Navigation-by-Preference: A New Conversational Recommender with Preference-Based Feedback

Arpit Rana
Insight Centre for Data Analytics
arpit.rana@insight-centre.org

Derek Bridge
Insight Centre for Data Analytics
derek.bridge@insight-centre.org

ABSTRACT

We present Navigation-by-Preference, *n-by-p*, a new conversational recommender that uses what the literature calls preference-based feedback. Given a seed item, the recommender helps the user navigate through item space to find an item that aligns with her long-term preferences (revealed by her user profile) but also satisfies her ephemeral, short-term preferences (revealed by the feedback she gives during the dialog). Different from previous work on preference-based feedback, *n-by-p* does not assume structured item descriptions (such as sets of attribute-value pairs) but works instead in the case of unstructured item descriptions (such as sets of keywords or tags), thus extending preference-based feedback to new domains where structured item descriptions are not available. Different too is that it can be configured to ignore long-term preferences or to take them into account, to work only on positive feedback or to also use negative feedback, and to take previous rounds of feedback into account or to use just the most recent feedback.

We use an offline experiment with simulated users to compare 60 configurations of *n-by-p*. We find that a configuration that includes long-term preferences, that uses both positive and negative feedback, and that uses previous rounds of feedback is the one with highest hit-rate. It also obtains the best survey responses and lowest measures of effort in a trial with real users that we conducted with a web-based system. Notable too is that the user trial has a novel protocol for experimenting with short-term preferences.

CCS CONCEPTS

• **Information systems → Users and interactive retrieval**.

KEYWORDS

Conversational recommender system, Preference-based feedback, Short-term preferences, User trial

ACM Reference Format:
Arpit Rana and Derek Bridge. 2020. Navigation-by-Preference: A New Conversational Recommender with Preference-Based Feedback. In *25th International Conference on Intelligent User Interfaces (IUI '20), March 17–20, 2020, Cagliari, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3377325.3377496

1 INTRODUCTION

Recommender Systems help their users discover novel content ('items') in a personalized manner [20]. These systems infer the user's long-term preferences from a user profile, which records her past interactions (e.g. ratings, etc.) with the items. Recommendation algorithms often assume *single-shot recommendation*: the recommender ranks the candidate items and allows the user to explore the top-*n* items from this ranking. The problem comes when the user is not fully satisfied by the recommended items. Other things being equal, the recommender cannot offer her a fresh set of recommendations unless she changes her profile, e.g. by consuming and rating an item. In *conversational recommendation*, by contrast, the user is invited to provide feedback on the current top-*n* recommendations, even without consuming them; for example, the user might simply indicate which one of the top-*n* recommendations comes closest to the kind of item she wants to consume on this occasion. The recommender takes account of the feedback in generating a fresh top-*n* recommendations. It may take several recommendation cycles before the user finds a suitable item, which is why these systems are called conversational recommender systems and why the interaction is referred to as a dialog.

The conversational recommender terminology dates back ten years, e.g. [3, 33]. More recently, the word "conversational" has implied not just a dialog, as before, but a dialog in natural language, e.g. [12]. In this paper, we continue to use the word in its earlier, broader sense; indeed, the research we describe in this paper uses a graphical user interface and not a natural language interface.

Conversational recommender systems cater for a user who is not satisfied with the initial top-*n* recommendations. This is particularly useful where the user has an ephemeral goal, so she has short-term preferences that differ from her long-term preferences. For example, a user might usually watch documentaries but this evening she is not in the mood for something so serious. Or, perhaps, this evening she wants something to watch with her mother, so she should accommodate her mother's tastes as well as her own. Conversational recommendation is therefore one approach to context-aware [1] and context-driven [25] recommendation: the user can give feedback to steer the recommendations toward ones that best suit the context [15]. However, most context-aware recommender systems are single-shot systems: they confine their contextual factors to ones that are observable at the start of the session [15]. The advantage of conversational systems is that they handle the case where requirements (e.g. context, the user's mood, her ephemeral goals, etc.) are uncertain, or even erroneous, and may be refined by exposure to the items that the system presents [26].

In this paper, we present a new conversational recommender system, which we call Navigation-by-Preference (*n-by-p*). As we will explain in more detail, it works in a content-based way on

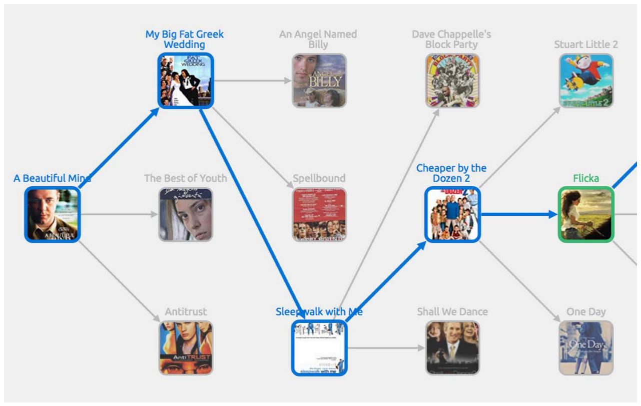


Figure 1: An example of navigation-by-preference, showing a *preference chain.*

unstructured item descriptions such as sets of keywords or tags. It uses what the literature calls *preference-based feedback* [35], in which a user simply selects from the current *n* recommendations the one that comes closest to the kind of item she wants to consume. It combines short-term preferences (from the feedback the user provides during the dialog) with long-term preferences (from the user profile). But it can be configured in a variety of ways including: how much it weights the long-term preferences; whether it takes account of feedback given only in the most recent cycle or feedback given throughout the dialog; whether it uses only the positive feedback revealed by the item that the user selects in a cycle or whether it also takes into account the negative feedback revealed by the items that the user does not select.

Figure 1 shows an example of an *n-by-p* conversation in the form of a tree of recommendations. In the example, the user provides a *seed* movie, in this case, *A Beautiful Mind.* The system recommends three movies (*My Big Fat Greek Wedding*, *The Best of Youth*, and *Antitrust*). The user indicates that, of these three, *My Big Fat Greek Wedding* comes closest to the kind of movie she wants to watch. From this feedback, in the next cycle, the system recommends a further three movies. This continues for several more cycles, forming a sequence of selected movies that we call a *preference chain*, which is highlighted in the diagram.

This paper makes the following contributions:

- The past work on preference-based feedback [35], and most past work on conversational recommender systems, assumes that items have structured descriptions, usually in the form of sets of attribute-value pairs. Recommending laptops (where attributes include price, screen dimensions and hard-drive size) and rental apartments (where attributes include number of bedrooms and distance to the supermarket) are examples of domains whose items have structured descriptions. *n-by-p* works on unstructured item descriptions, such as sets of keywords or tags. Hence, the paper extends preference-based feedback to domains where structured item descriptions are not available or less applicable. These domains include movies, music, art and news.
- Past work on preference-based feedback, and most past work on conversational recommender systems, takes into account only the most recent round of feedback: long-term preferences and feedback revealed in earlier cycles are ignored. *n-by-p* can be configured to also take into account long-term preferences, or feedback from earlier in the dialog, or both.
- Past work on preference-based feedback, and most past work on conversational recommender systems, has been evaluated only in offline experiments. In this paper, as well as offline experiments, we report the results of a user trial. Furthermore, our trial adopts a novel protocol for exploring short-term preferences in recommender system evaluation.

In Section 2, we describe the related work on conversational recommender systems; Section 3 presents *n-by-p* in detail. Section 4 reports the results of offline experiments, and Section 5 reports the results of a user trial.

