Rush: Repeated Recommendations on Mobile Devices

Dominikus Baur
University of Munich
dominikus.baur@ifi.lmu.de

Sebastian Boring
University of Munich
sebastian.boring@ifi.lmu.de

Andreas Butz
University of Munich
andreas.butz@ifi.lmu.de

ABSTRACT
We present *rush* as a recommendation-based interaction and visualization technique for repeated item selection from large data sets on mobile touch screen devices. Proposals and choices are intertwined in a continuous finger gesture navigating a two-dimensional canvas of recommended items. This provides users with more flexibility for the resulting selections. Our design is based on a formative user study regarding orientation and occlusion aspects. Subsequently, we implemented a version of rush for music playlist creation. In an experimental evaluation we compared different types of recommendations based on similarity, namely the top 5 most similar items, five random selections from the list of similar items and a hybrid version of the two. Participants had to create playlists using each condition. Our results show that top 5 was too restricting, while random and hybrid suggestions had comparable results.

Author Keywords
Interaction technique, mobile; recommender systems

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Design, Experimentation, Human Factors

INTRODUCTION
Recommender Systems have come a long way [2]: while initially conceived as a way to handle email information overload by collaborative filtering [9], they soon were adapted by online retailers (most prominently Amazon.com) to increase sales. With this history, recommender systems continued to be used mainly in web interfaces and for reducing large data sets to well-chosen subsets in order to conserve bandwidth and prevent information overload.

Despite broadband internet connection and increased processing power in mobile devices, explicit research on user interfaces for mobile recommender systems is scarce: existing systems ([14],[18]) mostly rely on established desktop interaction metaphors (e.g., critique-based recommendation [21]) and examine issues of mobility such as loss of connection ([19],[8]) and decentralization [13]. Peculiarities of mobile device interaction, such as occlusion problems [35], the influence of the reduced screen space [30] and possibly abrupt endings (e.g., when the bus arrives at the station) have mostly been ignored.

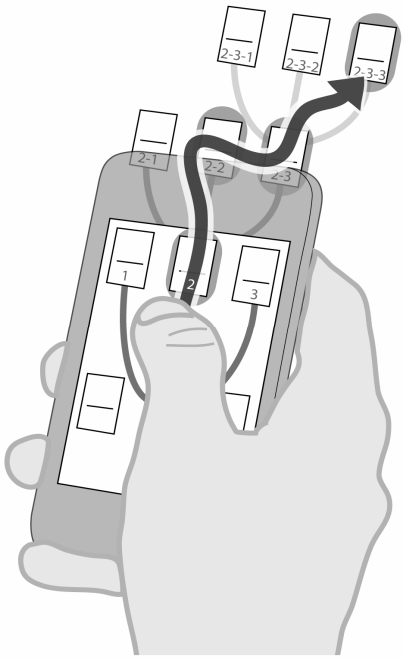

Figure 1. Repeated selection from recommendation sets

Ward et al. presented Dasher [36], a visual tool for text entry based on language models that has also been successfully ported to Pocket PCs. A continuous gesture allows selecting letters to form words and sentences. The underlying language model is used to enlarge more probable items and make selecting the correct one easier. With up to 60 words per minute in its original version, it is an efficient way to enter text. Despite being used in a variety of other ways (e.g., with an eye-tracker [37]), the original task of text entry has never been changed, though.

In this paper, we present rush (see Figure 1), a variation on Dasher, as an interaction technique for mobile touch-screen devices for repeatedly selecting items from a set of recommendations.

IUI'10, February 7–10, 2010, Hong Kong, China.

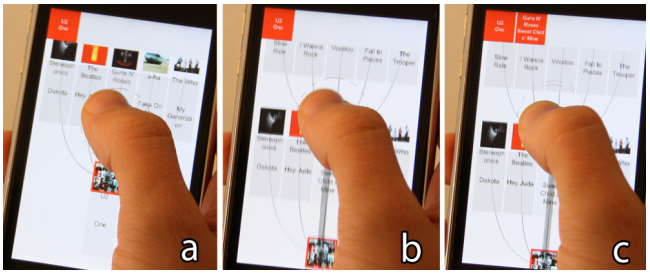

Figure 2. Rush overview: a) Starting from a seed item five recommendations are displayed. b) Touching the middle item causes a new set of recommendations tailored to this item to appear above. c) By completing the crossing gesture, the middle item is added to the selected set.

Similar to Dasher, rush's interaction takes place on a virtual two-dimensional canvas. Starting from a seed item, related items are selected by the underlying recommender engine and displayed close to it. The user can then select one of these suggestions, which in turn generates recommendations related to this item (see Figure 2). This iterative expansion of a recommendation tree continues until the user is satisfied with the set of selections. Navigation and selection happens with a single finger gesture: the canvas moves below the finger depending on the distance and angle to the screen's center. For example, the user's finger in the upper right part of the screen causes the canvas to slide towards the lower left. To allow fluid gestures and prevent the need to lift a finger, we used crossing gestures [1] for the selection of items instead of pointing. Selecting an item in rush is performed by drawing a line through it. In theory, the user's interaction thus limits itself to moving the finger on the screen: putting the finger down starts the process and lifting it again means the collection is finished.

In the following, we discuss the issues of device and interface orientation and a formative evaluation that led to rush's final design. We also present a user study where we examined a rush implementation for music playlists and the influence of the underlying recommendation on user satisfaction. Finally, we discuss possible extensions.

RELATED WORK

Interaction for recommender systems is often combined with approaches from information retrieval and visualization. O'Donovan et al. [22] built an interactive visualization as a way to provide users with explanations of the collaborative filtering process and as a way to influence the results.

