# Generation of Hints to Overcome Difficulty in Operating Interactive Recommender Systems

Yuri Nakao
nakao.yuri@fujitsu.com
Fujitsu Laboratories Ltd.
Kawasaki, Japan

Takuya Ohwa
takuyaohwa@fujitsu.com
Fujitsu Laboratories Ltd.
Kawasaki, Japan

Kotaro Ohori
ohori.kotaro@fujitsu.com
Fujitsu Laboratories Ltd.
Kawasaki, Japan

## ABSTRACT

In the field of recommendation, there have been many efforts to help users interact with recommender systems in ways that appropriately elucidate user preferences. To let users interact with recommender systems, it is desirable that recommender systems are as transparent as possible. However, it is difficult to achieve complete transparency even with a simple method for interactive recommender systems because the relationship between the item features and the user preference is not intuitive when there is a utility function to generate recommendations. We focus on multi-attribute utility theory (MAUT) as one of the simplest methods for recommender systems and clarify the difficulties with its usage in interactive environments. Then, to overcome the difficulties, we propose an algorithm to generate natural language hints to let users understand ways of operation and see more items that match their preferences. The results of an offline simulation demonstrate that our method can effectively recommend a diverse range of items to users. As future work, we will conduct empirical experiments to evaluate the performance of our method in online situations.

## CCS CONCEPTS

• **Information systems** → **Recommender systems**.

## KEYWORDS

interactive recommender systems; additional information; hints; multi-attribute utility theory

**ACM Reference Format:**
Yuri Nakao, Takuya Ohwa, and Kotaro Ohori. 2019. Generation of Hints to Overcome Difficulty in Operating Interactive Recommender Systems. In *IntRS '19: Joint Workshop on Interfaces and Human Decision Making for Recommender Systems, September 19, 2019, Copenhagen, Denmark*, 9 pages.

## 1 INTRODUCTION

Interactive recommender systems have been developed to elucidate user preferences [5, 6, 8]. To grasp the real preferences of the user, recommendations that cannot be changed by the user are sometimes inappropriate. This is because, while the user models are generated on the basis of the historical record of user behavior, the models are not one-size-fit-all and sometimes do not match new users. Therefore, users need to convey their preferences to the recommender systems through some form of interaction. To do this, the ideal interactions between recommender systems and users should be achieved in the situation where users understand how their operations are recognized by the systems and how they are represented as the preference.

To achieve this ideal interaction between recommender systems and users, the systems should be transparent in that users can understand the relationship between item feature space and user preference space. This is difficult, however, due to the relationship between the two spaces with utility functions, and according to previous research, users do not tend to obtain a detailed enough understanding of the intelligent system without a technical explanation [11]. Despite this knowledge, the system needs to be as simple as possible to make the process of recommendation transparent for users. Systems that are incomprehensible to users on the technical side, or that can not form deep mental models, are not transparent.

In this paper, we propose a new algorithm to help users understand the specific characteristics of the relationship between preference space and item space with one of the simplest utility functions and conduct an offline experiment. As a simple method to relate item space and preference space, we utilize Multi-Attribute Utility Theory (MAUT) [9], which is a general method for supporting human decision making using linear utility functions [4, 17, 19, 20]. While MAUT is a simple method, there are difficulties for users in operating MAUT-based interactive recommender systems because of the unintuitive characteristics of the interactions between the users and the systems. To overcome this difficulty, we develop an algorithm to generate natural language hints to provide users with information about the item feature space in the direction where the user preference moves in. The ultimate purpose of this algorithm is to let users obtain more detailed understandings of the algorithmic behaviors by seeing more diverse items. In this work, as a first step, we conduct an offline experiment and investigate the potential effect on the simulated behavior of users. The research question for our offline experiment is whether users can see more diverse items with the existence of hints generated with our method. We set up this research question because users need to observe as diverse items as possible to recognize the relationship between item space and feature space.

The remaining of this paper is organized as follows: In Section 2, we provide a brief overview of previous works related to interactive recommender systems and MAUT. We describe the difficulties in operating MAUT-based interactive recommender systems and propose our algorithm in Section 3. In Section 4, we explain the detailed settings of our experiment. In Section 5, we report the results of our experiment. Additionally, in Section 6, we discuss the results in more detail and outline future work. Finally, the conclusion of the paper shown in Section 7. Our contribution to the community is two-fold: first, we point out the critical difficulties in one of the

simplest utility functions, MAUT, when it is used in interactive recommender systems, and second, we offer an algorithm to generate natural language hints to overcome these difficulties.

## 2 RELATED WORK AND BACKGROUND

There have been many efforts to develop interactive recommender systems [6] for recommending movies [5, 18], music [2, 8], restaurants [16], researchers [21], and so on. In the previous works, the transparency of the system is evaluated as the subjective estimation of transparency or satisfaction through the evaluation methodology [10, 15]. It is appropriate to evaluate not the detailed understanding of the functions of systems but rather to investigate subjective impressions when evaluating the transparency of systems because, according to research on the formation of the mental models of a recommender system [11], people tend to have difficulty in forming deep mental models without any technical explanations. However, despite the negative results of the previous work, it is necessary to have users understand the characteristics of the method used in recommender systems in as much detail as possible in order to achieve complete transparency. In this paper, we offer a new algorithm that helps users understand the critical difficulties in one of the simplest recommendation methods, MAUT. It is necessary for users to see more diverse items related to their preferences to understand the characteristics of the recommendation method.