2 RELATED WORK

The distinction between single-shot and conversational recommender systems is explained in, e.g., [3, 33]. Conversational recommendation is common in knowledge-based recommender systems [4], where the recommender reasons about which items best satisfy the user's goals and preferences. *n-by-p* is a form of navigation-by-proposing, where the conversational recommender elicits the user's short-term preferences by showing her items [32]. *n-by-p* uses what the conversational recommender literature refers to as *preference-based feedback* [35]: the user expresses preferences at the item level (rather than, for example at the level of features or attributes), indicating simply which of the current recommendations she prefers ("more like this"). This is attractive in domains where users might struggle to articulate their preferences in more detail [34]. It avoids issues about the accuracy and stability of explicit ratings [2] and aligns with evidence that users prefer to compare items rather than to rate them [18]. But if preference-based feedback is to result in efficient dialogs, account needs to be taken too of the ways in which the rejected items differ from the selected one, as is done in *comparison-based recommendation* [22]. Extensions to the work reveal, for example, the usefulness of controlling the diversity of the recommendations in each cycle of the dialog [23, 24, 35].

Preference-based feedback is not the only way to obtain a user's preferences during a dialog. A recommender can also use navigation-by-asking [32], in which the recommender asks questions about the preferred values of attributes, chosen for example in an heuristic manner [10, 31] or using a model obtained by deep reinforcement learning [37]; and in the context of navigation-by-proposing there is *critiquing*, where users propose 'tweaks' to attribute values that would improve a recommended item (e.g. "like this but cheaper") [5]. The sizable body of critiquing work is surveyed in [6]. There is a small amount of work that combines question-answering with critiquing, e.g. [32].

One characteristic of the work we have surveyed so far is that it requires structured item descriptions, usually in the form of sets of attribute-value pairs: it is at attribute-level that the comparison-based recommendation, the question selection or critiquing take place. We are aware of only one exception: in [38], Vig et al. extend the critiquing idea to items whose descriptions are sets of tags. Similarly, one of the contributions of our work is that we extend the preference-based feedback idea to domains whose items have descriptions that are sets of features such as keywords or tags. There is a relationship here with the large body of work on *relevance feedback* in information retrieval, surveyed in, e.g., [30]. This work often involves modification of a query (e.g. adding terms or modifying weights), whereas we work at the level of items (see Section 3.1.1). There is also recent work on exploratory search that allows users to dynamically influence document ranking by interacting with summaries of their keywords or tags [8, 9].

An alternative is presented in [21], where a latent factor model is learned from user ratings (thus requiring no item descriptions); during a dialog, the user is repeatedly presented with, and may select between, a set of items each of which score low on a system-chosen latent factor and another set whose members score high on that factor, resulting in updates to a vector that captures the user's choices. Similar work is reported in [13] and [7], but with a focus on cold-start users.

An amount of work considers the user's short-term goals both in information retrieval, e.g. [36], and in recommender systems, e.g. [29], where there is sometimes also consideration of long-term preferences [11, 14, 17, 39]. Typically, the short-term preferences are inferred from browsing behaviour, rather than the kind of feedback that we explore in our work, which situates this work also within research into sequence-aware recommender systems [28]. Our approach differs too in that it does not involve learning models from training sets of within-session data.

3 NAVIGATION-BY-PREFERENCE

Navigation-by-Preference (*n-by-p*) is a conversational recommender system that uses preference-based feedback. It is novel in that it is desinged for domains that have unstructured item representations. In high-level terms, *n-by-p* works as follows. The user selects a seed item, s, typically from her user profile. *n-by-p* recommends n candidate items to the user. Let's call the set of recommendations R. From R, the user selects one item — the one which comes closest to what she wants to consume on this occasion (e.g. the movie that is closest to the kind of movie she wants to watch tonight). Her choice, s, becomes the 'query' item for the next cycle. This repeats until she finds a recommendation $s \in R$ that she wants to consume.

Let $\mathbb{I}$ be the set of all items. *n-by-p* works in a scenario of implicit ratings, where the user profile of the active user P is simply a set of items that the active user likes, $P \subseteq \mathbb{I}$. Candidate items for the active user u, I, are items that might be recommended to u; these are simply the items that are not in the user's profile: $I = \{i : i \in \mathbb{I} \setminus P\}$.

Each item i has a set of features (e.g. keywords or tags), denoted f_i. The similarity of two items, $\text{sim}(i, j)$, is given by the Jaccard similarity of their features, $\frac{|f_i \cap f_j|}{|f_i \cup f_j|}$. However, in the version of *n-by-p* that we are describing in this paper, the features are used only indirectly. When reasoning about an item i, this version of *n-by-p* uses a *set of related items*, N_i, which are candidate items that are neighbours of i, i.e. whose similarity to i exceeds a threshold θ: $N_i = \{j \in I, j \neq i : \text{sim}(i, j) > \theta\}$.

Suppose, during the dialog, the user selects an item s from the latest set of recommendations R. Which items might we recommend in the next interaction cycle? The obvious answer is: candidates that are similar to s, i.e. N_s. This is the essence of navigation-by-proposing when it uses preference-based feedback: recommending items that are like the one that the user selected ("more like this"). But, not every member of N_s should be a candidate for recommendation in the next cycle. We exclude any previously recommended items, for example, since we choose not to recommend an item more than once in a dialog. Furthermore, we might take into account the 'negative' feedback that we have received. We know that the user's most recent choice, $s \in R$, is preferred to the other members of R ($R \setminus \{s\}$). We might discard some members of N_s to reflect this fact. We postpone the details of ways of doing this to subsequent sections. But, by way of notation, let's refer to this subset of N_s as the *selection-consistent candidates* and denote it by S, $S \subseteq N_s$.

The next question is: how do we choose n items from S to recommend to the user? We want to ensure that the set of recommendations R on each cycle (a) reflect the user's long-term preferences, as revealed by the user's profile; (b) reflect her short-term preferences, as revealed by the items she has chosen (and not chosen) during the dialog, especially her most recent selection; and (c) are different from each other, to ensure diversity.

A user's long-term preferences L are represented by the candidate items that are neighbours of each item in the user's profile: $L = \cup_{i \in P} N_i$. For (a) above, for each candidate $i \in S$, we can measure how much N_i overlaps with L. Her short-term preferences are given by S itself, so for (b) above, we can measure how much N_i overlaps with S as a whole. For (c), diversity, we can make sure that N_i covers parts of S that were not covered by other recommendations. Again, we postpone the details to subsequent sections.

In this paper, we propose two versions of *n-by-p*: *Navigation-by-Immediate-Preference* (*n-by-i-p*) and *Navigation-by-Cumulative-Preference* (*n-by-c-p*). Both make use of the user's long-term preferences (given by L); they differ in how they handle short-term preferences. In *n-by-i-p*, only feedback from the most recent cycle affects the next cycle; in *n-by-c-p*, we allow feedback from earlier cycles to also affect the next cycle. We now present each in detail.

3.1 Navigation-by-Immediate-Preference

n-by-i-p is shown as Algorithm 1. It initializes the selection-consistent candidates, S, to candidate items that are neighbours of the user-provided seed, N_s. It repeatedly makes a set of n recommendations, R, drawn from S. In each cycle, the user makes a selection, $s \in R$. She also chooses an action, a. In the case that $a = STOP$, the dialog is over; the user has chosen to consume s; in the case that $a = CONTINUE$, s is not ideal but it is the member of R that comes closest to satisfying the user, and the dialog continues. In the latter case, S is updated based on the item that the user selected (s), the ones she did not select ($R \setminus s$) and an update policy (π), yet to be explained. We choose not to recommend an item more than once in a given dialog. We do this by keeping track of all recommendations made so far (*Tabu*) and then excluding these from S.

We will look at recommendation and update in more detail.