Swearingen et al. [32] analyzed eleven online recommender systems and identified the importance of transparency, familiarity with items and providing details. The advantages of transparency and explanations in recommender systems and a design adapted towards them have been addressed by Pu et al. [27] and Tintarev et al. [33]. The longstanding GroupLens and MovieLens projects also analyzed how to gather information on users through different interface additions (e.g., [34]).

Conversational recommenders and mixed initiative systems [31] ask questions or make repeated suggestions to help the user understand an item set and ultimately make a choice. They are used to specify the requirements of the user and make more refined recommendations. Still, their goal is to recommend a single item and not multiple ones. The recommendation process is over if this item has been found.

Recommending collections

Hansen et al. discuss the challenges and present the design space of automatically recommending collections [11]: In addition to finding suitable items, such systems also have to consider how well these items fit together and in which order they should be presented. The music domain already provides multiple systems for recommending song collections, mostly commercially driven: Websites like Last.FM, imeem or Pandora let users listen to a dynamically generated web-radio based on a chosen seed song. Similarly, playlist generators like iTunes Genius or Microsoft's Smart DJ produce playlists for desktop or mobile music players.

The underlying method for generating such playlists is mostly based on collaborative filtering ([24], [17]), but there are also systems that analyze user interaction, such as skipping behavior [23], audio similarity [26] or patterns in authored streams (e.g., radio playlists) [28]. While automatic playlist generation is fast and convenient, its results often lack variety or ignore the importance of song order.

Apart from fully automatic processes, the user can be involved to varying degrees: Aucouturier et al. [6] let the user define constraints and generate a playlist based on them. SatisFly [25] is an interface that is also based on constraints. Downsides of the constraint-based approach are: (1) it becomes complex if more elaborate constraints are used and (2) all constraints have to be known beforehand.

Music on Mobile Devices

Music has become mobile with the proliferation of handheld MP3-devices such as Apple's iPod. But with the growing storage space on such devices the problem of accessing items became worse. Mobile visualization of music promises to make collections manageable: Mapping approaches such as Artist Map [40], PocketSOMPlayer [20] or Mobile Music Explorer [10] visualize music items using dimensionality reduction techniques and provide an overview of the whole collection. Generating playlists in such visualizations is mostly done by drawing lines through the map [39], [10], thus causing the system to choose a list of songs following this trajectory. Due to the abstraction of the visualization, the user can influence the resulting playlist only on a very high level (mostly genre).

FORMING RUSH

In contrast to desktop user interfaces, mobile applications face additional problems such as readability issues due to the device's orientation. As mobile devices mostly have rectangular shapes, there are two ways of holding them: (1) vertically and (2) horizontally. In the latter case, users can

either grip both sides with both hands and interact with their thumbs or hold one side with the non-dominant hand while interacting with the dominant one. One-handed interaction on the whole screen is only possible in the former case.

		Interface Orientation	
		Vertical	Horizontal
Interface Direction	Up-Right	Up	Right
	Down-Left	Down	Left

Table 1. Which interface directions are available depends on the interface orientation. The "forward" direction where new items appear is based on these two factors.

Occlusion introduces a further problem when interacting with small screens: if important parts of the interface are regularly occluded by fingers, the performance drops [35]. Several solutions to this problem have been proposed: most of them require additional screen space (thus occluding other parts of the interface) [35] or special hardware [38]. One obvious solution is to re-arrange the interface content so that occlusion is minimized.

In rush, the interaction happens mostly in one direction: forward movements show new suggestions for a specific item and select it. When moving backwards, users can undo a selection or receive recommendations for a different item. However, as the latter case is rare, the forward direction should be optimized. The four directions, namely up, down, left and right, are feasible candidates for forward movement, leading to different kinds of occlusion: if new items appear to the right of an item, then right-handers will occlude them, while left-handers have no problems. The bottom-direction is mostly occluded with either hand, as the device is held there. The up-direction should not suffer from any occlusion-problems.

Two solutions to the problem of occlusion thus are feasible, namely shaping interaction towards the upper side of the device or flipping the interface for left- and right-handers. This last solution should lead to no problems with nominal data such as products, which are typically found in recommendation situations. For ordinal data such as letters (as in Dasher [36]), the reading direction might have an influence on the performance. This makes flipping for right-handers less attractive if they have a Western background (and thus a reading direction from left to right).

In order to find an optimal design for rush, we wanted to clarify these uncertainties. Therefore, we performed a pre-study to examine the influence of (1) device and interface orientation, (2) interface direction, (3) used hand and (4) handedness on the user's performance.

Method

We implemented a version of rush that was focused on these interface attributes. We chose all selectable items to represent single digits (see Figure 3). We then generated sets that contained random numbers and orders and presented them to each participant.

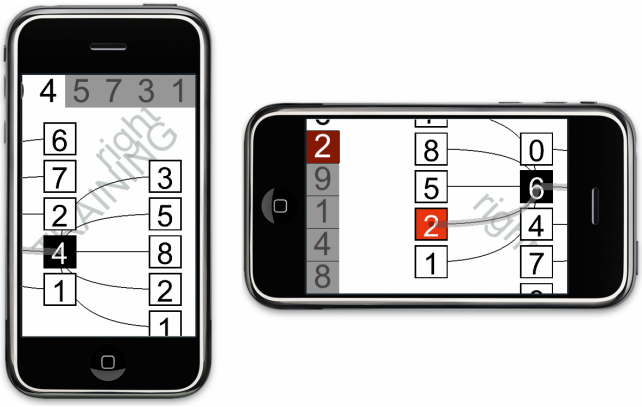

Figure 3. Two different conditions for the rush pre-study. Left: vertical device, horizontal interface, direction right. Right: horizontal device, horizontal interface, direction left (the "right" label in the background tells participants to use their right hand for interaction)