Relating to MAUT, there are many studies that utilize this method to support human decision making [4, 17, 19, 20, 23]. Moreover, MAUT is well known as a traditional decision support technology used in the early steps of decision makings [4], and at the same time, is versatile technology that is a mathematical analogy to matrix factorization [22]. While user preferences for items are calculated based on uninterpretable latent features of user preferences and item features with matrix factorization, with MAUT, the features are interpretable [22]. Additionally, some interactive recommender systems already utilize it as the method to generate recommendations [7, 23]. In Zhang et al. [23], MAUT was used with a critique-based interactive recommender system [13]. In their system, the weights for each feature of items were calculated based on the items chosen by users, and the resulting recommendations became more accurate. However, the detailed functions are not shown to the users, so the system is still not fully transparent. In this paper, we provide an algorithm that generates hints about the operation in an attempt to make the function fully transparent. Additionally, by evaluating the performance of our method as diversity and novelty, we investigate the effects of our method of letting users observe a wider range of items.

## 3 METHOD

In this section, first, we explain the functions of the MAUT-based interactive recommender system in detail. Then, we describe the difficulty experienced by users and present the method we developed to overcome the difficulty.

### 3.1 Characteristics of MAUT

*3.1.1 MAUT-based Interactive Recommendation.* First, we briefly explain the characteristics of our MAUT-based interactive recommender system. As a method of recommendation, we utilize MAUT to recommend top-N items (top-10 items in practice). To rank items, the personalized utility ($Vu(i)$) of each item ($i$) for each user ($u$) is calculated on the basis of MAUT, as follows:

$$V_u(i) = \sum_f \beta_f^{(u)} x_f(i) \tag{1}$$

where a feature for an item is denoted as $f$ and the normalized value for each feature is denoted as $x_f(i)$. In this paper, we use the word 'feature' in the same meaning of 'attribute' of MAUT. With this function, we calculate a user's preference as the values of coefficient ($\beta_f(u)$), which show to what extent the feature $f$ is considered an important one by the user $u$. An item $i$ is denoted as K-dimensional feature vector $x(i) : x_1(i), ..., x_K(i)$.

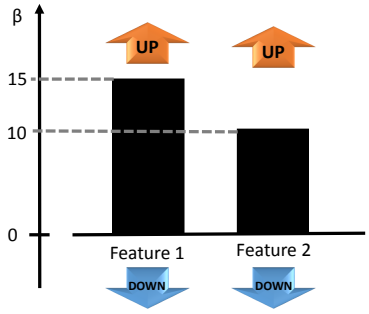


**Figure 1: Conceptual model of developed interface. The bar graphs indicating the preference for each feature are displayed. Users click above and below the bar graphs and move their own preferences. Users can move values of preferences for multiple features at a time. The values for each feature correspond to the user preference in Figure 2**

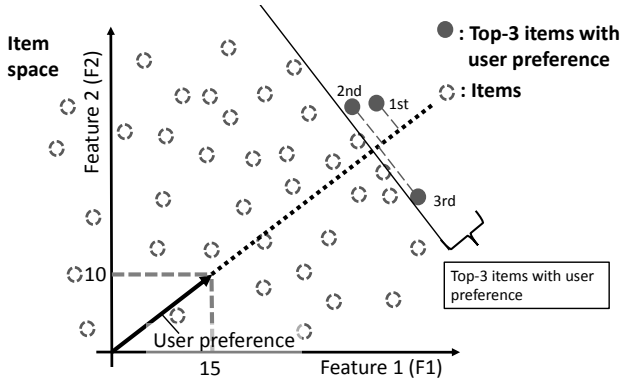


**Figure 2: A simple visualization of the functions of MAUT. The score of an item for a user is calculated as the distance between the origin and the foot of a perpendicular line drawn from the item to the user preference vector.**

We developed an interactive recommender system to collect real user data by using this method. In our system, the users' preferences are displayed as bar graphs that indicate the value of $\beta$. The concept of our interface is shown in Figure 1, where there are two bars

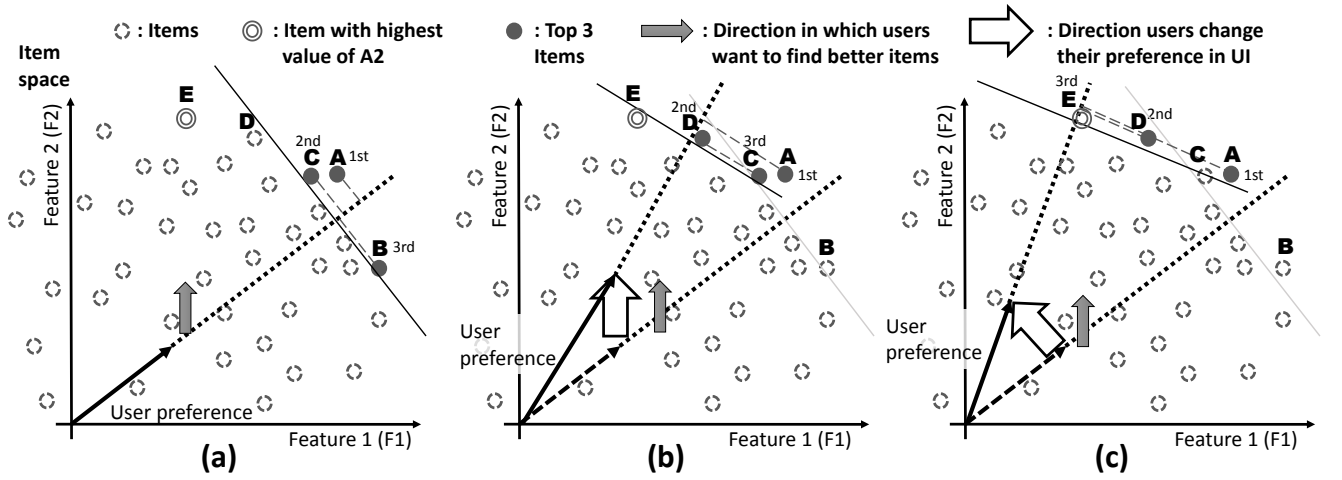


**Figure 3: Characteristics of user movement. (a) First, items A, B, and C are displayed for the user who has a certain preference. This user wants to find an item whose F2 score is as high as possible. The item with the highest F2 score is E, so item E should be displayed for the user ideally. (b) Following her/his inclination, the user increases the preference of F2. However, item E is not recommended in the top-3 list. (c) Due to the characteristics of MAUT, to find item E, the user should increase the preference of F2, and at the same time, decrease the preference of F1.**