3.1.1 *Recommending.*

Recommendation in *n-by-i-p* (Algorithm 2) greedily selects the n members of S that have highest score. The score for an item i, $\text{score}(i, S, L, R)$, depends on the selection-consistent candidates S (to capture short-term preferences), the

Algorithm 1 Navigation-by-Immediate-Preference (*n-by-i-p*)

```
Input: s: seed item, chosen by the user
       L: candidate items that are neighbours of items in P
       π: update policy
       n: number of recommendations per cycle
Output: i ∈ I, a candidate item to consume
 1: $S \leftarrow N_s$
 2: $Tabu \leftarrow \emptyset$
 3: while $|S| > n$ do
 4:     $R \leftarrow \textsc{Recommend}(S, L, n)$
 5:     $s, a \leftarrow$ user chooses $s \in R$ and $a \in \{STOP, CONTINUE\}$
 6:     if $a = STOP$ then
 7:         return $s$
 8:     $S \leftarrow \textsc{Update}(s, R \setminus \{s\}, \pi)$
 9:     $Tabu \leftarrow Tabu \cup R$
10:     $S \leftarrow S \setminus Tabu$
```

Algorithm 2 *n-by-i-p*'s Greedy Recommender

```
Input: S: selection-consistent candidates
       L: candidate items that are neighbours of items in P
       n: number of recommendations per cycle
Output: R, a list of n recommendations
1: function $\textsc{Recommend}(S, L, n)$
2:     $Candidates \leftarrow S$
3:     $R \leftarrow [\,]$
4:     while $|R| < n$ and $|Candidates| > 0$ do
5:         $i^* \leftarrow \arg\max_{i \in Candidates} \text{score}(i, S, L, R)$
6:         append $i^*$ to $R$
7:         $Candidates \leftarrow Candidates \setminus \{i^*\}$
8:     return $R$
```

candidates that are neighbours of items in the user profile L (to capture long-term preferences), and the incrementally-constructed set of recommendations R (so that the next item to be added to R can be different from the ones that have already been added, thus ensuring a level of diversity to the final set of recommendations).

More formally, the score for inserting a candidate i into a (partial) recommendation list R given S and L is a linear combination of short- and long-term scores:

$$\text{score}(i, S, L, R) = (1 - \eta) \cdot \text{ovrlp}(i, S, R) + \eta \cdot \text{ovrlp}(i, L \setminus S, R) \quad (1)$$

η in $[0, 1]$ controls the balance between the short- and long-term scores. In the second term, we pass in $L \setminus S$ instead of L, to ensure that members of S do not get double-counted in the scoring.

$\text{ovrlp}(i, X, R)$ simply measures the overlap between i's neighbours (excluding any that are already covered by R) and a set of items X (where X is either S or $L \setminus S$; see Eq. 1):

$$\text{ovrlp}(i, X, R) = \frac{2 \cdot |(N_i \setminus \text{cov}(X, R)) \cap X|}{|N_i \setminus \text{cov}(X, R)| + |X \setminus \text{cov}(X, R)|} \quad (2)$$

In essence, the numerator is the size of the intersection of the candidate items that are neighbours of i (N_i) and the set X, $N_i \cap X$. However, we do not want to reward i with a high score if it is similar to items that we have already decided to recommend, R. This might result in a set of recommendations R that would lack diversity. Hence, instead of simply computing $N_i \cap X$, we compute $(N_i \setminus \text{cov}(X, R)) \cap X$. We define $\text{cov}(X, R)$ to be the items in X that are already covered by neighbours of items in the partial recommendation list R, i.e. $\text{cov}(X, R) = \bigcup_{j \in R} N_j \cap X$.

The denominator in Eq. 2 is the sum of the sizes of both N_i and X (excluding $\text{cov}(X, R)$). If we divided only by the size of X, we would not penalize items that have high overlap simply by virtue of having more neighbours (large N_i). Since we divide by both, we also double the numerator in compensation, resulting also in a Harmonic mean.

Notice how the definition of ovrlp makes no explicit reference to features. We are reasoning about items through their neighbours (N_i) and considering how the set of neighbours covers the set of items X. Features are being used, but only implicitly: the set N_i contains candidate item that have high feature similarity with i.

Table 1: Update policies: $\textsc{Update}(s, R \setminus \{s\}, \pi)$
Here, s is the selected item; for brevity we write R' for the set of rejected items, $R' = R \setminus \{s\}$; and we write $\text{Sims}(j)$ for the set $\{\text{sim}(j, j') : j' \in R'\}$.

$\pi = Strict$: $S \leftarrow N_s \setminus \bigcup_{j \in R'} N_j$	Discard a member of N_s if it is a neighbour of any member of R'.
$\pi = Relaxed$: $S \leftarrow N_s \setminus \bigcap_{j \in R'} N_j$	Discard a member of N_s if it is a neighbour of every member of R'.
$\pi = Open$: $S \leftarrow N_s$	Do not discard any members of N_s (i.e. ignore R').
$\pi = Mean$: $S \leftarrow N_s \setminus \{j \in N_s : \text{sim}(j, s) < \text{mean}(\text{Sims}(j))\}$	Discard a member of N_s if its similarity to s is less than the mean of its similarities to the members of R'.
$\pi = Max$: $S \leftarrow N_s \setminus \{j \in N_s : \text{sim}(j, s) < \max(\text{Sims}(j))\}$	Discard a member of N_s if its similarity to s is less than the greatest of its similarities to the members of R'.

3.1.2 Updating. Suppose a user selects an item $s \in R$ and chooses action *CONTINUE*. Then we must update the selection-consistent candidates S. Remember that, in *n-by-i-p*, previous rounds of feedback are forgotten. In essence, S becomes N_s, candidate items that are neighbours of the most recently selected item, s. But, we might also take into account the negative feedback: the rejected items $R \setminus \{s\}$. We have defined five different update policies π, which differ in how they make use of the members of $R \setminus \{s\}$. One policy (*Open*) ignores the rejected item entirely; another (*Strict*) ensures that no rejected item will be recommended in the next cycle; and three policies (*Relaxed*, *Mean* and *Max*) lie somewhere between these two extremes. The details are given in Table 1.

3.2 Navigation-by-Cumulative-Preference

Navigation-by-Immediate-Preference ignores feedback that the user gives in all but the most recent cycle of the dialog. This means that the current set of recommended items may contain items that are not related to ones that the user selected earlier, or items that are related to ones that the user rejected earlier. This may confuse the user or prolong the dialog. To better utilize user feedback, we

Algorithm 3 Navigation-by-Cumulative-Preference (*n-by-c-p*)

```
Input: s: seed item, chosen by the user
       L: candidate items that are neighbours of items in P
       ρ: re-weighting policy
       n: number of recommendations per cycle
Output: i ∈ I, a candidate item to consume
 1: $S \leftarrow N_s$
 2: $Tabu \leftarrow \varnothing$
 3: REWEIGHT($s, \varnothing, \rho$)
 4: while $|S| > n$ do
 5:     $R \leftarrow$ RECOMMEND($S, L, n$)
 6:     $s, a \leftarrow$ user chooses $s \in R$ and $a \in \{STOP, CONTINUE\}$
 7:     if $a = STOP$ then
 8:         return $s$
 9:     $S \leftarrow$ UPDATE($s, R \setminus \{s\}, \pi = Open$)
10:     REWEIGHT($s, R \setminus \{s\}, \rho$)
11:     $Tabu \leftarrow Tabu \cup R$
12:     $S \leftarrow S \setminus Tabu$
```

formalise *Navigation-by-Cumulative-Preference* (*n-by-c-p*): each candidate item $i \in I$ has a weight w_i; weights are initially zero; but items are re-weighted based on the user's feedback; and, when scoring items for recommendation, overlap is weight-sensitive.

n-by-c-p (Algorithm 3) is very similar to *n-by-i-p* (Algorithm 1). There are two main differences. The first is that, when it calls UPDATE, it always uses the *Open* update policy. This means that the selection-consistent candidates for the next cycle are all candidates that are neighbours of the most recently selected item (Table 1): no item is discarded. The second difference is that the algorithm calls REWEIGHT. It calls it at the start, so that item weights reflect the user's choice of seed; and it calls it after the user has given feedback, so that weights reflect the user's most recent selection. We will explain recommendation and re-weighting in more detail.