This version of rush supported two different orientations for the device (horizontal and vertical). We divided the four possible movement directions into interface orientation (horizontal, i.e. sets of recommended items appear to the left or right of the current item (see Figure 3) and vertical, i.e. sets of recommended items appear above or below (see Figure 1)) and the two resulting interface directions (horizontal interface orientation: either left or right direction, vertical interface orientation: either up or down direction, see Figure 3 for two examples). The available directions are dependent on the interface orientation, so we combined the up and right (up-right) and down and left (down-left) directions to turn interface direction into a variable with two states. In addition to that, participants were told to use either their left or right hand for a task. The movement speed on the canvas depended on the finger's distance to the center of the screen, so that twice the distance resulted in twice the speed. Our test device was an Apple iPhone 3G, with a screen resolution of 320 × 480 pixels. To keep all orientations comparable, the movement speed was capped at a distance of 160 pixels to the center of the screen. Otherwise, interaction in the longer direction would have allowed higher movement speeds and thus better results.

Task and Study Design

Participants had to select ten numbers using the rush interface. For each item, five suggestions were given out of which only one item was the correct one. The location where new items appeared ("forward") depended on the interface orientation as well as the interaction direction (see Table 1). By forcing participants to use both of their hands,

we partially provoked occlusion and were able to measure its effects on performance.

We measured task time and error rates for each trial. The task time began as soon as the participant put a finger on the screen and ended when the last item was selected. Errors were counted for both selecting a wrong item as well as deselecting a correct one. All participants performed this task for every combination of display orientation, interface orientation, interface direction and used hand. The order of the tasks was randomized to counter learning effects. Before each task, participants performed a practice run with a different sequence of numbers using the identical interface condition.

We used a within-subjects study design. We had a *2 Device Orientations* (*Horizontal*, and *Vertical*) × *2 Interface Orientations* (*Horizontal*, and *Vertical*) × *2 Interface Directions* (*up-right*, and *down-left*) × *2 Used Hand* (*Left*, and *Right*) design. For each combination, participants had one practice block and one timed block. In each task, we measured task time and error rate. The resulting design was:

2 Device Orientations (*Horizontal*, and *Vertical*) ×
2 Interface Orientations (*Horizontal*, and *Vertical*) ×
2 Interface Directions (*up-right*, and *down-left*) ×
2 Used Hand (*Left*, and *Right*) ×
2 Blocks (Training, and *Timed*)

= 32 (16 timed) data points per participant.

Participants

We recruited 12 participants (3 female, 10 right-handed) from our institution with their age ranging from 21 to 32 (average age was 27.4 years). All participants had at least some previous experience with touch screens.

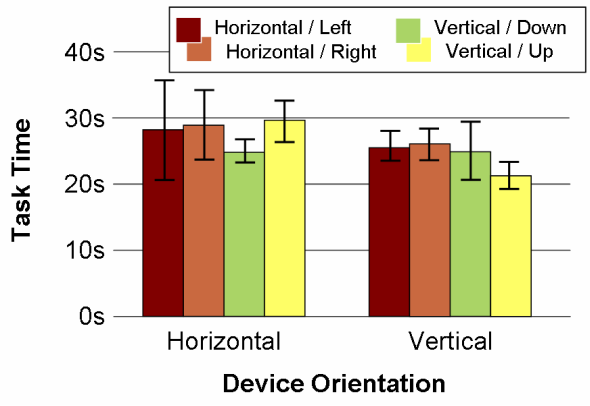

Figure 4. Task times from the pre-study

Hypotheses

Based on our understanding of performance of mobile interfaces we had three hypotheses: occlusion in general leads to higher task times as users have to adjust their hand's position to identify items (H1). The dominant hand outperforms the non-dominant one in both task times and error rates (H2). The interface and device orientation correlate with faster task times (H3).

Results

We conducted a repeated measures ANOVA test on mean completion times (see Figure 4 for results) and error rates. To identify the nature of interaction effects, we performed additional tests on subsets of our data. All post hoc pairwise comparisons used Bonferroni corrected confidence intervals for comparisons against $\alpha = 0.05$.

We first analyzed whether the handedness of users had any influence on the results. The mean completion time of left-handers was 24.12 seconds when they used their left hand and 26.46 when they used their right hand respectively. There was almost no difference for right-handers (26.34 seconds for the right hand compared to 26.39 seconds for the left hand respectively). However, we did not find any significant main effects or interactions for this between-subject factor on both task time and error rate. Thus, we excluded the handedness for sub-sequent analysis. This is contradictory to our hypothesis H2 as the handedness does not have any significant effects on task times or error rates.

We found significant main effects on completion time for both *Device Orientation* ($F1,10 = 13.056$, $p < 0.005$) and *Interface Orientation* ($F1,10 = 7.094$, $p < 0.024$). There were no significant interaction effects in our data. Overall, the *Vertical Device Orientation* (M=24.49, SD=1.29) was faster than the *Horizontal* one (M=27.85, SD=1.28). The *Vertical Interface Orientation* (M=25.16, SD=1.23) was also faster than the *Horizontal Orientation* (M=27.17, SD=1.28). The combination of both vertical directions was the fastest one (M=23.13, SD=1.76) with an average improvement of 4.05 seconds (≈ 15%) compared to all other combinations of device and interface orientations.

As there was no significant effect or interaction for *Interface Direction* we decided to use bottom to top as it is the fastest one when both the device and the interface are oriented vertically (M=21.23, SD=1.16). On average, participants were 4.7 seconds (≈ 18%) faster when using this direction compared to horizontal movements. In general, H1 is confirmed as the *Vertical Device Orientation* in combination with the *Vertical Interface Orientation* does not lead to occlusion effects.