representing the user preference ($\beta$) for each feature of items. The values in Figure 1 correspond to the states of user preference in Figure 2. This user has a value of 15 for the preference of *Feature 1* (F1) and a value of 10 for the preference of *Feature 2* (F2). In our system, users can see the bar graphs as well as the recommended item list and change their preferences with a fixed length by clicking the buttons above and below the graphs. Moreover, they can change preferences for multiple features at a time. In the following, the number of times a user changes her/his preference (i.e., the number of interactions) is denoted as $t$.

*3.1.2 Difficulties in MAUT.* Users are confronted with difficulty when MAUT is used in interactive recommender systems. With MAUT, the scores of the items are calculated as the inner products of user preference and the position vectors of items in the item feature space. This means that the ranking of the items is the same as the order of the perpendicular feet drawn from the items to the user preference vector, as shown in Figure 2. This leads to unintuitive behavior of the recommender system and to difficulties in operating preference for the user. For example, as we show in Figure 3, when there are two features and the top-3 item list is displayed to a user, if the user wants to see items with a higher value of *Feature 2*, s/he naturally increases the preference for *Feature 2*. Ideally, s/he will see the item with the highest value for *Feature 2* (item **E**) by increasing the preference for *Feature 2* just once. However, it is difficult to see item **E** by increasing only feature 2, as shown in Figure 3(b). To see item **E**, the user has to decrease the preference for feature 1 while simultaneously increasing that for *Feature 2* (Fig. 3(c)). This implicates that it is necessary for users to change their preferences in unintuitive manners.

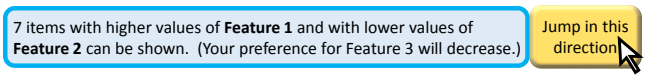


**Figure 4: The assumed natural language hint. Users can see new items matching their preferences with clicking 'jump in this direction' button.**

## 3.2 Method to Generate Hints

To overcome the difficulties in operating the MAUT-based interactive recommender system described above, we propose a technology that generates hints to achieve more effective operation for users. The hints will be shown in the form of natural language[1]. As indicated in Figure 4, the hint suggests a direction users can take to see more diverse items and users can move toward this direction by clicking the 'jump in this direction' button. Therefore, users can decide to use or not to use hints whenever the hints are displayed. Our algorithm calculates the appropriate direction to be suggested to users. The appropriate direction is determined on the basis of the tendency of the user's movement. First, our method calculates preferences that make as many items Top-1 as possible. With MAUT, items that can be top-1 are vertices of the convex hull of items. Therefore, we calculate every preference that makes each vertex of the convex hull of items Top-1 respectively. We call these preferences '**top-1 vectors**'. After that, we calculate which direction a user wants to move toward and which features s/he does not take into account. We call the direction in which the user moves '**favored direction**' or more simply, '**direction**', and the features that are not taken into account by users '**ignored features**'. Then,

[1]While we do not show the natural language hints to real users in our offline simulation, we plan to conduct an online experiment by showing natural language hints as future work.

we suggest the appropriate operation that enables the user to see items that have values matching the user's tendency to move based on the top-1 vectors. Additionally, we offer the user a way of operation that lets him/her see more items by adjusting the features that have been ignored until then. The details of this algorithm are described in the following subsection.

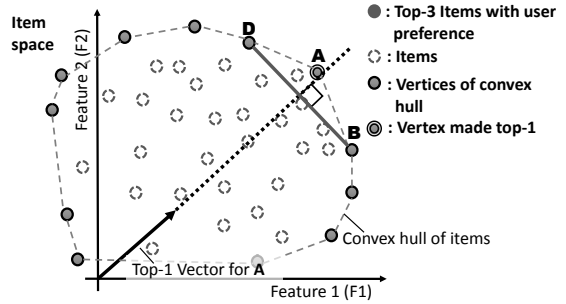


**Figure 5: Top-1 Vector for A. The vector is orthogonal to the line passing through the vertices B and D that are next to A.**

*3.2.1 Calculation of Top-1 Vectors.* To generate hints, we utilize the characteristics of MAUT that only vertices of the convex hull can be top-1 and calculate preferences making each vertex top-1. One of the preference vectors that makes a vertex top-1 (top-1 convex) is orthogonal to the plane (or hyper-plane) passing through vertices that are next to the top-1 vertex. This is because only vertices can be top-1 and rankings of items are decided as the order of the perpendicular feet drawn from items; therefore, when making a vertex top-1, it is enough that the perpendicular feet drawn from vertices that are next to the top-1 vertex are the same order. A 2-dimensional version is shown in Figure 5, where the top-1 vector for a vertex, **A**, is drawn. The vector is orthogonal to the line passing through the vertices **B** and **D**, which are next to **A**.

To generate top-1 vectors, we calculate the hyper-plane passing through vertices that are next to a vertex that is made top-1 with the vector. We can calculate this hyper-plane as the support vector of the hard margin support vector classification for two classes: the class of the top-1 vertex and the class of the other vertices [1]. Let $\mathbf{a_I}$ denote the position vector of item $I$ and let $\mathbf{z}$ be a vector that is orthogonal to the support vector. When $b$ denotes a bias term, the labels are $y_i = 1$ for the top-1 vertex and $y_i = -1$ for the class of the other vertices, and there are $M$ vertices. We calculate the hyper-plane as follows:

$$\begin{aligned} &\text{minimize } ||\mathbf{z}||^2 \\ &\text{subject to } y_i(\mathbf{z}^T\mathbf{a_i} + b) \geq 1 \text{ for } i = 1, ..., M. \end{aligned} \tag{2}$$

$\mathbf{z}^T\mathbf{a_i} + b = 0$ is the equation of the hyper-plane that is parallel to the hyper-plane passing through next vertices to top-1 vertex and $\mathbf{z}$ is a perpendicular line for the hyper-plane. Therefore, we obtain the top-1 vector as $\mathbf{z}$.