3.2.1 Recommending. Recommendation in *n-by-c-p* is almost identical to recommendation in *n-by-i-p* (shown earlier as Algorithm 2). To save space, we do not present the pseudocode. The only difference is that in line 5, *n-by-c-p* selects the item using a different scoring function. Line 5 becomes $i^* \leftarrow \underset{i \in Candidates}{\arg\max}\ \text{wscore}(i, S, L, R)$. The weighted score, wscore(i, S, L, R), is given by:

$$\text{wscore}(i, S, L, R) = (1 - \eta) \cdot \text{wovrlp}(i, S, R) + \eta \cdot \text{wovrlp}(i, L \setminus S, R) \tag{3}$$

We define wovrlp(i, X, R) as follows:

$$\text{wovrlp}(i, X, R) = \frac{2 \cdot \sum_{j \in (N_i \setminus \text{cov}(X,R)) \cap X} w_j}{|N_i \setminus \text{cov}(X, R)| + |X \setminus \text{cov}(X, R)|} \tag{4}$$

This is very similar to Eq. 2 except that overlap between an item j in $N_i \setminus \text{cov}(X, R)$ and X now counts for w_j, whereas in Eq. 2 it is as if $w_j = 1$ for all j.

We now explain how the weights get modified during the dialog.

3.2.2 Re-weighting. In each cycle, *n-by-c-p* updates the weight w_i of each candidate item i to incorporate the most recent feedback:

$$w_i \leftarrow w_i + \Delta w_i \qquad \forall i \in I \tag{5}$$

Table 2: Re-weighting policies: REWEIGHT($s, R \setminus \{s\}, \rho$) Here, s is the selected item; for brevity we write R' for the set of rejected items, $R' = R \setminus \{s\}$; and d is the depth of the tree, i.e. the number of interaction cycles between the original seed and this set of recommendations R.

ρ = *Directional (Direc)*:

$$\Delta w_i = C_{is} - \sum_{j \in R'} C_{ij}$$

Only considers whether i is a neighbour of s or members of R'.

ρ = *Similarity (Sim)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s) - \sum_{j \in R'} C_{ij} \cdot \text{sim}(i, j)$$

Considers similarities when i is a neighbour of s or members of R'.

ρ = *Similarity-Mean (Smean)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s) - \text{mean}(\{C_{ij} \cdot \text{sim}(i, j) : j \in R'\})$$

Considers similarity when i is a neighbour of s, and the mean similarity when i is a neighbour of members of R'.

ρ = *Similarity-Max (Smax)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s) - \max(\{C_{ij} \cdot \text{sim}(i, j) : j \in R'\})$$

Considers similarity when i is a neighbour of s, and the maximum similarity when i is a neighbour of members of R'.

ρ = *Recency (Rcy)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s)^{1/d} - \sum_{j \in R'} C_{ij} \cdot \text{sim}(i, j)^{1/d}$$

As per *Sim* above, but with updates counting more for later recommendations.

ρ = *Recency-Mean (Rmean)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s)^{1/d} - \text{mean}(\{C_{ij} \cdot \text{sim}(i, j)^{1/d} : j \in R'\})$$

As per *Smean* above, but with updates counting more for later recommendations.

ρ = *Recency-Max (Rmax)*:

$$\Delta w_i = C_{is} \cdot \text{sim}(i, s)^{1/d} - \max(\{C_{ij} \cdot \text{sim}(i, j)^{1/d} : j \in R'\})$$

As per *Smax* above, but with updates counting more for later recommendations.

We have seven different policies ρ for computing Δw_i, and they are given in Table 2. In the policies in Table 2, we use a binary indicator C_{ij}, whose value indicates whether items i and j are related. Specifically, they are related if i is one of the candidate items that are neighbours of j: $C_{ij} = 1$ if $i \in N_j$ and 0 otherwise.

The policies differ in the ways they increase Δw_i when i is related to the item that the user has just selected (given by C_{is}) and decrease Δw_i when i is related to items that the user has just rejected (given by C_{ij} for $j \in R \setminus \{s\}$). In all policies except *Direc*, the amounts added or subtracted are based on the similarities of i to s and to the members of $R \setminus \{s\}$. In three of the policies (*Rcy*, *Rmean* and *Rmax*), updates that come later in the dialog count for more.

3.2.3 Restoring. We have implied that, in every cycle, the user must select an item s from the current set of recommendations R. In fact, in our implementation, we display all the previous recommendations on the screen also (see Figure 1). This affords an option that we have not explained so far. We allow a user to 'jump' back to a previous recommendation. In other words, she can decide that no member of R suits her but that some item that was recommended earlier is more suitable. She can select that earlier item, either to consume (a = *STOP*) or as the basis for a new round of recommendations (a = *CONTINUE*). We have excluded this from the pseudocode shown in this paper (Algorithms 1 and 3) in order to keep the pseudocode simple and intelligible. In *n-by-i-p*, 'jumps' are straightforward because updates are based only on the most recent selection. In *n-by-c-p*, 'jumps' are more complicated because the weights must be restored to previous values. This can be is achieved either by storing the weights for all items on every cycle or, as in our implementation, through a form of backtracking that reverses changes in weights by multiplying them by -1.

4 OFFLINE EXPERIMENTS

We designed an offline experiment, with simulated users, to evaluate the different approaches to *n-by-p*. We wanted the experiment to reveal the effect of the differences between the following:

- *n-by-i-p* versus *n-by-c-p*: The former takes into account only the most recent user feedback and the latter takes into account the feedback across all cycles of the dialog so far.
- *n-by-i-p*'s five update policies (Table 1) and *n-by-c-p*'s seven re-weighting policies (Table 2). These represent different ways of taking negative feedback into account. In all cases, once the user selects item $s \in R$, the next set of recommendations will be drawn from N_s, the candidate items that are neighbours of s. But the policies afford different ways of handling the rejected items $R \setminus \{s\}$.
- The influence of η: η controls the balance between short-term and long-term preferences (Eqs. 1 and 3). When $\eta = 0$, overlap with selection-consistent neighbours S contributes to the score but overlap with profile neighbours L does not, hence only short-term preferences are taken into account. When $\eta = 1$, short-term preferences are ignored. We vary η from 0 (short-term preferences only) to 1 (long-term preferences only) in steps of 0.25.

We also considered different values (0.03, 0.06, 0.09) for the threshold θ in the definition of N_i. However, due to space limitations we only show results for $\theta = 0.03$.

Twelve variations of *n-by-p* with five values of η gives 60 configurations. This justifies the use of an offline experiment: we could not recruit enough participants to compare so many configurations in a user trial. Instead, we use the offline experiment to help us decide which configurations to use in a user trial.

4.1 Experiment settings

We used the hetrec2011-movielens-2k dataset[1] but, in place of the tags given in that dataset, we assigned each movie its keywords from IMDb[2]. We do not modify these keywords in any way, e.g. we do not lemmatize them, nor add their synonyms.

The dataset comprises 2113 users, 5992 movies, 80639 keywords, and over half a million ratings. On average, each movie has 107 keywords (ranging from 2 to 626) and has non-zero similarity with 77% of the other movies. This is quite high, which explains why we did not choose to lemmatize or to add synonyms.

[1]https://grouplens.org/datasets/hetrec-2011/

[2]http://www.imdb.com

We randomly selected 500 users from the dataset to use in the experiments. In *n-by-p*, user profiles simply contain items that the user likes (Section 3). We treated ratings in the dataset of 4 and 5 as 'likes', so active user u's profile P is given by $\{i : r_{u,i} \geq 4\}$. Ratings are otherwise not used in our experiments.

