methods to detect serendipitous programs. Detection rate means the probability of detecting a serendipitous program and precision means rate of serendipitous programs in recommendation list.

4.3 Results

Table 1 lists the evaluation results of the two proposed methods, random recommendation and only unexpectedness. Accuracy metrics are calculated as an average of users. Parameter α is set to 0.05, so the second method has the highest accuracy, .

The results in Table 1 show that detection rate and precision of random recommendation are low, so it suggests how difficult it is to recommend serendipitous programs. On the other hand, the accuracy of the second method (i.e., using unexpectedness of programs) is higher than the other methods, detection rate is 78.2% and precision is 21.6%. This result means that the second method recommends serendipitous programs accurately.

Table 1: Accuracy results

Method	Random	First	Second	Only unexpectedness
Detection Rate [%]	51.9	49.8	78.2	32.8
Precision [%]	7.98	7.51	21.6	5.21

While accuracy of the first method (i.e., using distance only) is the same as that of the random method, accuracy of the second method is much higher than the random one, and accuracy of the unexpectedness-only method is lower than that of the random one. This result shows it is possible to recommend serendipitous programs by using both distance reflecting a user's preference and unexpectedness of programs.

The first method recommends programs including not-interesting ones, because it recommends items that are not similar to recognized programs. On the other hand, the second method distinguishes "unexpected and interesting programs" and "unexpected but not-interesting programs" from programs with low similarity according to unexpectedness. Consequently, the accuracy of the second method is high.

Figure 10 shows the concept of user preference by distance and unexpectedness inferred from these results. Serendipitous programs and not-interesting programs are distant from the recognized area. According to the result "only unexpectedness" in Table 1, serendipitous programs exist in extra high-unexpectedness areas because they tend to have more combinations of terms whose tendency of co-occurrence is low. Moreover, in the right lower box, not-interesting programs may exist. It seems very possible that the user would already know the highly unexpected programs near to recognized programs and not select them, because "unexpectedness" is a general metric and does not depend on a user's record.

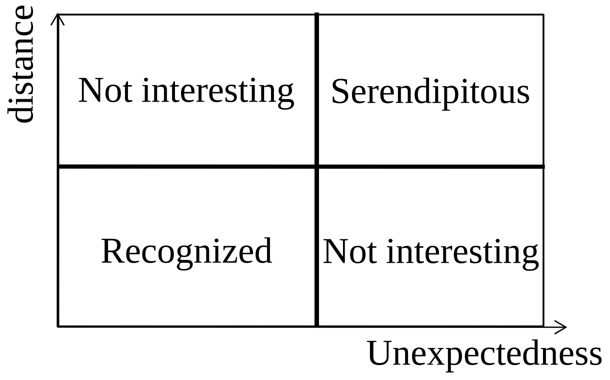

Fig. 10: Concept of user preference with distance and unexpectedness

Unexpectedness of programs calculated from tendency of co-occurrence of terms in the programs is introduced here. For example, users find programs by reading TV guides and EPGs on web sites. Therefore, programs that have rare contents in TV guides are supposed to be serendipitous. TV guides and EPG are not provided by users but by the surroundings of users, so we simply introduce unexpectedness independently from a user's characteristics. Examinees in this experiment are deemed to live in similar environments. The influence of unexpectedness for users living in totally different environments (e.g., living in different countries) might be significant. Unexpectedness may therefore be a frequency of contact with items similar to the relevant item which a user contacts with so far, with or without intention.

Finally, our proposed method is compared with the other related methods. It is hard to compare by accuracy because serendipity depends on user's subjective impression, so we compare these by requirements in Table. 2.

Table 2: Comparison of serendipitous recommendation

Requirement	Proposal	Different from Habit	Different from Interesting & Not	Collaborative
User's Impression	**Unnecessary**	**Unnecessary**	**Necessary**	**Unnecessary**
Other user's record	**Unnecessary**	**Unnecessary**	**Unnecessary**	**Necessary**
User's Habit	**Unnecessary**	**Necessary**	**Unnecessary**	**Unnecessary**
Information of Programs	**Necessary**	**Necessary**	**Necessary**	**Unnecessary**

Related works require some information concerning users, one requires a user's impression of recommended items, another requires other users' records, and another requires user's habits. The proposed recommendation method requires few evaluation values to learn a user's preference and does not depend on user's surroundings. On the other hand, it requires information concerning programs, but recently there is much information regarding programs on Internet reference sites like *Wikipedia*. In short, the proposal method has most broad utility regarding various systems because it is useful for both devices and servers. As for our future work, however, which method satisfies users must be verified by a user's subjective evaluation. On the other hand, we suggest using suitable terms for each user.