*3.2.2 Tendency of Users' Behavior.* Next, we calculate which directions a user should move toward. A direction is defined as the combination of features and the movement of increasing or decreasing of the value for the features. We name the value that represents to what extent a user wants to make a value of a feature up or down *Tendency*. To identify the direction favored by the user, we calculate a *Score* to generate the values of *Tendency*. When calculating the *Score*, we consider the more recent movement of the user as more important. First, we set the weight $w_t = w^t$, which indicates the importance of the movement of $t$th interaction as the number that increases in proportion to the number of interactions ($t$) between the recommender system and the user. This means that if $w_1 = w = 1.2$, then $w_2 = w * w = 1.44$, that indicates the importance of the movement of second interaction. The vector consisting of all weights is denoted as $\mathbf{w_t}$ and the history of choice for up (+) or down (−) for a certain feature value is denoted as $\mathbf{h_{f,\pm}}$. Here, we define the *Score* as

$$Score_{f,\pm} = \mathbf{h_{f,\pm}} \cdot \mathbf{w_t}. \tag{3}$$

For example, when a user change the preference for *Feature 1* in the three interactions like $UP \rightarrow DOWN \rightarrow UP$, $h_{1,+} = (1, 0, 1)$ and $h_{1,-} = (0, 1, 0)$. When $w = 1.2$, $\mathbf{w_3} = (1.2, 1.44, 1.73)$. Therefore, $Score_{1,+} = (1, 0, 1) \cdot (1.2, 1.44, 1.73) = 2.93$ and $Score_{1,-} = (0, 1, 0) \cdot (1.2, 1.44, 1.73) = 1.44$.

Then, we define the *Tendency* as

$$Tendency_{f,\pm} = \frac{Score_{f,\pm}}{\sum_t w_t}. \tag{4}$$

When consider the example we explained above, $Tendency_{1,+} = Score_{1,+}/\sum_t w_t = 2.93/4.37 = 0.67$, and similarly, $Tendency_{1,-} = 0.33$. We infer the direction in which the user wants to move by judging whether the *Tendency* surpasses the upper threshold, $UT$. For example, if $Tendency_{1,+} > UT$, we assume the user wants to increase the value of *Feature 1*. Additionally, the features ignored by the user can be known by judging whether the *Tendency* of a feature for both up and down is lower than the lower threshold, $LT$. For example, if $Tendency_{2,+} < LT$ at the same time $Tendency_{2,-} < LT$, *Feature 2* is ignored by the user.

*3.2.3 Combinations of Directions.* After inferring the direction toward which the user wants to move and the features ignored by her/him, we generate the combination of the directions favored and ignored by the user. Initially, we set the number of hints to display and the number of directions to consider. Then, we calculate the sum of *Tendencie*s for the combinations of directions that a user moves toward. For example, consider a case where there are three directions users tend to move toward, (*Feature 1, +, Tendency = 0.6*), (*Feature 2, -, Tendency = 0.7*), (*Feature 3, +, Tendency = 0.55*). When we take two directions into account, we obtain three combinations of directions and the sum of *Tendencie*s, i.e., (*Feature 1, +*, and *Feature 2, -*), (*Feature 1, +*, and *Feature 3, +*), and (*Feature 2, -* and *Feature 3, +*). For example, in terms of the sum of *Tendency*, the top-1 in the combinations is (*Feature 1, +*, and *Feature 2, -*). The sum of *Tendency* of this combination is 1.3. Now, we obtain the list of combinations of directions.

If there are any features ignored by the user, we add the direction related to the ignored features to the list. For example, when *Feature 4* is ignored, (*Feature 4, +*) and (*Feature 4, -*) are added to the list. Therefore, the top-1 in the list of combinations is (*Feature 1, +*, *Feature 2, -*, and *Feature 4, +*) or (*Feature 1, +*, *Feature 2, -*, and *Feature 4, -*). The plus or minus of *Feature 4* is decided on the basis of how many items will be changed by the hint. If the number of items that

**Algorithm 1:** Calculate Candidate Vector

**input :** $d$: Combination of directions, $I$: Vertices, $SI$: Shown items, $P$: User preference
**output :** Candidate vector

```
GenerateCandidateVector(d, I, SI)
    MaxItem ← i most matching d in SI;
    foreach j in I do
        if j more matching d than MaxItem then
            Add j to CandidateItem
    Dif ← 0;
    foreach k in CandidateItem do
        V ← Top-1 Vector for k;
        NR ← top-N ranking on the basis of V;
        l ← No. of different items in NR from SI;
        if Dif < l then
            CandidateVectors = ∅;
            Add V to CandidateVectors;
            Dif ← l
        else if Dif = l then
            Add V to CandidateVectors
    if No. of CandidateVectors > 1 then
        foreach V in CandidateVectors do
            C ← cos sim of V and P;
            if C = Max cos sim of V and P then
                Candidate Vector ← V
    return Candidate Vector
```

appear in the new recommendation list when the combination with *Feature 4, +* is more than that with *Feature 4, -*, the combination with *Feature 4, +* is applied.

