Exploring Personalized Command Recommendations based on Information Found in Web Documentation

Md Adnan Alam Khan, Volodymyr Dziubak, Andrea Bunt
Department of Computer Science
University of Manitoba
Winnipeg, MB
{akhan, vdziubak, bunt}@cs.umanitoba.ca

ABSTRACT
Prior work on command recommendations for feature-rich software has relied on data supplied by a large community of users to generate personalized recommendations. In this work, we explore the feasibility of using an alternative data source: web documentation. Specifically, our approach uses QF-Graphs, a previously proposed technique that maps higher-level tasks (i.e., search queries) to commands referenced in online documentation. Our approach uses these command-to-task mappings as an automatically generated plan library, enabling our prototype system to make personalized recommendations for task-relevant commands. Through both offline and online evaluations, we explore potential benefits and drawbacks of this approach.

Author Keywords
Feature-rich software; Software learnability; Recommender systems; Web-based documentation

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Feature-rich software applications (e.g., image-manipulation programs, spreadsheet software and statistical analysis packages) are highly versatile, in part owing to the hundreds (or even thousands) of commands or features that they make available. At the same time, this high volume of commands can make feature-rich software difficult for users to master [17]. For example, studies of long-term application use have shown that most users have fairly limited *command vocabularies* [14], typically using much less than one quarter of the available command set (e.g., [26, 30]).

IUI 2015, March 29–April 1, 2015, Atlanta, GA, USA.

ACM 978-1-4503-3306-1/15/03…$15.00.
http://dx.doi.org/10.1145/2678025.2701387

One promising approach to increasing a user's awareness of the available command set is to present them with intelligently generated, personalized command recommendations (e.g., [30, 31, 35]). Central to this approach is an understanding of the potential relationships between commands – knowledge that enables an intelligent system to recommend commands that could complement those currently being used. Most prior work has extracted this relevancy information from community usage logs, for example, by applying collaborative filtering algorithms to a large corpus on logged data (e.g., [30, 31, 35]). While these usage-data centric approaches have shown a great deal of promise, their practical success hinges on the existence of a large community of users willing to upload their usage data to a central repository.

In this work, we propose an alternative approach to personalized command recommendations that uses command-to-task mappings mined from online documentation. Specifically, our approach uses Query-Feature Graphs (QF-Graphs) [13], a technique that maps common web search queries to collections of interface elements referenced in the resulting online documentation. Within the context of the GNU Image Manipulation Program (GIMP), we illustrate how our prototype recommender system uses a QF-Graph as an automatically generated plan library.

Through offline experimentation with previously collected usage data, we explore this documentation-centric approach's potential to both increase a user's command awareness through task-relevant recommendations, and to enable a system to provide efficient access to needed commands. In a controlled laboratory experiment, we also explore the impact of two alternative recommendation presentation techniques on immediate task performance as well as on incidental awareness of relevant commands not selected during the primary task [10].

Our results suggest that web documentation can be leveraged to generate recommendations for commands that are relevant to the task at hand. In terms of its ability to streamline access to needed commands, our findings indicate that the approach works best for users with diverse feature usage. Frequency-based approaches, on

the other hand, achieve higher predictive accuracy for users with stable and homogenous usage patterns. The results of our laboratory evaluation reveal that presenting recommendations in-place, as opposed to in a separate palette, increases users' incidental command awareness without negatively impacting their immediate task performance.

The contributions of this work are as follows. We present a novel approach to intelligent command recommendation that uses web documentation as a knowledge source. We demonstrate potential strengths and weaknesses of this approach given a variety of command usage patterns. Finally, we present an initial empirical exploration of how such a system could display its recommendations within the interface.

RELATED WORK

Prior work on supporting feature-rich software use has focused on three main areas: supporting application learning through improved tutorials, providing in-application assistance, and techniques for streamlining access to needed commands. We describe each of these bodies of work below. We conclude the section with a brief description of techniques for mining web documentation for command-to-task mappings, one of which we apply in our work.

Improved Tutorial Interfaces

Prior work has shown that tutorials play a large role in application learning (e.g., [7, 25]), but that authoring effective tutorials can be difficult [32]. Consequently, prior research has focused on automated or semi-automated tutorial authoring (e.g., [5, 16]), and creating novel and engaging tutorial formats (e.g., [6, 8, 18, 29]). Work has also begun to explore ways to harness crowds or community contributions to improve the utility of web-based tutorials, for example, through integrated community refinements [3], by augmenting tutorials with community demonstrations [27], and by using crowds to help segment video tutorials into steps to permit easier tutorial navigation [23]. Finally, prior work has sought to make it easier for users to select appropriate tutorials, by highlighting the commands used [7, 24].