We want to simulate dialogs between each of these users and each of the different configurations of *n-by-p*. The initial seed is chosen at random from the user's profile. But there comes a problem in modeling the simulated user's preferences. Her long-term preferences are obvious: they are given by her profile P. But how do we simulate her short-term preferences? Given a set of $n = 3$ recommendations in each cycle, how do we simulate her preference for one of these over the others? We cannot have her choose the $s \in R$ randomly: that is not the same as exhibiting a short-term preference. Neither can we have her choose the one that is most similar to the items in her profile P because this would make her short-term preferences the same as her long-term preferences. We follow, e.g., [22]: in advance, we choose at random a *target item t* from N_s. In each cycle, from the current set of $n = 3$ recommendations, the user will select the one that is most similar to the target: $\arg\max_{i \in R} \text{sim}(i, t)$. The simulated dialog stops when the target is one of the recommendations, $t \in R$, or after 15 cycles in the case where t itself does not get recommended.

In each cycle, we measure hit-rate up to that cycle (i.e. the proportion of users who have been recommended their target item) and the Jaccard similarity between the item that the simulated user selects in that cycle and her target ($\text{sim}(s, t)$), which we average over all users. We have also measured the diversity of the $n = 3$ recommendations in that cycle, again averaged over all users; and the average surprise of the recommendations in that cycle, averaged over all users. In the case of diversity, we use the measure that [19] denotes by Div_{cont}, which is the average all-pairs distance between items in the recommendation list; for distance, we use the complement of Jaccard similarity. For surprise, we use the measure that [19] denotes by S_{cont}, which is the minimum of the distances between recommended items and items in the user's profile.

4.2 Experiment results

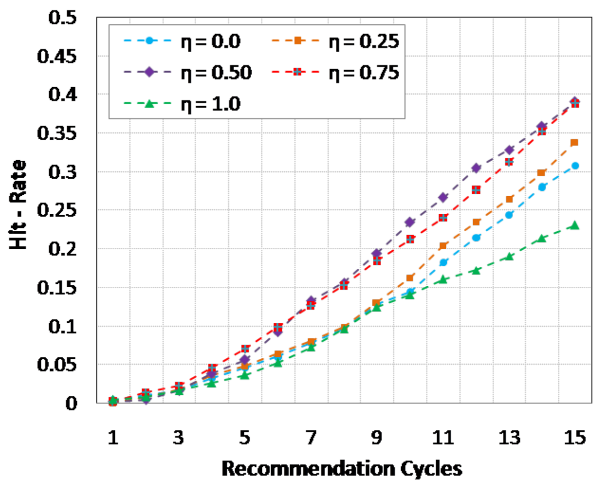
Table 3 shows the results for hit-rate. The columns of the table are the different versions of *n-by-p*. Rows are for different values of η.

For *n-by-i-p*, the highest hit-rate (underlined in the table) is obtained by using the *Open* update policy and with $\eta = 0$. The *Open* policy is the one that does not take the negative feedback into account; and using $\eta = 0$ means that long-term preferences are ignored. We see that, for *n-by-i-p*, increasing the value of η nearly always reduces hit-rates. But there are exceptions where values of η other than 0 give better hit-rates. We also see that policies, such as *Relaxed*, that make most use of the negative feedback, have among the lowest hit-rates.

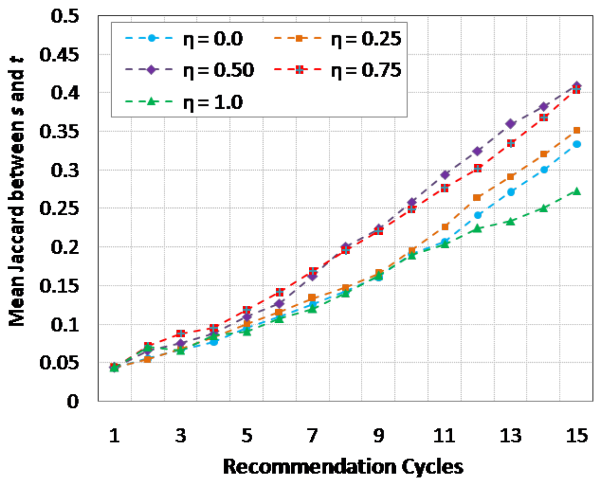
n-by-c-p for the most part has higher hit-rates than *n-by-i-p*, which means that taking previous feedback into account is advantageous. For several of the *n-by-c-p* re-weighting policies, $\eta = 0$ again gives the best results, with hit-rates decreasing as η is increased, but again with exceptions. Of the seven different re-weighting policies, *Smean* is clearly the best. *Smean* with $\eta = 0.5$ attains the highest

Table 3: Offline experiment. The table shows the hit-rate, i.e. the proportion of the 500 simulated users who find their target item within 15 cycles. All systems use $\theta = 0.03$. All differences except those shown in italics are statistically significant with respect to the corresponding version for $\eta = 0.0$ (two-sample Z-test, with $p < 0.05$). The best-performing configurations of *n-by-i-p* and *n-by-c-p* are underlined; the overall best-performing configuration is shown in bold.

	n-by-i-p					*n-by-c-p*						
η	*Strict*	*Relaxed*	*Open*	*Mean*	*Max*	*Direc*	*Sim*	*Smean*	*Smax*	*Rcy*	*Rmean*	*Rmax*
0.00	0.102	0.042	<u>0.204</u>	0.146	0.098	0.112	0.070	0.308	0.282	0.028	0.204	0.174
0.25	0.058	*0.034*	0.152	0.072	0.044	*0.112*	0.040	*0.338*	*0.270*	*0.016*	*0.196*	*0.158*
0.50	0.032	*0.046*	0.036	0.038	0.050	*0.084*	0.040	**<u>0.390</u>**	*0.250*	*0.018*	*0.222*	*0.154*
0.75	0.016	*0.038*	0.014	0.014	0.042	0.072	*0.042*	0.388	*0.266*	*0.018*	*0.162*	*0.134*
1.00	0.012	*0.030*	0.010	0.024	0.042	0.070	0.040	0.230	0.220	*0.020*	0.112	0.122



(a) Hit-rate in each cycle



(b) Mean Jaccard similarity in each cycle between the user's selected item and their target item

Figure 2: Results per cycle for *n-by-c-p* with ρ = *Smean* and different values of η for *random* targets.

hit-rate among all twelve approaches (shown in bold). Since it is a clear winner, we plot further results for this approach only.

Figure 2 shows how *Smean* with different values of η performs over 15 cycles for two of the metrics that we presented in Section 4.1. As explained, a dialog stops when the target item is one of the recommendations. In Figure 2, if a dialog stops before the 15th cycle, we *forward fill* the value of the metric to subsequent cycles. For example if a dialog stops at cycle 8 then, when computing the results in cycles 9 to 15, we include that dialog's values from cycle 8. This ensures that each value that we plot is an average over 500 users. If we did not do this then, in later cycles, we would be plotting an average for a smaller number of users than in the earlier cycles. Plotting over a smaller number of users makes it harder to see trends: differences arise simply by the extra variation that comes from averaging over fewer users.

In Figures 2a and 2b, we see that hit-rate and similarity with the target increase near linearly: as the interaction proceeds, the system leads the user ever closer to her target. We do not have space to show graphs for diversity and surprise but we can summarize them, as follows. Diversity starts at about 0.96 and then mostly decreases for up to 4 cycles (but by only a small amount) and then remains almost the same at about 0.94. Decreasing diversity implies convergence on the item of interest. Surprise starts at about 0.91 and increases very, very slightly to just short of 0.92, which indicates that the process takes the user away from her profile.

5 USER TRIAL

We built a web-based system in order to conduct a user trial. In this trial, we wanted to reveal the effect of using long-term preferences along with short-term preferences, hence we chose to use a system with η not equal to 0.0. The obvious choice was the best-performing configuration from our offline experiment, namely *n-by-c-p* with ρ = *Smean* as its re-weighting policy and $\eta = 0.5$. In this section, we will designate this system by *Smean*@0.5.