*3.2.4 Calculation of Candidate Vector.* On the basis of the listed combinations of directions, we calculate the candidate vector to generate hints by using algorithm 1. First, the values of features in all vertices ($I$) are compared with the values of displayed items ($SI$) to judge whether there are items that have higher (or lower) values suited to the listed combinations of directions. Each item in $SI$ is represented as $i$. If there are vertices that match the combination of directions ($d$), they are added to the list of candidate items. For example, when the combination of directions toward which the user wants to move is (*Feature 1, +*, and *Feature 2, -*), if there are vertices whose values of *Feature 1* is higher than the highest value for *Feature 1* in the displayed items, and at the same time, if the values of *Feature 2* of the vertices are lower than the lowest value for *Feature 2* in the displayed items, the vertices are added to the list of candidate items.

Next, for each candidate item, the top-1 vector for the item is selected and the top-N item list is calculated on the basis of the top-1 vectors. At the same time, the number of different items in the top-N list that is generated on the basis of the top-1 vector from the currently displayed top-N list is calculated. Then, the top-1 vector that generates the top-N list in which the number of different items from the currently displayed items is the biggest is chosen as the candidate vector. If there remain multiple candidate vectors at this time, we select the one candidate vector that has the biggest cosine similarity (*cos sim*) with the current user preference ($P$).

*3.2.5 Generating Hints.* Finally, we display hints for users with natural language. Although multiple hints can be displayed, the number of hints should be limited to avoid information overload. When multiple hints are displayed, the hints generated based on top-N combinations in terms of the sum of $Tendency$ are displayed. To display hints in the form of natural language, we generate hint vectors (**HV**) from the chosen candidate vectors (**CV**) as

$$\mathbf{HV} = \frac{\mathbf{CV}}{||\mathbf{CV}||} * ||\mathbf{P}||, \tag{5}$$

where the user preference is denoted as **P**. Each hint vector is the unit vector of the candidate vector multiplied by the norm of the current user preference. This calculation is executed because the norm of candidate vectors is not fixed at this time and $Tendency$ is calculated on the basis of the difference between current preference and the hint vector, so the norm of the vector and the preference should be aligned.

Then, natural language hints are generated in two different forms. The first one is hints without ignored features. In this form, we focus on making the user preference move to the hint vector. For the hint, we display a natural language hint and offer a 'Jump in this direction' button to make the current preference jump to the hint vector as shown in Figure 4. This means that, when the user clicks the 'Jump in this direction' button, displayed user preference changes to the values of the selected hint vector and the displayed list of items changes at the same time. Additionally, we display the information on features that are not included in the combinations of directions if the difference of values for the feature between the current user preference and the hint vector is large enough. For example, when the chosen combination is (*Feature 1, +* and *Feature 2, -*), seven items will be newly shown with the selected hint vector, and at the same time, the difference of *Feature 3* between current user preference and the hint vector is large enough, the hint is shown as "*7 items with higher values of Feature 1 and with lower values of Feature 2. (You need to decrease the preference for Feature 3.)*" The second form is hints with ignored features. In this form, while we generate the same natural language hint as the first one, the user can move toward the suggested direction with the same value by clicking the 'Jump in this direction' button, as users can move with the normal operation. By "normal operation," we mean the operation with the up or down button in the interface displayed regardless of the existence of hints. This is because the hint vectors considering the ignored features can change the user preference in radically different directions from the direction in which the user wants to move, and so the movement toward the hint vectors should be gradual.

## 4 EXPERIMENT

To conduct the offline experiment with our method, we have to define the simulation environment. In this section, we explain the dataset we use, the settings of the parameters, the policy of the simulated user behaviors, and the metrics to evaluate the performance.

### 4.1 Dataset

To conduct the offline experiment, we use the data gathered from our MAUT-based interactive recommender system to let users find their favorite areas to relocate to. There are two reasons why this application domain is chosen. First is that there is a trade-off relation between convenience and safety in this domain. This trade-off is needed because it makes users search for their appropriate preference between conflicting characteristics. Otherwise, users will just make the preference for each attribute maximum or minimum. Second is that users interact with recommender systems more seriously when making important decisions related to their own lives.

When using our recommender system, users input their demographic characteristics to the system and are recommended areas that match them. After that, users can adjust their preferences and approach their final preferences. The recommender system recommends blocks in a city (areas from one street to the next street) as items that are characterized by six features:

i). Transportation: the number of trains arriving at the station in the area.
ii). Shopping: the scale of the shopping facilities.
iii). School proximity: the time it takes to walk to the nearest elementary school.
iv). Neighborhood activity: the attendance ratio to events of neighborhood associations.
v). Hospital: the number of hospitals.
vi). Safety: crime rate.

We normalize these features as 0 for the minimum number and 1 for the maximum number when calculating users' utility for each area. The number of 0 is set to show that there are not any positive or negative values related to the feature. With our system, 163 blocks can be recommended, and 83 blocks are vertices. Therefore, we generated 83 top-1 hint vectors. While the system was open publicly on the Internet, we use the data of users who came into our office to use the system because the motivation to use the system should be aligned. The users are interested in relocating to the same or similar cities that the system recommends. They are asked to look at the recommended areas and mark the ones they think they might like to move to as favorite areas. We used the data from 49 users who marked at least one favorite area. Users who could not find any favorite areas or who did not use the system seriously are not included in the data. We evaluated the seriousness of users with a 5-scale questionnaire of impressions about the system after using it. If a user checked the same point for all questions or the answers were not coherent, we excluded the user from the data set.