When analyzing the error rate we found a significant main effect for *Device Orientation* ($F_{1,10} = 6.139$, $p < 0.033$) but no significant interaction effects. Post-hoc multiple means comparisons revealed that the *Vertical Device Orientation* (M=0.53, SD=0.15) performs better than the *Horizontal* one (M=0.86, SD=0.14). Considering low error rates and short task times, H3 is supported by our results.

Discussion

The higher error rates and task times for conditions where occlusion was a problem for participants can be explained as follows: (1) participants touched an item which caused

recommended items to appear beyond the display's boundaries and were thus invisible. (2) They put their finger to the far end of the screen to reach those as fast as possible. (3) Participants then had to precisely pick the moment when the items appeared, but sometimes still selected the wrong item, which increased the error rate. Furthermore, the task time got higher as they had to deselect the item and select the correct one. For the final design of rush, this implies separating interaction into getting recommendations for an item and selecting an item.

Another effect we observed in the study was that participants were either not aware of all suggested items or had to pan orthogonally to the *Interface Direction* to see all of them. This caused frustration among our participants. Hence, in the final design of rush, we decided to restrict panning to one dimension and only show one set of recommendations at one time.

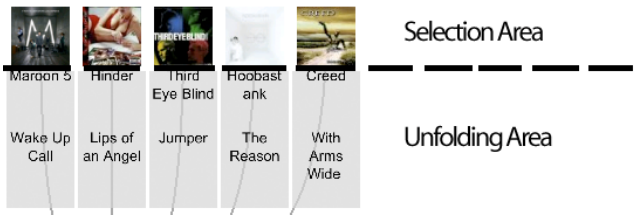

Figure 5. Representation of items

RUSH: DESIGN

The results of the pre-study led to the final design of the rush interaction technique (see Figures 1 and 2) with vertical device and interface orientation and interaction direction from bottom to top.

Touch interaction and Crossing-based interfaces

In the final version, one finger is still sufficient for navigating the complete item set and select items. The distance from the screen's center determines the speed, the angle the direction of movement (but as mentioned above, only along the vertical dimension). This movement is indirect and caused by a sliding of the underlying canvas into the opposite direction. After launching the application and choosing a seed item from a list or entering it manually, it is displayed in the center of the screen.

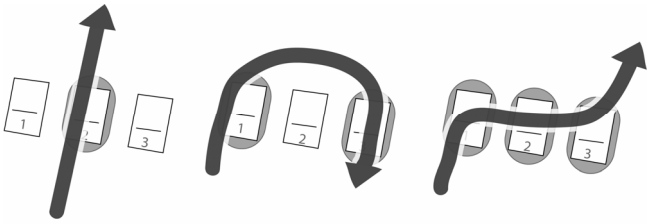

Figure 6 Selecting multiple items with one stroke

The visual representation of items is separated into two areas (see Figure 5): One area triggers the display of recommended items ("unfolds" the item) while the other one can be used to select the item for the result set. Also, the items are no longer squares but rectangles and aligned with the movement direction, making it harder to erroneously select them by drawing a complete line.

As soon as the user touches an item, recommendations are presented, but the item is not selected until a full line is drawn through it. Accot and Zhai have shown in [1] that continuous crossing-based interaction is comparable in performance to pointing-based alternatives. As the user's finger is on the screen anyway, crossing-based selection is an obvious choice for rush: the finger on the touch-screen not only causes navigation on the item plane, but also produces an (invisible) line that can be used for selecting items. In addition to that, continuously drawing a line contains additional information: the user is, for example, able to select multiple items in a row by simply drawing a longer line through them instead of repeatedly lifting, aiming and lowering the finger (see Figure 6). Also, instead of just hitting a single (more or less random) point within an item, a crossing line has an entering and exiting side which also can be used as a way for "richer semantics" [1]: We decided to minimize the number of erroneous de-selections by coupling the interface direction with the crossing direction: Drawing a line from bottom to left, top or right (along the interface direction) selects an item, while drawing a line from top to bottom (against the interface direction) deselects it.

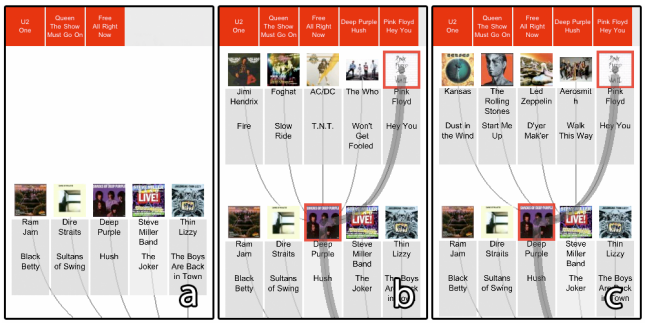

Figure 7. On-screen item layout while unfolding and selecting: a) initially, only row 1 is visible. b) After unfolding and selecting two songs from rows 1 and 2. c) After unfolding a different song from row 1 the recommendations in row 2 change.

We found that while in theory a complete interaction process can be started and ended by putting the finger down and up again, it is strenuous for users to keep their fingers pressed to the screen. To end the process, the user can alternatively wait for a short time to allow a dialog to pop up asking him if he wants to quit or shake the device, which can be detected by the integrated accelerometers and is sufficiently diametric to the regular interaction to not be triggered unintentionally.

As rush is intended for building collections of items, providing an overview of recently selected items is necessary to reduce the cognitive load and prevent the necessity to memorize recent decisions. The alternative of going back and following the trail of previous choices is time-consuming, so we preferred the alternative of using a portion of the screen space to display these recent choices. Similar to the version used for the pre-study, we used the top of the screen to display a textual representation of the last five selected items (see top of Figure 7).