We compare *Smean*@0.5 with a baseline system. The baseline has to be a version of *n-by-p* since we do not have other conversational recommenders for domains whose items have unstructured representations. For our baseline, we choose a configuration of *n-by-p* that is as similar as possible to *Smean*@0.5 but which does not take long-term preferences into account, namely *n-by-c-p* with $\rho = Smean$ but with $\eta = 0.0$, whch we will designate by *Smean*@0.0.

We recruited participants through personal email lists and Twitter. In total, 139 people attempted the trial, of whom 102 completed it and have their results reported here. Participants were fully anonymized and we collected no demographic data.

We use the dataset that we used for the offline experiments. However, to increase the chances of user familiarity with the movies, we use only movies released between the years 2000 and 2011 inclusive: 1851 ($\approx$ 30%) of the 5992 movies in the whole dataset.

The user trial is a between-subject trial: participants are assigned at random to interact either with the *Smean*@0.5 recommender or the *Smean*@0.0 recommender.

5.1 User trial protocol

Each participant began by creating a user profile containing 10 movies. The instructions were that the movies should be ones that the user likes. The user interface offers both a scrollable grid of movies and a search box to enable the user to find these movies.

The user profile captures a user's long-term preferences. The challenge in designing an experiment of this kind is to create the conditions under which a user also has ephemeral (short-term) preferences that she wants to satisfy [27]. Most likely, because of these ephemeral preferences, the user should be dissatisfied to some extent with recommendations based purely on her profile, because these will satisfy only her long-term preferences. We rejected the idea of picking a target item and showing it to the user. We felt that this would lead to an approximation to the offline experiments that we have already described, where in every cycle the (simulated) user always selects the recommended item that is most similar to the target. What we wanted was a scenario in which a user would have an ephemeral goal, but where she would not know exactly what movie she wanted to watch, and yet where she would be able to make reasonably consistent judgements about a set of recommendations on the basis of that ephemeral goal.

The strategies for doing this in [27] rely on having structured item descriptions (e.g. sets of attribute-value pairs), which we do not have. In the end, we designed a novel protocol, which we believe is one of the contributions of our work. The scenario is that the user is trying to find a movie to watch with another person, hence she has to find one that she thinks both she and her putative companion will enjoy watching *together*. From a list of eight people (Mother, Father, Brother, Sister, Aunt, Uncle, Nephew, Niece), we ask her to select a person she knows but whose movie preferences differ from her own (see the top third of Figure 3). We tell her to choose from her profile the movie that she thinks is the best movie to watch together with this person (middle third of Figure 3). We ask her how much she thinks they will enjoy watching the movie together (*Not at all*, *Barely at all*, etc.) (lower third of Figure 3). If she thinks they will enjoy watching the movie *Somewhat* or *A lot*, we ask her to repeat the whole process (selecting a different person) in the hope of finding a scenario where short-term preferences will differ from long-term preferences. At most, a user goes through this process a total of three times. The movie that she has selected from her profile at this point becomes the initial seed in the dialog. (We do make sure to include results that distinguish between 'easy' and 'difficult' dialogs, depending on how much room for improvement on the seed is possible: Section 5.2.1.)

We believe this scenario satisfies the criteria above: the user has an ephemeral goal, does not know exactly what movie she wants, but can make judgments when faced with descriptions of movies that we recommend. We emphasize that this protocol is simply a way of creating a scenario in which a user has an ephemeral goal. We are not building a group recommender system. [16]. In our user trial, the recommender does not have any explicit representation of the other person's tastes.

Now that the scenario has been established and the seed has been chosen, a dialog of eight cycles begins. In each cycle, the system displays the next $n = 3$ recommendations, building a tree from left to right on the screen (Figure 1). The user can mouse-over the nodes and edges to find out movie details and keywords that connect movies, respectively. She must choose the recommendation that she thinks she and her putative companion will most enjoy watching together. If none of the three recommendations seem right, the user can choose a movie from earlier in the tree, in which case the system reverts to an earlier state (see Section 3.2.3). We require every user to run the system for a full eight cycles, so that the tree has a depth of eight, even if she sees a movie earlier that she thinks is ideal. The advantage of this is that every participant's responses are based on the same number of movies on the screen, which makes for fair comparisons. We think this outweighs the possible disadvantage that, if a user has seen a 'perfect' item, she must nevertheless continue with the dialog, presumably receiving suboptimal recommendations until she has completed eight cycles, which may negatively affect her opinion of the system.

At the end of the dialog, the screen will be displaying a tree, rooted by the seed and containing 24 recommended movies (see Figure 1). We ask the user to select one of the 24 movies, the one that she thinks she and her putative companion will most enjoy watching together. Then we ask her five questions:

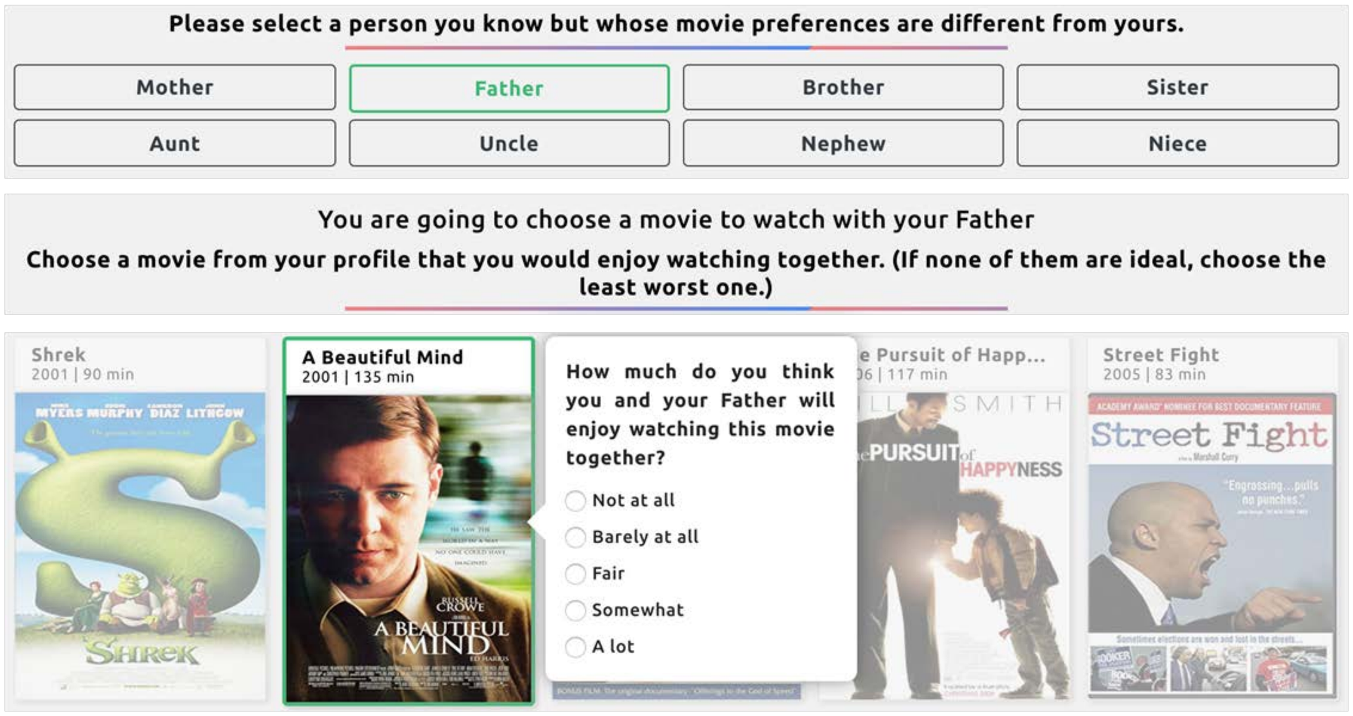
- *Familiarity*: Have you actually seen the movie <*selected movie*> before?
- *Relevance*: How much do you think you and your <*selected person*> will enjoy watching <*selected movie*> together?
- *Serendipity*: Is <*selected movie*> a pleasantly surprising recommendation?
- *Effectiveness*: Did you find the recommendations helpful?
- *Satisfaction*: Did you enjoy using the system?