### 4.2 Parameter Settings

We set the parameters for our experiment as follows. First, the number of features ($f$ in Equation 1) is six. Therefore, we conduct our experiment in the six-dimensional feature space. We consider the situation where the top-10 ranking of recommended items is shown to users. This is the same number used in our collection of the data set. Next, we set the value for the weight ($w$) when calculating the *Score* and *Tendency* of the favored direction as 1.2 ($= w = w_1$ in Equation 3 and 4). We set the upper threshold $UT$ to judge the direction a user wants to move toward as 0.5 and the lower threshold $LT$ to judge the ignored features as 0.3. Then, we set the number of displayed hints to 1. In our experiment, this means that when the hints are displayed, users always choose the hint at the top of the list. Additionally, when we generate the favored direction, we take three features into account. This is because we want to display hints with natural language, and more than three hints would be too many to take in at a glance.

### 4.3 Simulation Method

To conduct the offline experiment, we need a policy for the simulation. The offline simulation of interactive recommendation methods is considered difficult because the reactions of real users cannot be known. While simulation methods are investigated for critique-based recommender systems [14], there are not any detailed investigations of simulation methods for MAUT-based interactive recommender systems. In our experiment, we assume that the final preference of each user (i.e., the preference when s/he quits the system) is the settled one. Therefore, in our scenario, the users' preferences move toward the final preference[2]. When the preference comes near enough to the final preference, the movement of preference stops. As real users can, simulated users can change their preferences for multiple features at a time with a fixed length.

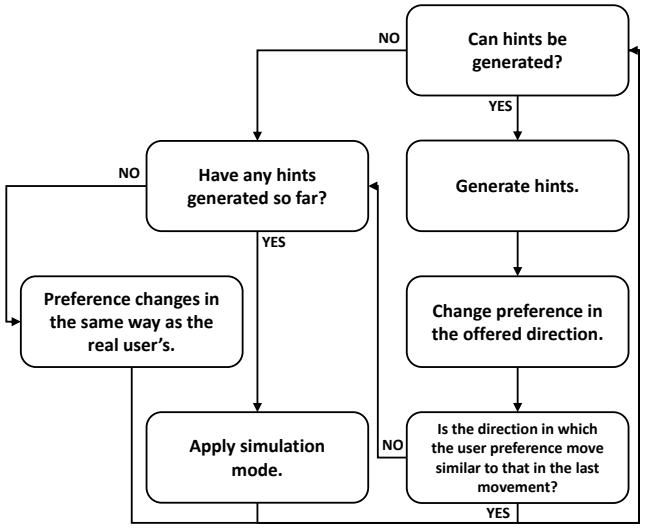


**Figure 6: The flow of our simulation.**

The detailed flow of our simulation is as follows. We summarize the flow in Figure 6. In our scenario, there are three ways of movement. The first is the movement that is the same as a real user's movement. When hints have not been displayed yet, the movements of preference follow the log data of real users' movement. The second is movement when the hints are displayed. The simulated users always click the 'jump in this direction' of the hints at the top of the displayed hints button whenever the hints are displayed in our scenario. If the number of items that are different from previous recommended items is less than four, we start to

[2] We use the deterministic approach, not stochastic approaches mainly because we do not know how often users keep directions suggested by hints and how the final preferences changes with the existence of hints.

generate hints. If we generate the hints to recommend items that match a user's preference successfully, we make the preference of the user jump to the hint vector. If there is a hint vector based on the ignored features, the user preference moves toward the hint vector generated on the basis of the ignored feature regardless of the existence of the hint vectors generated on the basis of the favored directions.

As the third type of movement, we set the simulation mode. If the user's *Tendency* changes radically resulting from the jump to hint vectors, it is not appropriate to continue to generate hints. Therefore, if the sign of favored direction for the same feature becomes opposite to the previous tendency resulting from the application of hint, we stop generating hint vectors and start to apply the simulation mode. In this simulation mode, the preference is made to move straight toward the final preference. We calculate the distance between the current preference and the final preference, and if the distance is more than that the user can move at one time, we make the preference move toward the final preference. If the distance from the current value and the final value is more than the distance a user can move at one time, we select the top-3 feature whose values are distant from those of final preferences more than the distance that a user can move at one time, and after that, make the preference moves toward the final preference at one time. In the simulation mode, hints can be generated.

### 4.4 Evaluation Metrics

We evaluate the performance of our method with several metrics. First, we identify the percentage of users provided with hints. In our method, a user needs to move her/his preference in the same direction to some extent to be provided with hints. Naturally, there are some users who do not have any specific direction they tend to move in. Therefore, by dividing users into those who are given hints and those who are not, we investigate the possibility of the provision of hints. We count a user as one who is shown hints if s/he is given hints at least just once. Then, we investigate the performance of our method only for the users provided with hints.

Next, for the users provided with hints, we use a dependent sample t-test to compare the number of interactions needed to move close to the final preference and the number of favorite items shown for each user. The number of interactions is evaluated to confirm the behaviors of the simulated users is not far from that of original users. With our method and simulation policy, it is possible for user preferences to approach final preferences in a different way from the original one. By evaluating the number of interactions, we can analyze the effects of our method on the simulated user behavior. Additionally, the number of favorite items is investigated to evaluate to what extent users become unable to see their favorite items with the existence of hints. When our method is applied, the user preference sometimes changes radically and skip the items that are marked as favorite items originally. Therefore, by investigating the favorite items, we can evaluate whether or not users overlook their favorite items in our simulated environment.

Finally, we investigate the diversity and novelty of the recommended items. Our research question is whether users can see more diverse items when there are hints generated on the basis of our method. Therefore, the diversity and the novelty are our main results. As the metric to evaluate the diversity and novelty achieved with our method, we apply temporal diversity [12], which is the diversity in the sequence of recommendation lists our method produces over time. While there are several ways of evaluating diversity [3], we utilize this method because our intention is to recommend items for users sequentially in the course of interactions. In this method, we calculate diversity between two recommendation lists, $L_1$ and $L_2$, whose lengths are $N$ when $L_2 \backslash L_1 = \{x \in L_2 | x \notin L_1\}$, with following equation:

$$diversity(L_1, L_2, N) = \frac{|L_2 \backslash L_1|}{N}. \quad (6)$$

This equation can only calculate the diversity between two lists. Therefore, we calculate *novelty* to compare new recommendations to the set of all items that have been recommended ($A_t$) on the basis of [12] as follows:

$$novelty(L_1, N) = \frac{|L_1 \backslash A_t|}{N}. \quad (7)$$

If both the *diversity* and *novelty* increase significantly with the existence of the hints, it is clarified that our method has positive effects on the recommendation. We calculate the average of *diversity* and *novelty* for the number of users' interaction (i.e. how many times a user renewed the displayed recommendations) as *MD* and *MN* respectively.