Recommendation sets

By crossing either of the two areas with his finger, the item is unfolded and a set of recommended items is displayed. When thinking about the purpose of rush, the number of recommended items is crucial: As an information reduction technique, the number of items reflects the trade-off between freedom of choice and time spent deciding and browsing (cf. [29]). A small number of suggestions heavily restrict the possible choices, while a large one increases the time necessary for each decision. As every item has to be visually scanned, this time increases linearly with the number of items. The unsorted items allow no subdivision, so logarithmic decisions as in the Hick-Lyman Law cannot be applied [15]. Additionally, the available screen space is a restricting factor: To display many items they either have to become very small (and possibly unreadable) or disappear beyond the screen borders (making panning necessary to reach them). We decided for five items in our implementation as a compromise between choice and convenience. The participants of our second user study (see below) generally appreciated this choice (58% of them said that the number of suggestions was neither too high nor too low). The new set of suggestions appears in a row above the original item, using the available screen space as well as possible (see Figure 7).

One additional decision was how to handle a change in selection of the original item. With every row containing five items, five corresponding sets of recommendations are available. While in the regular case only one of these sets will be required (the one building on the selected item), the user is free to access the other sets as well. Displaying all 25 available items would lead to a large panning overhead and make it almost impossible to gather which item originated from where, so only one set of recommendations is available at one time. If a different item is unfolded, all unselected items from the last visible set are hidden and available spots are filled with new recommendations (see Figure 7b) and c)). Already selected items from the old set stay put. Every time an item is touched its recommendations are displayed, which means that when selecting multiple items using a single stroke only the last item's recommendations are visible afterwards.

An additional issue is the arrangement of recommended items, as the horizontal dimension (i.e., the order of the five items) can be used to encode additional information. We used the similarity of the recommended items for that and placed in one setup (*Hybrid*, see user study) the most similar item in the middle of the screen, next to two items that were reasonably similar but not too much and finally two items with a very low similarity as a way to "break out" of a certain direction. With this layout, the user is able to replicate the work of an automatic playlist creator based on similarity such as [26] by drawing a straight line up, thus always selecting the most similar item.

Sorting the items based on the probability returned by the recommendation engine allows users to have a clear conception about the relevance and changing their visual scanning depending on the current task. We compared the hybrid layout to one displaying the top five items in our study and found that chances were high that users had no way to maneuver out of a certain direction with the latter. In the hybrid layout, we circumvent the common "more of the same" problem of recommender systems and allow serendipity ([12], [5]) - but, of course, only if the user chooses to.

IMPLEMENTATION

For our second user study, we chose the domain of playlist creation. We implemented rush for the Apple iPhone 3G which has a 3.5" touch screen with a resolution of 320 x 480 pixels. We used the iPhone SDK and Objective-C for the implementation and OpenGL ES 1.1 for drawing. All songs and their relations were saved in a 10 Megabyte SQLite database directly on the device. Also, album covers or, if not available, artist photos were deployed as JPEG images together with the application to increase the loading speed. We wanted to allow participants to listen in on songs to make it easier for them to decide whether they fit in the current context or to help them recall a song if cover, artist and title name are not enough. For this purpose, we streamed 30 second samples from a web server through the phone's wireless LAN connection, which decreased the application's performance slightly but was received favorably by our study's participants.

USER STUDY

Rush is in the middle of the spectrum between fully automated and manual. Naturally, this hybrid approach causes longer task times than a fully automatic one (which effectively takes no time at all). We further assumed that the results were better in terms of quality. Our expectation for the manual approach, however, was that it produces the highest quality but is by far the slowest technique.

In a user study we investigated whether our assumptions were correct. We wanted to examine how well rush performed compared against automatic and manual playlist construction and what influence the choice of suggestion sets had on the user's satisfaction and performance.

Song Set Used in the Study

To give a realistic scenario and show that recommendation was indeed useful (i.e., browsing is not sufficient), we created a data set of 3900 songs, including samples and repre-

sentative images. Our goal for the data set was to create a collection that only includes songs which are commonly known. With this, participants were not confronted with completely unknown recommendations. The alternative of asking participants for sufficiently large song sets would have made the results less comparable. Therefore, we started with a manual selection of all time favorites from the genres rock, pop and R'n'B. Based on that, a script extracted similar songs from Last.FM[1]. We only considered songs that had been listened to at least 500,000 times, which made them sufficiently popular. For each song in the set, we created a list of similar songs (again based on Last.FM data). Songs with less than ten connections to songs by different artists were erased and we arrived at a final set of 3900 songs (from originally 4500).

While the similarity was based on Last.FM data, we added two constraints to improve the quality of resulting playlists. First, songs that were already in the playlist were not suggested again. Second, suggestions for a song did not have the same artist. Additional constraints on, for example, tempo or genre were not used.

Conditions, Task and Study Design

During the study, participants had to create four playlists: three using rush in different conditions and one manually. The three different versions of rush were identical regarding the interface, but the approach to recommendation changed: The *Top 5* condition presented users with the five most similar songs for an item. The *Random* condition took five songs at random from the list of similar songs. The already mentioned *Hybrid* condition presented the top similar song, two songs from the middle of the similarity list and two songs from the bottom. In the manual condition, participants had access to a web browser on a desktop PC with a list of all 3900 available songs. In order to keep the results comparable, participants were able to see the (full) list of similar items for each song. They also had the option to listen to samples of those. The last set of conditions was automatic playlists, created out of the 3900 songs. The playlist generator replicated the user's choices in rush but picked a random song from the five suggestions. In the automatic condition, there were also three suggestion strategies (*Top 5*, *Random*, and *Hybrid*).