5. FUTURE WORK

Although the accuracy of our proposal serendipitous recommendation method was verified, the following three tasks remain as future work: improve accuracy, evaluate by more users, and tune performance of actual system

To improve accuracy, it is necessary to select the recognized area outside of which many serendipitous programs exist; in fact, there are some recognized areas outside of which serendipitous programs do not exist. By considering the radius and number of programs included in recognized areas, it is possible to select the best recognized area. Moreover, another approach to improving accuracy is to get rich information concerning programs via metadata and information on web sites.

It is also necessary to satisfy users by capturing user context with their spatial temporal information; for example, a user does not want to watch a program in the morning but in the evening instead. It is also important to capture time-dependent user preferences, for example, users feel serendipity if a recommended program was not watched recently but has been watched in the past. With our recommendation method, a user's preference is described in a feature vector space generated by the user's selection history, so the structure of the space and distribution of user preference depends on time.

To make the user-preference model statistically strong, it is necessary to evaluate our proposed method by more users, because the concept of serendipity is supposed to depend strongly on user's subjective impression. Moreover, it is important to establish methods for evaluating a user's satisfaction quantitatively.

To introduce our recommendation method in an actual system, it is necessary to design an optimal data structure and speed up the method.

In this study, we verified the recommendation method by using TV programs, but this approach can be applied to recommend items like books and DVDs from a user's record of selecting TV programs. We plan to use this approach to capture the meanings of users like and dislike by collecting and analyzing user's records.

6. CONCLUSION

To realize serendipitous recommendation, a recommendation method for extracting a user's preference was proposed and evaluated. In particular, based on actual data obtained by giving a questionnaire to thirty users, a user-preference model using distance between programs was established. Based on this model, a serendipitous recommendation method using the distance and unexpectedness of programs was proposed. This method recommends a serendipitous program accurately at a detection rate is 78.2%. Moreover, it was found that the impression of unexpectedness depends on a user's living environment rather than his or her character. This result is an important fact in regard to understanding a user's preference in principle.

7. REFERENCES

[1] E. Toms: Serendipitous Information Retrieval, Proc. of DELOS Workshop, 2000

[2] Herlocker, J., et al.: Evaluating Collaborative Filtering Recommender Systems, ACM Transactions on Information Systems, Vol. 22, No. 1, pp. 5-53 (2004)

[3] K. Swearingen and R. Sinha: Beyond Algorithms: An HCI Perspective on Recommender Systems, ACM SIGIR Workshop on Recommender Systems (2001)

[4] S. M. McNee, J. Riedl, and J. A. Konstan: Making Recommendations Better: An Analysis Model for Human-Recommender Interaction, In proc. of ACM Special Interest Group on Computer-Human Interaction (ACM SIGCHI), pp. 997-1101 (2006)

[5] S. M. McNee, J. Riedl, and J. A. Konstan: Being accurate is not enough: How accuracy metrics have hurt recommender systems, In proc. of ACM Special Interest Group on Computer-Human Interaction (ACM SIGCHI), pp. 997-1101 (2006)

[6] C. N. Ziegler, G. Lausen, and L. S. Thieme: Taxonomy-driven Computation of Product Recommendations, In proc. of the 2004 ACM CIKM Conference on Information and Knowledge Management, pp. 406-415 (2004)

[7] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and George Lausen: Improving Recommendation Lists Through Topic Diversification, In proc. of World Wide Web Conference, pp. 22-32 (2005)

[8] Y. Hijikata, T. Shimizu, and S. Nishida: Discovery-oriented Collaborative Filtering for Improving User Satisfaction, In proc. of the 14th ACM International Conference on Intelligent User Interfaces(ACM IUI 2009), pp. 67-76 (2009)

[9] Leo Iaquinta, Macro de Gemmis, Pasquale Lops, Giovanni Semeraro, Michele Filannino, and Piero Molino: Introducing Serendipity in a Content-based Recommender System, Hybrid Intelligent Systems, 2008. HIS '08. Eighth International Conference on 10-12 Sept. 2008, pages 168 – 173

[10] T. Murakami, et al.: A Method to Enhance Serendipity in Recommendation and its Evaluation, Transactions of the Japanese Society for Artificial Intelligence, Vol. 24, Issue 5, pp. 428-436 (2009).

[11] Li-Ping Jing, et al.: Improved Feature Selection Approach TFIDF In Text Mining, Proceedings of the First International Conference on Machine Learning and Cybernetics (2002)