## 5 RESULTS

In this section, we report our results. We initially explain the ratio of simulated users who are provided with hints. After that, the number of interactions and shown favorite items are analyzed. Finally, the diversity and the novelty of the recommendation is described.

### 5.1 Percentage of Users Provided with Hints

First, we show the number of users who are provided with hints. Data from 49 users are used. A total of 20 users (40.8%) are provided with hints generated on the basis of users' favored direction, and only one user (2.0%) is provided with hints generated on the basis of ignored features. This latter user is included in the users who are shown hints based on the favored directions.

### 5.2 The Number of Interactions and Displayed Favorite Items

We check the results of the number of interactions and of shown favorite items to analyze the effects of our method to the user behaviors. The results are summarized in Table 1. The number of users analyzed ($N$) is 20 because we only consider the users provided with hints. Between the original data and the simulated data (with the existence of hints), while the number of interactions

**Table 1: The number of interactions and of shown favorite items.**

| | | Original | | With Hint | | | | |
|---|---|---|---|---|---|---|---|---|
| | N | M | SD | M | SD | t | df | p |
| Interactions | 20 | 14.9 | 2.36 | 11.9 | 2.01 | -.428 | 19 | .67 |
| Favorites | 20 | 3.67 | 28.5 | 3.47 | 9.10 | -1.87 | 19 | .067 |

does not decrease significantly ($p = 0.67$), the number of favorite items shown to users decreases with the existence of hints, the p-value is more than but close to 0.05 ($p = 0.067$).

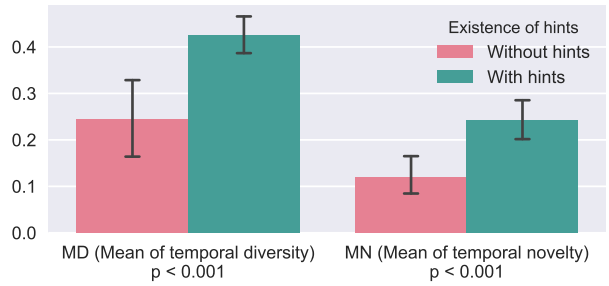


**Figure 7: Comparison of $MD$ and that of $MN$. The error bars shows standard errors.**

### 5.3 Diversity and Novelty

Finally, we compare the average of the mean of the temporal diversity for the times of interactions ($MD$) and that of the temporal novelty ($MN$) per user. In Figure 7, we show the average of $MD$ and of $MN$ for the number of data (users) with bar graphs. We obtain the results that both $MD$ and $MN$ increase significantly with the existence of hints from when there are not any hints. The representative values are as follows: For $MD$, $M = 0.245, SD = 0.190$ when there are not any hints, and $M = 0.425, SD = 0.100$ with the existence of hints. With the t-test of $MD$, $t = 7.07$, $df = 19$, and $p = 1.00 * 10^{-06}$. For $MN$, $M = 0.120$, $SD = 0.098$ when there are not any hints, and $M = 0.242$, $SD = 0.096$ with the existence of hints. With the t-test of $MD$, $t = 6.52, df = 19$, and $p = 3.03 * 10^{-06}$. These results implicate that the generation of hints results in the effective recommendation of new items that a user has not seen before in our experiment.

## 6 DISCUSSION AND FUTURE WORK

We discuss the results and their implications in this section. Related to the ratio of users provided with hints, more than half of the simulated users are not provided with the hints. This implies that more than half the users do not have a definite tendency of movement. One possible explanation for this is that users search items at the same time as they change their preference, so sometimes there is not a strong tendency in the behavior. On the other hand, this means that it is possible that we can not provide hints more than half of the users when we conduct an online study. This implicates that we need to set some devices to make users have specific tendencies as fast as possible with intelligent user interfaces.

With the change in the number of interactions, it is indicated that our method does not have any positive or negative effects on the simulated users. While it is possible that the number of interaction decreases because the user preferences can jump to other values radically with our method, there is no significant effect on the number of interaction with our simulation policy. For the number of shown favorite items, when our method is applied, users sometimes can not find items that they originally marked as favorite items. Skipping the favorite items can generate the missing of opportunities for both users and the item providers, so it is necessary to display skipped items in some ways. However, this may occur the problem of information overload and devices to overcome this problem is needed. Finally, the results of the average of temporal diversity and of temporal novelty indicate that our method can display more items for each user. This means that, as the answer to our research question, our method can provide more diverse range items for users than when there are not any hints.

However, we cannot confirm these implications are valid when we conduct an online experiment due to the limitations of our simulation policy. In our experiment, we adopt a simulation policy that assumes the final preference of each user that collected in the situation where there are not any hints does not change regardless of the existence of hints. However, in reality, the final preference of the same users can change when there are hints because different items are recommended to users and the cognitive load will change as a result of this additional information. It is possible that these differences will lead to negative effects, such as users exit the system earlier or difficulties in identifying a clear tendency of user movement. There may also be other effects from the hints, such as favorite items changing or the final preference changing.