In all conditions the task was the same: (1) participants initially chose a seed song which was the same in all conditions. (2) Starting from this song, they had to construct a playlist with ten songs. We asked the participants to create a playlist for other people (e.g., a social event) to make them think about what constitutes a good playlist.

We used a within-subject study design. Our independent variable was the used *Tool* with 7 factors: *Manual, Rush Top 5, Rush Random, Rush Hybrid, Automatic Top 5, Automatic Random,* and *Automatic Hybrid*. Prior to the study, participants were allowed to get comfortable using the system. The order of the three rush conditions was counterbalanced across our participants and the automatic ones were created in the background during the study. When they completed the playlists using the rush conditions, participants had to manually create another playlist. We measured the completion time for each of the rush tools and the time spent using the manual condition.

In the beginning of the study, participants chose their seed song. They then created the playlists using each rush condition. However, after each constructed playlist, they had to fill out a questionnaire on how useful they rated the suggestions and how random they appeared. They then built their final one manually. In the end, participants had to fill out a post-questionnaire with a modified set of the IBM Computer Usability Satisfaction questions and statistical data. Also, they were asked to rank the rush and manual tools (as the automatic versions allowed no interaction) and the resulting seven playlists.

Participants

We recruited 12 participants for our second user study (4 female, 2 left-handed, 4 had participated in the pre-study). All participants declared they had experience with touch screens. Their age ranged from 24 to 35 (average: 28 years).

Hypotheses

We had three hypotheses: playlists can be constructed fastest using the automatic tools, the slowest using the manual version. Rush takes a time between the two (H1). The quality of the resulting playlists is higher with rush than the *Automatic* conditions (H2). And, *Rush Hybrid* gives better results and is preferred to *Rush Random* (H3).

Results

The creation time of automatic is, of course, always 0 seconds. The average times for rush were 123.8 (*Rush Top 5*), 142.1 (*Rush Hybrid*), and 162.3 seconds (*Rush Random*). The manual condition – as expected the slowest one – had an average time of 388.6 seconds. This supports our hypothesis H1.

The participants were overall satisfied with rush's usability as the operation speed was the only point of criticism (average ranking of 2 on a 5-point Likert-scale where 1 translates into "too slow"). Analyzing the tool's quality ranking using the Condorcet Ranked-Pairs system reveals the *Manual* as the winning candidate (3 wins), followed by *Rush Hybrid* (2 wins, 1 loss), *Rush Random* (1 win, 2 losses) and *Rush Top 5* (3 losses).

Measuring the quality of playlists is hard ([4],[3]), as the results are always personal and thus should be evaluated by their creators only. On the other hand, we would add a bias because participants are expected to rank the playlists they were involved with better and the automatic playlists worse.

[1] Last.FM is a platform for tracking listening behavior and based on this data, similarity values are created by collaborative filtering.

This "emotional bond" can be explained by the fact that participants would generally rank playlists better if they had spent time on their creation. The "novelty effect" could further explain this bias. Thus – to learn about the impartial playlist quality – we started an online questionnaire where everyone was asked to rank a random set of playlists from the study. We received 10 ranked sets of playlists.

An analysis of the study participants' rankings using the Condorcet Ranked-Pairs system showed that playlists built manually were clearly favored (6 wins), followed by *Rush Hybrid* (5 wins, 1 loss), *Rush Random* (4 wins, 2 losses), *Automatic Random* (3 wins, 3 losses), *Rush Top 5* (2 wins, 4 losses), *Automatic Top 5* (1 win, 5 losses), and *Automatic Hybrid* (6 losses). As expected, the online participants had different opinions: *Rush Hybrid*, *Rush Random* and *Automatic Hybrid* are tied for first place (4 wins, 2 unresolved), followed by *Automatic Random* (3 wins, 3 losses), *Manual* (2 wins, 4 losses), *Rush Top 5* (1 win, 5 losses), and at the last position *Automatic Top 5* (6 losses).

Our conclusion is that the participants were clearly biased towards their own playlists and thus ranked the automatic results negatively. The more independent online vote shows that the *Hybrid* and *Random Automatic* and *Rush* versions yielded better results than the *Manual* and *Top 5* versions. We suppose these results stem from the participants' lack of experience in playlist building: Thus, adding recommendations helped to improve the quality of produced playlists, but only if the suggestions were not too restricting (as in the *Top 5* versions). The freedom that participants gained from the manual version had the downside of reducing the quality. Restricting the participants' choices might decrease the tool's satisfaction but actually helps them in producing (objectively) better results. Thus, *Rush Hybrid* and *Rush Random* brought the overall best results in subjective and objective quality. Unfortunately, we were not able to confirm H3, which means that a random set of suggestions and the more elaborate hybrid set ranked equally well.

DISCUSSION

Rush's flexibility is inherently restricted: The convenience of only having to choose between five and not all items of a collection can also be seen as the limitation of only *being able* to choose between five items. As the second study showed, depending on the underlying recommendation engine rush can yield very different results. In general, the suggestions by rush can be local, i.e., personalized for the user with one of the various recommendation techniques like collaborative filtering (for an overview see [7]). Yet, with rush being an (interactive) recommendation technique itself, suggestions can also be global (identical for each user) and thus based on, for example, a similarity metric. While we used the second approach in the user study to keep the results comparable, we suppose that introducing personalized suggestions might improve the user experience.

Shaping Recommendations

Depending on the use case several adaptations of the recommendation engine might be useful. First of all, chosen items can be interpreted as votes, thus adapting the user profile while he interacts with the system. Every item that is chosen receives a positive rating, while other items from the same set are downgraded, thus refining the adaptation to the user. The downside of this approach is the growing restriction in suggestions, with diminishing serendipity being a common problem of recommender systems [12].