In-Application Assistance

As is the case with our work, prior research has also sought to improve software learning from within the application itself. One approach has focused on helping users understand how to use commands through, for example, intelligent task assistants [20], Q&A forums embedded within the interface [34] and video-based tool-tips [19].

Most relevant to our work are systems that provide unobtrusive personalized command recommendations by mining large corpuses of community usage data. For example, the OWL system used long-term usage histories from individuals within an organization to recommend commands that were underused in comparison to their peers. Similarly, CommunityCommands [30, 35] used collaborative filtering algorithms on a large corpus of usage data to generate personalized command recommendations. Patina, on the other hand, provided subtle command recommendations by overlaying heat maps on the interface to highlight commands commonly used by the community [33]. We extend this prior work by exploring a new data source for task-relevant command recommendations – web-based documentation. We also further explore the question of how to present these types of command recommendations within the interface.

Providing Efficient Access to Needed Commands

Given that users of feature-rich software tend to use only a small subset of the available command set (e.g., [26, 30]), prior work has examined ways to provide more efficient access to needed commands by reducing the search space. These approaches have ranged from user-customizable subset interfaces (e.g., [2, 36]), to community-authored task-specific interfaces accessible through in-application keyword search [28], to interfaces that adaptively promote commands according to recency and frequency information (e.g., [10, 15]). Our recommendation presentation techniques are informed by this body of work. We also extend this prior research by exploring a new way to compute relevant command subsets.

Extracting Command-to-Task Mappings from Online Documentation

Given the prevalence of online documentation for feature-rich software, prior work has explored the feasibility of using these resources to generate task-specific command groupings. For example, Fourney *et al.* [13] proposed Query-Feature Graphs (QF-Graphs) as a way to relate users' search queries for feature-rich applications to individual interface elements referenced in the resulting webpages. The very recent CommandSpace work also uses web documentation to model the relationships between tasks and commands, but does so using a vector-space representation as opposed to a graph [1]. We extend this work by systematically exploring the potential of these mappings to support personalized command recommendations.

PROTOTYPE DESCRIPTION

The core component of our system, which we refer to as QFRecs (see Figure 1), is the Query-Feature Graph that was originally introduced by Fourney *et al.* [13]. As illustrated in the simplified example in Figure 2, a Query-Feature Graph (QF-Graph) is a weighted bipartite graph, which associates user search queries (i.e, natural language descriptions of high-level tasks [26], Figure 2 a, left), with the features of a target application (Figure 2 a, right). The weight of the edge between a query node and a feature node represents their strength of association (see Table 1 for example weights).

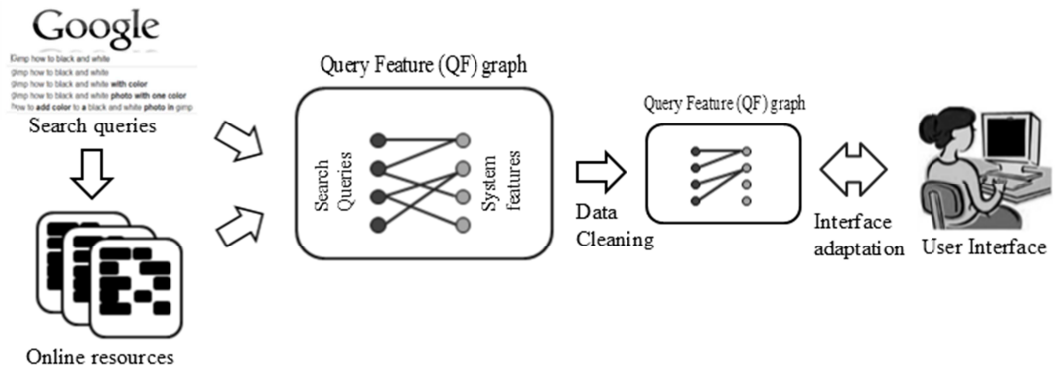

Figure 1. The general architecture for our prototype command recommender system QFRecs.