The user chooses between Yes and No in answer to the question about *Familiarity*. For the other questions, she chooses from a 5-point scale: *Not at all*; *Barely at all*; *Fair*; *Somewhat*; and *A lot*.

5.2 User trial results

102 participants completed the trial, 51 per system. Table 4 summarizes their responses.

- *Familiarity*: 62.8% of users of *Smean*@0.5 have actually seen their selected movie compared with 54.9% of users of *Smean*@0.0.
- *Relevance*: 76.5% of the users of *Smean*@0.5 judged their selected movie to be one that they and their putative companion would enjoy *Somewhat* or *A lot*; in the case of *Smean*@0.0, this was just 49.0% of the users.


Please select a person you know but whose movie preferences are different from yours.

Mother | Father | Brother | Sister | Aunt | Uncle | Nephew | Niece

You are going to choose a movie to watch with your Father

Choose a movie from your profile that you would enjoy watching together. (If none of them are ideal, choose the least worst one.)

How much do you think you and your Father will enjoy watching this movie together?

- Not at all
- Barely at all
- Fair
- Somewhat
- A lot

Figure 3: Parts of screenshots showing selection of a companion and a seed movie.

Table 4: Responses to survey questions in the user trial.

User's Response	*Smean*@0.5 *Relevance*	*Smean*@0.5 *Serendipity*	*Smean*@0.5 *Effectiveness*	*Smean*@0.5 *Satisfaction*	*Smean*@0.0 *Relevance*	*Smean*@0.0 *Serendipity*	*Smean*@0.0 *Effectiveness*	*Smean*@0.0 *Satisfaction*
Not at all	0	2	0	1	3	5	5	2
Barely at all	1	4	5	5	7	4	4	5
Fair	11	12	14	12	16	21	17	14
Somewhat	18	19	21	20	12	12	16	18
A lot	21	14	11	13	13	9	9	12

- *Serendipity*: 64.7% of users of *Smean*@0.5 thought their selected movie was pleasantly surprising (*Somewhat* or *A lot*); for *Smean*@0.0, this was just 41.2% of the users.
- *Effectiveness*: 62.7% of the users of *Smean*@0.5 found the recommendations to be helpful (*Somewhat* or *A lot*); in the case of *Smean*@0.0, this was just 49.0% of the users.
- *Satisfaction*: 64.7% of the users of *Smean*@0.5 enjoyed using the system (*Somewhat* or *A lot*); in the case of *Smean*@ 0.0, this was just 58.9% of the users.

On all criteria, *Smean*@0.5 produced better recommendations. However, the difference was statistically significant only for the *Relevance* and *Serendipity* questions. (We used a one-sided Z-test for proportions, with significance level $p < 0.05$. The null hypothesis was that those preferring *Smean*@0.0 are greater than or equal to those preferring *Smean*@0.5, ignoring those who were neutral i.e. who answered *Fair*.)

5.2.1 Change in relevance. The results we have given so far ignore the user's opinion of the initial seed: do the systems *improve* upon the initial seed? We designed a statistic to answer this question. We assign integers in [1, 5] to the responses, 1 = *Not at all*, 2 = *Barely at all*, etc. We let α_v be the number of participants who assigned a value of v to the initial seed, i.e. α_1 is the number of people who judged the seed to be 1 (= *Not at all*) suitable, α_2 is the number who judged the seed to be 2 (= *Barely at all*) suitable. Similarly, let ω_v be the number of participants who assigned a value of v to the final selected movie. Then, we can compute the improvement that the system makes by taking the difference in the responses divided by the maximum improvement that could be made:

$$\text{improvement} = \frac{\sum_{v=1}^{5} v \cdot \omega_v - \sum_{v=1}^{5} v \cdot \alpha_v}{\sum_{v=1}^{5} 5 \cdot \omega_v - \sum_{v=1}^{5} v \cdot \alpha_v} \tag{6}$$

For *Smean*@0.5, improvement = 0.5114, whereas for *Smean*@0.0, improvement = 0.0723. It is clear that in terms of expected movie

Table 5: Comparison of decision effort. All values are averaged over participants who liked their final selected movie *Somewhat* or *A lot.*

Measure of effort	*Smean*@0.5	*Smean*@0.0
Nodes displayed	26.85	29.56
Node mouse-overs	19.28	22.84
Edge mouse-overs	10.03	11.92
Cycles	5.36	4.12
Time taken (secs.)	251.93	300.58

enjoyment, *Smean*@0.5 does a better job of taking users from their initial seeds to a final movie selection.

We have also re-computed the improvement, this time excluding those cases where there is little or no scope for improvement, i.e. those cases where the user thinks that she and her putative companion would enjoy watching the seed movie *Somewhat* or *A lot.* Now, there are only 34 users of interest for *Smean*@0.5 and 33 for *Smean*@0.0. For these users only, we obtain improvement = 0.6049 for *Smean*@0.5 and improvement = 0.2533 for *Smean*@0.0. This gives a fairer picture of *Smean*@0.0, but *Smean*@0.5 performs even better in these more difficult cases.

5.2.2 *User effort.* Finally, we consider how much effort users expended. Table 5 summarizes the effort for users whose final selected movie was one they thought that they and their putative companions would like *Somewhat* or *A lot.* (We excluded other users because, in some sense, their dialog is incomplete since they have not found a satisfactory movie. Since this gives only 25 users for *Smean*@0.0, we used a one-sided *t*-test, with $p < 0.05$, with null hypothesis that *Smean*@0.0 needs less or equal effort.)

- *Nodes displayed*: As described before, we required users to explore for eight cycles. In these eight cycles, every user is shown 25 nodes (1 seed and 24 recommended movies). But a user can jump (reverting to an earlier set of recommendations), which leads to more recommendations being made. In both systems, the average is a little above 25, which shows that there was some jumping. But the average was higher for *Smean*@0.0, which implies a greater need for jumping ($p = 0.022$, which is statistically significant).
- *Node mouse-overs*: This refers to the average number of movies whose descriptions were viewed by mousing-over the node. More movies were examined by users of *Smean*@0.0 ($p = 0.016$, which is statistically significant).
- *Edge mouse-overs*: Mousing over an edge reveals keywords that the movies at each end have in common. On average, more of this was viewed by users of *Smean*@0.0 ($p = 0.077$, which is not statistically significant).
- *Cycles*: This refers to the average number of cycles needed in order for the final selected movie to be shown. It was slightly higher for users of *Smean*@0.5 ($p = 0.021$, which is statistically significant). But it must be remembered that users of *Smean*@0.5 find movies that they regard as better final choices.
- *Time taken*: This is the average task completion time in seconds. It was higher for users of *Smean*@0.0 ($p = 0.211$, which is not statistically significant).

It can be seen that both systems require quite similar effort from users. There seems to be a little more effort in the case of *Smean*@0.0 (more jumps and more time spent making sense of the recommendations by mousing-over their details). On the other hand, the final movie is found around the 5th or 6th cycle on average for users of *Smean*@0.5 and around the 4th cycle for users of *Smean*@0.0, but is a less satisfactory movie in the latter case.

6 CONCLUSIONS

Navigation-by-Preference (*n-by-p*) is a conversational recommender system that works on unstructured item descriptions to help a user construct and articulate her short-term preferences, while aiming to minimize the effort of reaching an item of interest. We believe that *n-by-p* has the following characteristics:

- *Preference-based feedback*: Preference-based feedback (where the user simply selects one of the current recommendations) is the simplest form of feedback. It does not require the user to articulate which features of the item she likes or dislikes or how she wants to change them. This simplicity for the user means ambiguity for the system: there is no explicit feedback about the features [24].
- *Configurability*: *n-by-p* is highly configurable. We have described two variants, the first with five update policies; the second with seven re-weighting policies. These allow us to choose how to combine the user's preferences in ways that are best suited to the domain of application.
- *Interpretability*: *n-by-p* is a content-based approach based on keywords or tags that items have in common, which makes it easy to understand the relationship between pairs of consecutive items in a preference chain.