Second, constraints might be applied to the set of suggestions depending on the use case. In addition to the design space for recommending collections proposed by Hansen et al. [11] we suggest two main categories based on the time frame of consumption: Items in a *concurrent* collection are consumed at the same time (e.g., apparel, extras for a car or a hotel room). Items in a *sequential* collection are consumed sequentially (e.g., a song playlist, travel plans, dinner courses). The type of collection leads to different constraints: For concurrent collections, order is not applicable but all items have to work together all the time. For sequential collections, internal consistency is important as well, but can be alleviated by a clever use of sequence. This sequence, on the other hand, adds additional constraints in that sequential items have to work after one another.

Finally, for certain user tasks and requirements, additional rule-based constraints might be added to the generation of suggestions. One such rule might be that a playlist should not contain songs twice or two songs by the same artist in a row. Also, constraints like "two sequential songs have to be similar in tempo or rhythm" might be applied. Such constraints can be used to minimize the number of inappropriate suggestions that otherwise take up one of the five precious slots. But they can also help an inexperienced user (who does not know these rules that guarantee quality) produce suitable results.

Steerable Recommendations

Rush provides users with "steerable" recommendations: The choice of an item not only makes it part of the current selection but also shapes the form of the newly suggested set as these are related to it, thus guaranteeing a certain coherence of the resulting set.

The major advantage of making recommendations interactive is the flexibility that it brings. First of all, the user is able to shape the results of the recommendation process to his liking at every step of the process. Items that the recommendation engine might find suitable but the user clearly knows he does not like can immediately be skipped and the recommendations can also be adapted to the current mood of the user - something that recommendations based on a user profile are not able to do without additional questioning. But flexibility is not only restricted to this local level: The sum of all these small decisions, the resulting collection, can also be actively adapted to the likings of the user: While other tools allow setting a general mood or tone of the items [25], rush adds a temporal flexibility to that: Depending on the underlying

recommendation strategy, musical playlists that start with a certain mood and change over time in a gradual buildup are possible. Also, there is no predefined length of the collection, which lets users add items as long as they like. Lastly, the user is also flexible in its interaction with the tool: Depending on temporal constraints, the user can quickly finish building a collection, spend some more time exploring the recommendations and choosing more deliberately, or switch between the two whenever appropriate.

In addition to flexibility, rush provides users with an easy way to generate collections: Constraint-based recommender systems might also be able to yield similarly complex results but demand more mental effort from users: All constraints have to be known beforehand and expressing, for example, a gradual buildup in mood in a music playlist might be hard to express in the underlying constraint language. Also, a spontaneous change of plan is only possible by repeating the complete process, while rush users have no problem with adapting their preferences in the middle of the process. Finally, rush has the "I know it when I see it" advantage: Users can decide what they want on the go and listen to their gut feeling instead of having to decide and rationalize beforehand (cf. [16]).

Conclusion

We have presented rush, an interaction technique for creating item collections on mobile devices. An underlying recommendation algorithm decreases the user's options and thus makes it easier to build suitable collections. Still, compared to fully automatic recommendation, the user is able to influence the resulting set in a less complicated way than with constraint-based recommenders. One-touch and crossing-based interaction makes rush suitable for mobile use. Our studies showed that the vertical orientation performed best for interface and device. We also found that the choice of suggestions had a strong influence on the user's liking of the system and the quality of the results. Too restricting suggestions should be avoided in order not to frustrate the users (as one participant put it: "There are always the same songs!"). We evaluated rush in a playlist scenario but the technique itself is applicable to various recommendation tasks.

An alternate use of rush might lead to a single item selection instead of that of a whole collection: Using a "navigation by proposing" [31] approach, each row of suggestions and succeeding choices can be interpreted as a vote, thus more and more restricting the search space. We plan to evaluate this version as well.

It might also be interesting to see if a different type of interaction (e.g., panning by finger flicking, selection by tapping) has an influence on user performance.

Lastly, we plan to evaluate rush in a real-world mobile context and not just the laboratory.

ACKNOWLEDGMENTS

We thank the state of Bavaria for funding. We would also like to thank the participants of our studies, Alexander De Luca for helping with the evaluation and Petteri Nurmi for valuable feedback on the paper.

REFERENCES

1. Accot, J. and Zhai, S. More than dotting the i's - foundations for crossing-based interfaces. *Proc. SIGCHI conference on Human factors in computing systems*, (2002), 73-80.

2. Adomavicius, G. and Tuzhilin, a. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (2005), 734-749.

3. Andric, A. and Haus, G. Estimating Quality of Playlists by Sight. *First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'05)*, (2005), 68-74.

4. Andric, A. and Haus, G. Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools and Applications* 29, 2 (2006), 127-151.

5. André, P., Teevan, J., and Dumais, S. From x-rays to silly putty via Uranus: serendipity and its role in web search. *Proc. 27th international conference on Human fac-tors in computing systems*, ACM (2009), 2033–2036.

6. Aucouturier, J. and Pachet, F. Scaling up music playlist generation. *Proc. IEEE International Conference on Multi-media and Expo*, IEEE (2002), 105–108.

7. Burke, R. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction* 12, 4 (2002), 331–370.

8. Cöster, R. and Svensson, M. Incremental collaborative filtering for mobile devices. *Proc. 2005 ACM symposium on Applied computing - SAC '05*, ACM Press (2005), 1102.

9. Goldberg, D., Nichols, D., Oki, B., and Terry, D. Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 61, 10 (1992), 1-10.