The process of building a QF-Graph starts with collecting common user search queries issued for a target application using an application of the CUTS method [12], which leverages the Google Suggest API. These queries are then executed and the resulting webpages are examined for occurences of application-specific features (using a list of features that can be generated semi-automatically using the application's localization data).

	Blur/sharpen	Blend	…	Color balance
how to blur image background in GIMP	9.33	6.82	…	0
…	…	…	…	…
Gimp make image black and white	0	0	…	8.77

Table 1. Example connection weights between tasks and features.

QFRecs's general architecture is depicted in Figure 1. As a starting point, we use the "raw" QF-graph generated by Fourney *et al.* [13] for the GNU Graphics Manipulation Program (GIMP). We focus on GIMP for two reason: it allows for greater modification possibilities than propietary software, and we are able to perform offline evalutations of our approach using data previously collected through the Ingimp project [26, 37]. In the next section we describe some of the steps that we took to clean this raw QF-Graph. We follow this by describing how our approach uses the cleaned QF-Graph to generate personalized command recommendations.

Data Preparation

In moving from the original QF-Graph concept to a concrete application, we discovered two sources of noise that impacted the quality of our prototype's recommendations requiring that we "clean" this original QF-Graph for it to be suitable for our purposes. As an overview, our cleaning process took part in two steps. First, we pruned tasks from the left-hand side of the QF-Graph that were not representative of high-level tasks. Second, since the goal was to recommend specific GIMP commands, we made sure that all system features on the right-hand side of the graph corresponded to actual elements in the GIMP interface. We describe these sources of noise and our methods for cleaning the data in further detail here to illustrate some of the challenges of using the document-based approach to command recommendation in practice.

In terms of the high-level tasks (see Figure 2 a, left for examples), manual exploration of 12,311 search queries used to build the original QF-Graph revealed that many queries were not actually representative of high-level tasks. Therefore, we removed queries from this original graph if they met any of the following criteria: they contained digits (e.g. "gimp review 2010", "gimp 2.6 fonts", etc.), operating system names (e.g. "gimp for mac

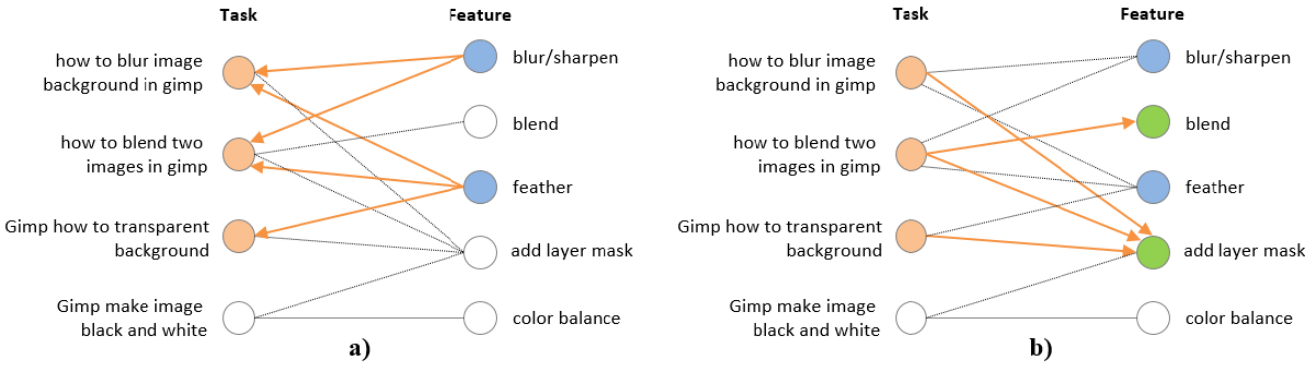

Figure 2. A simple illustration of the QF-Graph based approach.

OS x"), or the "vs" string (e.g. "gimp vs adobe", "adobe illustrator vs gimp"). This left us with 9889 queries. An altnerative would have been to restrict the search queries to those containing the string "GIMP how to", however, this strategy appeared to result in unnecessary information loss (e.g., it removed over 90% of the queries).