Therefore, as future work, we need to conduct an online user experiment by designing real user interfaces. In order to minimize the negative effects, we should design the parameters for the hints on the basis of user observations while, in our simulation, we set parameters without user observations using real user interfaces. Additionally, effective user interfaces have to be developed. In addition to the assumed effect of the existence of hints described above, we will investigate whether displayed hints are actually chosen by users. In the experiment reported in this paper, we assumed that users always utilize the hints whenever they are displayed. However, in a real situation, users might ignore the hints for several reasons, e.g., if the calculated tendency does not show the user's preferred directions or if the hints are not highlighted enough. At the same time, the impact of hints on the user experience of the recommender systems should be investigated in the online experiment.

## 7 CONCLUSION

In this paper, to achieve transparent interactive recommender systems, we offered an algorithm to generate natural language hints to let users know the appropriate way of movement that matches users' preferences. We used MAUT as one of the simplest methods to relate user preference to the item features and clarified the difficulties in operating MAUT-based interactive recommender systems. With an offline experiment, it is found that our algorithm achieves more diverse recommendations to the users. This will leads to the more transparent interactive recommender systems in that users can understand the detailed algorithmic behaviors of the systems. As future work, we will conduct online experiments to evaluate the performance of our algorithm in the real interactive environment.

## REFERENCES

[1] Shigeo Abe. 2012. *Support Vector Machines for Pattern Classification* (2nd ed.). Springer Publishing Company, Incorporated.

[2] Svetlin Bostandjiev, John O'Donovan, and Tobias Höllerer. 2012. TasteWeights: A Visual Interactive Hybrid Recommender System. In *Proceedings of the Sixth ACM Conference on Recommender Systems (RecSys '12)*. ACM, New York, NY, USA, 35–42.

[3] Pablo Castells, Neil J. Hurley, and Saul Vargas. 2015. *Novelty and Diversity in Recommender Systems*. Springer US, Boston, MA, 881–918.

[4] Ward Edwards and Basolo Fasolo. 2001. Decision Technology. *Annual Review of Psychology* 52, 1 (2001), 581–606.

[5] Michael D. Ekstrand, Daniel Kluver, F. Maxwell Harper, and Joseph A. Konstan. 2015. Letting Users Choose Recommender Algorithms: An Experimental Study. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. ACM, New York, NY, USA, 11–18.

[6] Chen He, Denis Parra, and Katrien Verbert. 2016. Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. *Expert Systems with Applications* 56 (2016), 9 – 27.

[7] Dietmar Jannach and Gerold Kreutler. 2007. Rapid Development of Knowledge-based Conversational Recommender Applications with Advisor Suite. *J. Web Eng.* 6, 2 (June 2007), 165–192.

[8] Yucheng Jin, Nava Tintarev, and Katrien Verbert. 2018. Effects of Personal Characteristics on Music Recommender Systems with Different Levels of Controllability. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*. ACM, New York, NY, USA, 13–21.

[9] Ralph L. Keeney and Howard Raiffa. 1993. *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Cambridge University Press.

[10] Bart P. Knijnenburg, Martijn C. Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. 2012. Explaining the User Experience of Recommender Systems. *User Modeling and User-Adapted Interaction* 22, 4-5 (Oct. 2012), 441–504.

[11] Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. 2012. Tell Me More?: The Effects of Mental Model Soundness on Personalizing an Intelligent Agent. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1–10.

[12] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. 2010. Temporal Diversity in Recommender Systems. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*. ACM, New York, NY, USA, 210–217.

[13] Lorraine McGinty and James Reilly. 2011. *On the Evolution of Critiquing Recommenders*. Springer US, Boston, MA, 419–453.

[14] Quang Nhat Nguyen and Francesco Ricci. 2007. Replaying Live-user Interactions in the Off-line Evaluation of Critique-based Mobile Recommendations. In *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys '07)*. ACM, New York, NY, USA, 81–88.

[15] Pearl Pu, Li Chen, and Rong Hu. 2011. A User-centric Evaluation Framework for Recommender Systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. ACM, New York, NY, USA, 157–164.

[16] Francesco Ricci and Quang Nhat Nguyen. 2007. Acquiring and Revising Preferences in a Critique-Based Mobile Recommender System. *IEEE Intelligent Systems* 22, 3 (May 2007), 22–29.

[17] Amir Sanayei, S. Farid Mousavi, M. Reza Abdi, and Ali Mohaghar. 2008. An integrated group decision-making process for supplier selection and order allocation using multi-attribute utility theory and linear programming. *Journal of the Franklin Institute* 345, 7 (2008), 731 – 747.

[18] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. *Collaborative Filtering Recommender Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 291–324.

[19] Niranjan Suri, Giacomo Benincasa, Rita Lenzi, Mauro Tortonesi, Cesare Stefanelli, and Laurel Sadler. 2015. Exploring value-of-information-based approaches to support effective communications in tactical networks. *IEEE Communications Magazine* 53, 10 (October 2015), 39–45.

[20] George W. Torrance, Michael H. Boyle, and Sargent P. Horwood. 1982. Application of Multi-Attribute Utility Theory to Measure Social Preferences for Health States. *Operations Research* 30, 6 (1982), 1043–1069.

[21] Chun-Hua Tsai and Peter Brusilovsky. 2018. Beyond the Ranked List: User-Driven Exploration and Diversification of Social Recommendation. In *23rd International Conference on Intelligent User Interfaces (IUI '18)*. ACM, New York, NY, USA, 239–250.

[22] Martijn C. Willemsen, Mark P. Graus, and Bart P. Knijnenburg. 2016. Understanding the role of latent feature diversification on choice difficulty and satisfaction. *User Modeling and User-Adapted Interaction* 26, 4 (01 Oct 2016), 347–389.

[23] Jiyong Zhang and Pearl Pu. 2006. A Comparative Study of Compound Critique Generation in Conversational Recommender Systems. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, Vincent P. Wade, Helen Ashman, and Barry Smyth (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 234–243.