10. Goussevskaia, O., Kuhn, M., and Wattenhofer, R. Exploring Music Collections on Mobile Devices. *Proc. 10th international conference on Human computer interaction with mobile devices and services*, ACM (2008), 359-362.

11. Hansen, D.L. and Golbeck, J. Mixing It Up: Recommending Collections of Items. *Proc. 27th international con-ference on Human factors in computing systems*, ACM (2009), 1217-1226.

12. Herlocker, J.L., Konstan, J.A., Terveen, L.G., and Riedl, J.T. Evaluating Collaborative Filtering Recom-mender Systems. *Transactions on Information Systems (TOIS)* 22, 1 (2004), 5-53.

13. Jacobsson, M., Rost, M., and Holmquist, L.E. When Media Gets Wise: Collaborative Filtering with Mobile Media Agents. *World Wide Web Internet And Web Information Systems*, (2006).

14. Kim, C.Y., Lee, J.K., Cho, Y.H., and Kim, D.H. VISCORS: A Visual-Content Recommender for the Mobile Web. *IEEE Intelligent Systems* 19, 06 (2004), 32-39.

15. Landauer, T. and Nachbar, D. Selection from alphabetic and numeric menu trees using a touch screen: breadth, depth, and width. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM New York, NY, USA (1985), 73–78.

16. Lehrer, J. How We Decide. Houghton Mifflin, New York, 2009.

17. Lehtiniemi, A. and Seppänen, J. Evaluation of Automatic Mobile Playlist Generator. *Proc. Mobility*, (2007), 452-459.

18. Miller, B., Konstan, J., and Riedl, J. Pocketlens: Toward a personal recommender system. *ACM Transactions on Information Systems* 22, 3 (2004), 437–476.

19. Miller, B.N., Albert, I., Lam, S.K., Konstan, J.A., and Riedl, J. MovieLens Unplugged: Experiences with an Oc-casionally Connected Recommender System. *Proc. IUI*, (2000), 263-266.

20. Neumayer, R., Dittenbach, M., and Rauber, A. Playsom and pocketsomplayer, alternative interfaces to large music collections. *Proc. ISMIR*, (2005), 618–623.

21. Nguyen, Q. and Ricci, F. Long-term and session-specific user preferences in a mobile recommender system. *Proc. 13th international conference on Intelligent user interfaces*, ACM New York, NY, USA (2008), 381–384.

22. O'Donovan, J., Smyth, B., Gretarsson, B., Bostandjiev, S., and Höllerer, T. PeerChooser: Visual Interactive Recommendation. *Proc. twenty-sixth annual SIGCHI conference on Human factors in computing systems*, (2008), 1085-1088.

23. Pampalk, E., Pohle, T., and Widmer, G. Dynamic Playlist Generation Based on Skipping Behaviour. *Proc. ISMIR*, (2005), 634-637.

24. Pauws, S. and Eggen, B. PATS: Realization and user evaluation of an automatic playlist generator. *Proc. 3rd International Conference on Music Information Retrieval*, (2002), 222–230.

25. Pauws, S. and Wijdeven, S.V. User evaluation of a new interactive playlist generation concept. *Proc. ISMIR*, (2005).

26. Pohle, T., Pampalk, E., and Widmer, G. Generating similarity-based playlists using traveling salesman algorithms. *Proc. 8th International Conference on Digital Au-dio Effects (DAFx-05)*, (2005), 1-6.

27. Pu, P. and Chen, L. Trust building with explanation interfaces. *Proc. 11th international conference on Intelli-gent user interfaces*, ACM Press (2006), 93.

28. Ragno, R., Burges, C.J., and Herley, C. Inferring similarity between music objects with application to playlist generation. *Proc. 7th ACM SIGMM international workshop on Multimedia information retrieval*, ACM Press New York, NY, USA (2005), 73-80.

29. Schwartz, B. The paradox of choice: Why more is less. Harper Perennial, New York, 2005.

30. Smyth, B. and Cotter, P. Personalized adaptive navigation for mobile portals. *Proc. ECAI*, (2002), 608–612.

31. Smyth, B. Case-based recommendation. *In Lecture Notes in Computer Science*. Springer, Berlin / Heidelberg, 2007, 342-376.

32. Swearingen, K. and Sinha, R. Interaction design for recommender systems. *Proc. of Designing Interactive Systems*, ACM (2002).

33. Tintarev, N. and Masthoff, J. Effective explanations of recommendations: user-centered design. *Proc. 2007 ACM conference on Recommender systems*, ACM New York, NY, USA (2007), 153–156.

34. Vig, J., Sen, S., and Riedl, J. Tagsplanations: explaining recommendations using tags. *Proc. 13th international conference on Intelligent User Interfaces*, ACM (2009), 47-56.

35. Vogel, D. and Baudisch, P. Shift: a technique for operating pen-based interfaces using touch. *Proc. SIGCHI conference on Human factors in computing systems*, ACM (2007), 657-666.

36. Ward, D.J. and Blackwell, A.F. Dasher---a data entry interface using continuous gestures and language models. *Proc. 13th annual ACM symposium on User interface software and technology - UIST '00*, ACM Press (2000), 129-137.

37. Ward, D.J. and MacKay, D. Fast Hands-free Writing by Gaze Direction. *Nature* 838, August (2002), 4-6.

38. Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., and Shen, C. Lucid touch: a see-through mobile device. *Proc. 20th annual ACM symposium on User interface software and technology*, ACM (2007), 269-278.

39. van Gulik, R. and Vignoli, F. Visual playlist generation on the artist map. *Proc. ISMIR*, (2005), 520-523.

40. van Gulik, R., Vignoli, F., and van De Wetering, H. Mapping music in the palm of your hand, explore and dis-cover your collection. *Proc. ISMIR*, (2004).