The second source of noise concerned the features themselves (see Figure 2 a, right for example features in a QF-Graph). We found that not all the features extracted from the Web documentation mapped to commands in the actual GIMP interface. The primary cause was minor textual differences in labelling (e.g., "by color select" vs. "select by colour"). Most differences were resolved automatically by string matching using regular expressions. The matching rules for the regular expressions were crafted manually based on actual command names from the GIMP menus. After this process, there remained 40-50 unmapped features (of the original 617 distinct features in the raw QF-Graph). For these remaining features, the mapping was done manually based on our knowledge of the target application. Manual inspection also revealed that a feature's parent entity (e.g., menu name) was sometimes present in the raw data in addition to the command itself. This is because Web documentation will often specify a command's full path. For example, the line "Tools > Paint Tools > Paintbrush" would lead both "Tools" and "Paint Tools" to appear in the graph. We removed such menu names from the graph.

Generating Recommendations

To generate personalized, contextually relevant command recommendations, QFRecs uses the cleaned QF-graph as an automatically generated plan library. Based on a user's last *x* command selections (i.e., the history size), QFRecs "activates" the corresponding nodes in the graph (i.e., the recently used features/commands). Our system currently uses the last 5 distinct observed commands for this initial activiation phase, however, this history size is a configurable parameter. Larger history sizes will mean recommendations tailored more to the user's overall usage than their current context. After the the user's last *x* commands are activiated, QFRecs "activites" possible task (query) nodes based on their connection weights. This step amounts to estimating which of the tasks in the graph are most likely to be the user's current task. Using these estimations, QFRecs then "activates" other relevant commands for those candidate tasks.

This process is illustrated in Figure 2. In this example, the history is size 2 and the last two observed commands are "Blur/Sharpen" and "feather". QFRecs first finds the set of tasks that are strongly associated those two features (see the left-hand nodes in Figure 2 a) using the edge weights in Table 1. In the next step, QFRecs uses those strongly associated tasks to isolate other features (see the right-hand nodes in Figure 2 b) associated with those tasks. These features are then ranked according to the summed weights of all of their associated tasks activated in the previous step, enabling the system to recommend the top *k* features. In this small example (k = 2) QFRecs recommends "blend" and "add layer mask" to the user.

Recommendation Types

Using the process described above, the recommended commands will be one of the following two types:

1. *"Familiar" Recommendations:* These are commands in the user's existing command vocabulary that are predicted to be most relevant to the current usage context.
2. *"Unfamiliar" Recommendations:* These are contextually-relevant recommendations for commands that are not yet in the user's command vocabulary.

These recommendations serve different purposes. "Familiar" recommendations will not introduce users to new commands, but if promoted effectively, have the potential to improve task effeciency. "Unfamiliar" recommendations, on the other hand, have the potential to enhance feature awareness. That this method is capable of generating both types of recommendations raises a number of interesting interface presentation questions, which we begin to explore in our laboratory evaluation. However, we first explore the accuracy and potential utility of these document-based recommendations using offline analysis.

OFFLINE EVALUATION

We evaluated our QF-Graph approach on a corpus of GIMP usage data that was collected as part of the Ingimp project [26, 36]. This corpus contains feature usage histories (or logs) from 207 users, collected over a period of approximately two years. We used this data to evaluate our prototype along two dimensions. The first was its ability to generate relevant recommendations, to gain an initial understanding of the approach's potential to improve users' feature awareness. We also evaluated the approach's accuracy in predicting a user's next command selection, comparing our approach to frequency- and recency-based prediction algorithms (e.g., [9. 15]).

Potential to Promote Awareness of Relevant Commands

In assessing the approach's potential to make users aware of new commands, we first examined (according to the definitions above), how many of the generated recommendations could be classified as "unfamiliar". Figure 3 illustrates the mean breakdown of the recommendations into the two types when QFRecs generates 5, 10, 15 and 20 recommendations (Recommendation Size in Figure 3). These results indicate that the QF-based approach tends to favor "unfamiliar" recommendations, particularly as the number of recommendations increases.

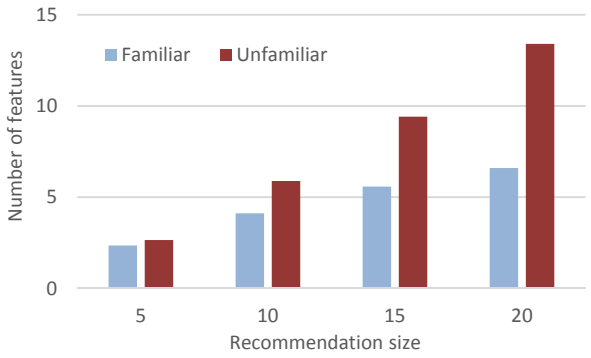

Figure 4. The mean number of familiar features and novel unfamiliar features for different recommendation sizes.