We presented an offline experiment, with simulated users, that selected the best of 60 different configurations of *n-by-p.* Then we used a web-based system to conduct a user trial with a novel protocol. The trial showed with statistical significance that this configuration, which combines short-term preferences with long-term preferences, produces more accurate and serendipitous recommendations without greater effort from its users.

In future work, we will investigate versions of *n-by-p* that use the item features more directly and compare these versions with the ones described in this paper.

ACKNOWLEDGMENTS

We thank Arpit Jain for help with web design. This paper emanates from research supported by a grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 which is co-funded under the European Regional Development Fund.

REFERENCES

[1] Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. 2011. Context-aware recommender systems. *AI Magazine* 32, 3 (2011), 67–80.

[2] Xavier Amatriain, Josep M. Pujol, Nava Tintarev, and Nuria Oliver. 2009. Rate it again: Increasing recommendation accuracy by user re-rating. In *Procs. of the Third ACM Conference on Recommender SSystems.* 173–180.

[3] Derek Bridge, Mehmet H. Göker, Lorraine McGinty, and Barry Smyth. 2005. Case-based recommender systems. *Knowledge Engineering Review* 20, 3 (2005), 315–320.

[4] Robin Burke. 2000. Knowledge-based recommender systems. In *Encyclopedia of Library and Information Systems, vol.60, no.32.* Marcel Dekker, 180–200.

[5] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. 1997. The FindMe approach to assisted browsing. *IEEE Expert: Intelligent Systems and Their Applications* 12, 4 (1997), 32–40.

[6] Li Chen and Pearl Pu. 2012. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction* 22, 1 (2012), 125–150.

[7] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Procs. of the Twenty-second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 815–824.

[8] Cecilia di Sciascio, Peter Brusilovsky, and Eduardo Veas. 2018. A study on user-controllable social exploratory search. In *Procs. of the Twenty-third International Conference on Intelligent User Interfaces.* 353–364.

[9] Cecilia di Sciascio, Vedran Sabol, and Eduardo E. Veas. 2016. Rank as you go: User-driven exploration of search results. In *Procs. of the Twenty-first International Conference on Intelligent User Interfaces.* 118–129.

[10] Michelle Doyle and Pádraig Cunningham. 2000. A dynamic approach to reducing dialog in on-line decision guides. In *Procs. of the Fifth European Workshop on Case-Based Reasoning.* 49–60.

[11] Yingpeng Du, Hongzhi Liu, Yuanhang Qu, and Zhonghai Wu. 2018. Online personalized next-item recommendation via long short term preference learning. In *Procs. of the Pacific Rim International Conference on Artificial Intelligence.* 915–927.

[12] Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational AI. In *Procs. of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.* 1371–1374.

[13] Mark P. Graus and Martijn C. Willemsen. 2015. Improving the user experience during cold start through choice-based preference elicitation. In *Procs. of the Ninth ACM Conference on Recommender Systems.* 273–276.

[14] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yinglong Wang, Jun Ma, and Mohan S. Kankanhalli. 2018. Attentive long short-term preference modeling for personalized product search. *CoRR* abs/1811.10155 (2018).

[15] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2014. Context adaptation in interactive recommender systems. In *Procs. of the Eighth ACM Conference on Recommender Systems.* 41–48.

[16] Anthony Jameson and Barry Smyth. 2007. Recommendation to groups. In *The Adaptive Web*, Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl (Eds.). Springer-Verlag, 596–627.

[17] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015. Adaptation and evaluation of recommendations for short-term shopping goals. In *Procs. of the Ninth ACM Conference on Recommender Systems.* 211–218.

[18] Nicolas Jones, Armelle Brun, and Anne Boyer. 2011. Comparisons instead of ratings: Towards more stable preferences. In *Procs. of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, Vol. 1. 451–456.

[19] Marius Kaminskas and Derek Bridge. 2016. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Transactions on Interactive Intelligent Systems* 7, 1 (2016), 2:1–2:42.

[20] Joseph A. Konstan and John Riedl. 2012. Recommender systems: From algorithms to user experience. *User Modeling and User-Adapted Interaction* 22, 1-2 (2012), 101–123.

[21] Benedikt Loepp, Tim Hussein, and Jüergen Ziegler. 2014. Choice-based preference elicitation for collaborative filtering recommender systems. In *Procs. of the SIGCHI Conference on Human Factors in Computing Systems.* 3085–3094.

[22] Lorraine Mc Ginty and Barry Smyth. 2002. Comparison-based recommendation. In *Procs. of the Sixth European Conference on Case-Based Reasoning.* Springer, 575–589.

[23] Lorraine Mc Ginty and Barry Smyth. 2003. On the role of diversity in conversational recommender systems. In *Procs. of the Fifth International Conference on Case-Based Reasoning.* Springer, 276–290.

[24] Lorraine Mcginty and Barry Smyth. 2006. Adaptive selection: An analysis of critiquing and preference-based feedback in conversational recommender systems. *International Journal of Electronic Commerce* 11, 2 (2006), 35–57.

[25] Roberto Pagano, Paolo Cremonesi, Martha Larson, Balázs Hidasi, Domonkos Tikk, Alexandros Karatzoglou, and Massimo Quadrana. 2016. The contextual turn: From context-aware to context-driven recommender systems. In *Procs. of the Tenth ACM Conference on Recommender Systems.* 249–252.

[26] Pearl Pu and Li Chen. 2008. User-involved preference elicitation for product search and recommender systems. *AI Magazine* 29, 4 (2008), 93–103.

[27] Pearl Huan Z Pu and Pratyush Kumar. 2004. Evaluating example-based search tools. In *Procs. of the Fifth ACM Conference on Electronic Commerce.* 208–217.

[28] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *ACM Comput. Surv.* 51, 4 (2018), 66:1–66:36.

[29] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Procs. of the Eleventh ACM Conference on Recommender Systems.* 130–137.

[30] Ian Ruthven and Mounia Lalmas. 2003. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.* 18, 2 (2003), 95–145.

[31] Sascha Schmitt. 2002. *simVar*: A similarity-influences question selection criterion or e-sales dialogs. *Artificial Intelligence Review* 18, 3–4 (2002), 195–221.

[32] Hideo Shimazu. 2002. ExpertClerk: A conversational case-cased reasoning tool for developing salesclerk agents in e-commerce webshops. *Artificial Intelligence Review* 18, 3-4 (2002), 223–244.

[33] Barry Smyth. 2007. Case-based recommendation. In *The Adaptive Web*, Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl (Eds.). Springer-Verlag, 342–376.

[34] Barry Smyth and Lorraine McGinty. 2003. An analysis of feedback strategies in conversational recommenders. In *Procs. of the Fourteenth Irish Artificial Intelligence and Cognitive Science Conference.*

[35] Barry Smyth and Lorraine McGinty. 2003. The power of suggestion. In *Procs. of the International Joint Conference on Artificial Intelligence*, Vol. 3. 127–132.

[36] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. 2004. Adaptive web search based on user profile constructed without any effort from users. In *Procs. of the Thirteenth International Conference on the World Wide Web.* 675–684.

[37] Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *Procs. of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.* 235–244.

[38] Jesse Vig, Shilad Sen, and John Riedl. 2011. Navigating the tag genome. In *Procs. of the 16th International Conference on Intelligent User Interfaces.* 93–102.

[39] Chao-Yuan Wu, Christopher V. Alvino, Alexander J. Smola, and Justin Basilico. 2016. Using navigation to improve recommendations in real-time. In *Procs. of the 10th ACM Conference on Recommender Systems.* 341–348.