As a measure of recommendation relevance, we used a modified version of the K-tail evaluation method, introduced by Li *et al.* to evaluate their CommunityCommands recommender system [30, 35]. A K-tail evaluation divides a series of used features F into two sets: a training set (F_{train}) and a test set (F_{test}), such that the test set F_{test} contains k distinct features which are not in F_{train}. The training set is then used as the user's history and the prediction algorithm's performance is measured according to how well it predicts those k-distinct features in F_{test}. As an initial measure of the relevance of the "unfamiliar" recommendations to the user's current usage context, we adapted this evaluation method as follows: We measured whether or not our system's recommendations predict at least one new feature in the next k feature invocations (i.e., whether or not at least one "unfamiliar" recommendation appears in the next k feature invocations).

Figure 4 depicts our results for a range of recommendation and tail sizes. With a recommendation size of 20, QFRecs achieves a k-tail "accuracy" that is up to 80%. We also analyzed the k-tail accuracy of QFRecs's "familiar" recommendations, with Figure 5 illustrating similar trends.

To provide further insight into the relevance of the novel command recommendations, we also examined how many of the "unfamiliar" recommendations are "accurate" according to the above definition. Table 2 illustrates that, on average, 3 - 9% of "unfamiliar" recommendations in a given set appear in the user's next k commands. The table also illustrates a large variability in accuracy, with up to 80% of the recommendations appearing in the user's next k selections. In some respects, the results in Figure 4 and Table 2 represent a lower bound on the relevance of the system's recommendations; users may be failing to use certain features not because of lack of relevance, but because of lack of awareness.

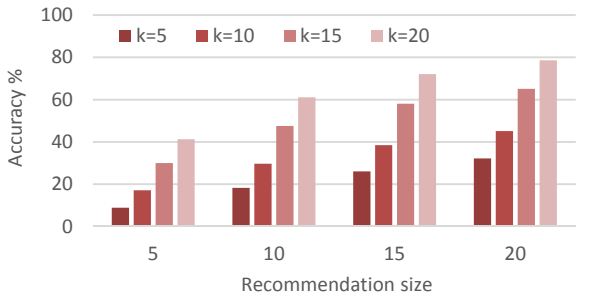

Figure 3. Performance of our QF-based approach recommending an "unfamiliar" feature that is then used in the next k feature invocations.

Combined, these results suggest that the QF-based approach (as currently instantiated in our QFRecs system) is able to recommend at least some relevant commands. The results also suggest that if promoted effectively within the interface, the recommendations could potentially aid users in completing their current task. We explore this question of presentation in our laboratory study and describe potential ways to improve the recommendations in the Discussion.

Predicting the Next Command

The above results provide an initial indication that the QF-based approach does generate some user-relevant recommendations in that they appear later in the user's command stream. In this section, we explore the approach's potential to immediately streamline access to needed commands, by analyzing the degree to which the recommended set accurately predicts the next command in the stream. We also compared our QF-based approach with two algorithms commonly used in prior work on adaptive interfaces (e.g. [9, 15]): frequency-based predictions, and recency-based predictions. In this case, we defined an "accurate" prediction as one where the user's next action is within the recommended set of commands.

When considering all users' data, we found that the frequency-based algorithm dramatically outperformed the others when it came to predicting the user's next command (see Figure 6). When examining the reasons why, we found that the users in this particular dataset tended to have very homogenous and stable command

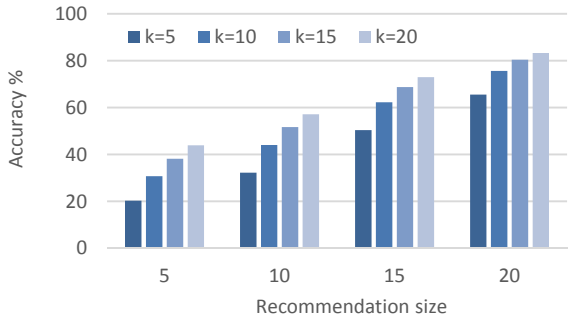

Figure 5. Performance of our QF-based approach in recommending a "familiar" feature that is then used again in the next k feature invocations.

	K = 5			K = 10			K = 15			K = 20		
	Accuracy %	Max %	Standard deviation	Accuracy %	Max %	Standard deviation	Accuracy %	Max %	Standard deviation	Accuracy %	Max %	Standard deviation
R = 5	3.3	60	10.79	5.79	80	11.95	6.68	80	12.75	7.25	80	13.76
R=10	3.85	40	6.11	5.81	50	7.54	7.3	50	8.46	8.53	50	9.13
R=15	4.39	33.33	5.05	6.39	40	6.29	7.68	40	7.07	8.7	40	7.64
R=20	4.52	25	4.07	6.38	30	5.1	7.84	35	5.76	9	35	6.24

Table 2. Average and maximum percentage of correct recommendations for each tail (k) and recommendation set size (R).

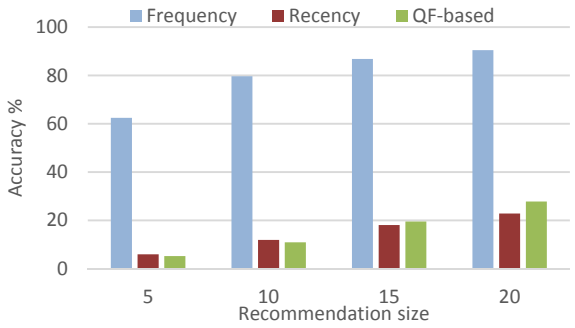

Figure 6. The overall accuracy in predicting the next command of the Frequency-based, Recency-based and QF-based approaches for different recommendation sizes.

usage, which naturally favors the frequency-based approach. As an example, consider a feature invocation sequence of length 149, but that consists of only 10 distinct features. With a recommendation size of 10, once these 10 features are observed, the algorithm will never fail.

To better characterize and explore the dataset, we defined a user's feature usage diversity, R_d, as follows

$$R_d = \frac{Number\ of\ Distinct\ Features\ Used}{Sequence\ Length}$$

In this dataset, the mean R_d for all 178 users with usage sequences longer than 20 is 0.1668 with a standard deviation of 0.1329. The max(R_d) was 0.6666 (sequence length = 42 and number of distinct commands = 28) and min(R_d) was 0.0043 (sequence length = 23459 and number of distinct commands = 101). Whereas the frequency-based approach substantially outperformed the QF-Graph approach for users with low feature diversity, the results are much more promising for users with high feature diversity. As an example, Figure 7 compares the accuracy of the different algorithms for a user with near mean feature diversity (0.1667). In this case, the QF-based approach actually outperforms the alternatives when the recommendation size is 5.

A second potential downside of the frequency-based approach is that it requires usage patterns to stabilize before it can be effective. For example, Figure 8 compares the frequency-based approach's accuracy over its first 30 feature invocations to its overall performance. Our QF-based approach, on the other hand, requires less start-up time; it is currently set to generate recommendations based on the last five commands.

Thus, to summarize, we found that a frequency-based or any history-based approach is sensitive to the usage diversity and session length. While the QF-based approach may not be an effective predictor when faced with long, homogenous sessions, it has advantages for short, diverse user sessions. Further, a frequency- or recency-based approach is (by definition) unable to recommend "unfamiliar" commands.

Given the potential for the QF-based approach to produce recommendations that are both needed and novel, in the

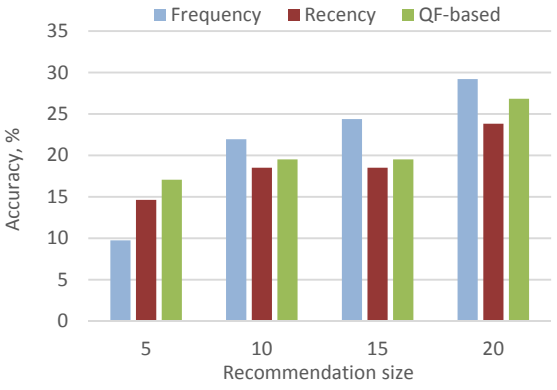

Figure 7. Accuracy of the Frequency-based, the Recency-based and QF-based approach for an average diverse session.

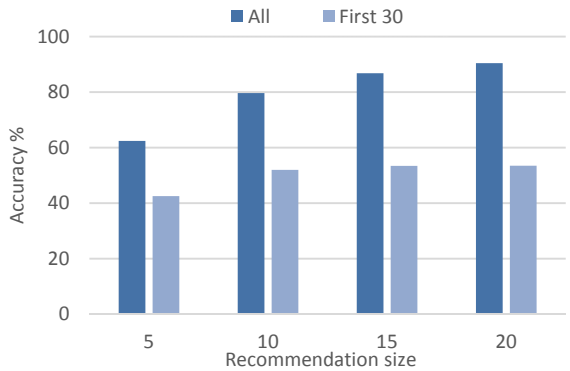

Figure 8. The accuracy of frequency-based approach over users' first 30 command invocations and over all command invocations.

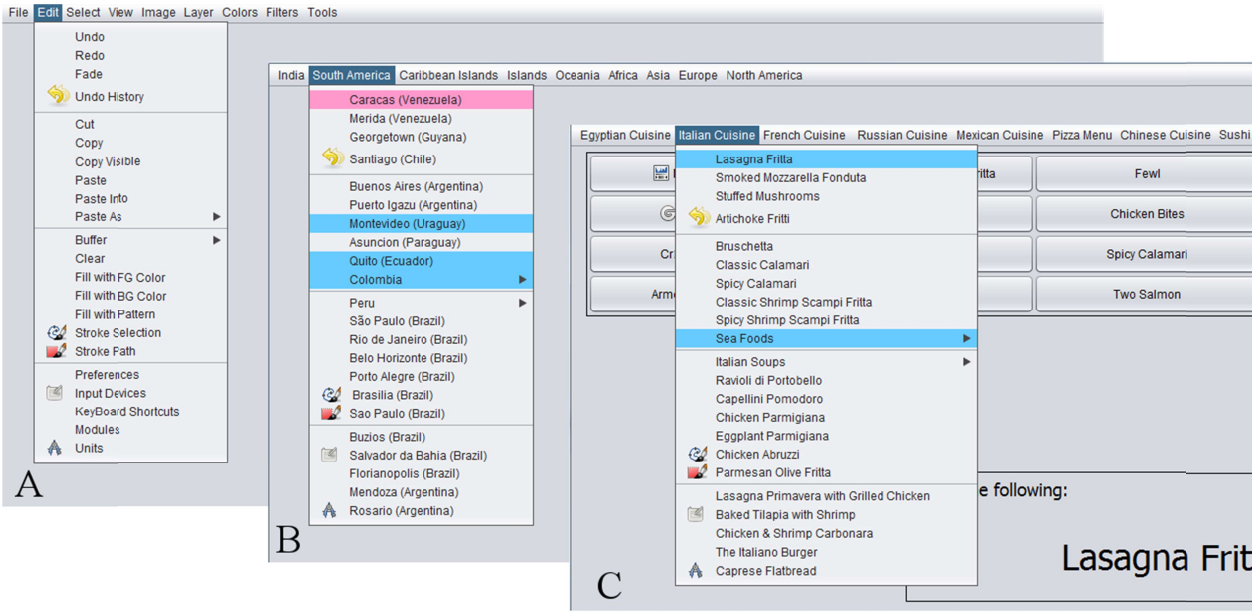

Figure 9. Sample snapshot of the interface variants used in study: Basic interface (A), Combined interface (B), and Separated interface (C). The recommendations highlighted in pink are the "familiar" features and the features highlighted in blue are "unfamiliar" features.

next section, we explore how the system might present these recommendations to the user.

LABORATORY EVALUATION

The goal of our laboratory evaluation was to explore two different ways to present the system's recommendations. As described in the section "Recommendation Types", QFRecs's recommendations can be divided into two types: commands that are "familiar" to the user (i.e., they are part of the user's usage history) and those that are "unfamiliar" (i.e., they are not part of the usage history). We experimented with two different ways to present these recommendations and compared both presentation techniques to a control condition with no recommendations. We explore the impact of the presentation techniques on both primary task performance and on incidental awareness of recommended features not part of the primary task.

Participants

Eighteen participants (2 females) were recruited from a university campus. Participants were between the ages of 18-25 and were rewarded with a 15$ gift card.

Conditions

Our three interface variants were as follows:

1. **Basic Interface:** A control condition with traditional static menus (Figure 9 A).
2. **Combined Interface:** Recommendations for both "familiar" and "unfamiliar" features were highlighted in place (i.e., within the menus) using one of the best known visual highlighting techniques: ephemeral adaptation [11]. With ephemeral adaptation, recommended items appear immediately when a menu is opened, with the remaining items gradually appearing after an initial delay (500 ms as in [11]). The two types of recommendations ("familiar" vs. "unfamiliar") were distinguished only by colour: the "familiar" features were highlighted with a pink overlay and the "unfamiliar" features were highlighted with a blue overlay (Figure 9 B).
3. **Separated Interface:** "Familiar" recommendations were ephemerally highlighted within the menus (using also a blue overlay) but "unfamiliar" recommendations were presented in a separate palette (as well as appearing as non-recommended features in the menus) (Figure 9 C). In the separate palette, the full command path each feature was also available on mouse over. In comparison to the Combined Interface, with the Separated Interface, users could choose to ignore these "unfamiliar" recommendations completely in favor of focusing on their primary task. This palette-based approach has been commonly explored in prior work on novel command recommendations (e.g. [30, 31]).

To enable our system to make recommendations using real GIMP usage data (see the "Tasks" section), and our QF-graph to be built from actual web queries and documentation, the menu hierarchy in all interfaces was modeled after The GNU Image Manipulation Program (GIMP) version 2.8.6. To simplify the interface slightly for our participants, we excluded the "Windows" and "Help" menus, resulting in 9 top-level menus containing a total of 368 features.

Design, Tasks, and Procedure

The experimental task was a sequence of menu selections using each of the three interfaces described above. In other words, our study used a within-subjects design, where all participants experienced all three interface variants. The menu selections were based on a real user's data from the Ingimp dataset described in the "Offline Evaluation" section. We selected data from a user with a sufficiently long sequence that was also close to the data set's mean diversity. The selected usage sequence was 74 selections long and had a diversity of .2065 (defined in the "Offline Evaluation" section). From this sequence, we used the first 20 selections as "training", and the next 50 features as the main task (discarding last 4 selections in the interest of participant time).

For each feature selection, our experimental interface provided participants with the name of the feature, but not the menu name. As a result, participants had to explore the interface (using the top-level menu names as a guide) to find their needed commands. Once the participant correctly selected the displayed feature, the next feature to be selected was displayed. We used the same selection sequence in all interface variants, but used different interface "masks" (the GIMP menus, Geography-related menus, and Cuisine-related menus) to mitigate learning effects between conditions. The structure of all three masks was identical. To further account for potential order effects, the order of interface and the assignment of masks to interface were counterbalanced using a Latin Square.

After each main task, we measured incidental command awareness [11], by having participants perform a recall test. During this recall test participants selected 24 distinct commands that were recommended by the QF-based system but that was not part of the main task.

The procedure for the 1.5 hour experiment was as follows: Participants first completed a background and demographics questionnaire. Then, for each interface variant, participants completed a training task consisting of 20 selections, followed by the main task consisting of 50 selections. After the main task, participants completed the NASA-TLX [22], which measures perceived workload. Participants then completed the recall test described above prior to repeating the above steps with the next interface variant. The session concluded with a comparative questionnaire.

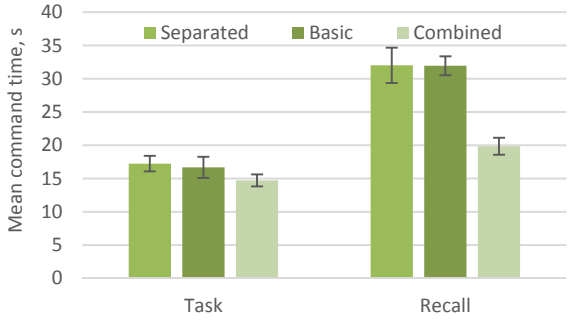

Figure 10. Mean selection time between commands (with standard errors) for the main task (left) and the recall task (right)

In the Combined and Separated interfaces, our prototype system presented its top 20 recommendations based on the user's last 5 command selections. With this particular usage stream, this recommended set accurately predicted the next command in the stream 18% of the time.

Apparatus

An Intel Core i7 desktop with 8 GB of RAM and Microsoft Windows 7 was used for the experiment. The system was connected to a 22'' LCD monitor with a 1920x1080 resolution. The experiment software recorded all timing and selection data.

Hypotheses

Given the low predictive accuracy of QFRecs's recommended set in comparison to those studied in prior work (e.g., [11]), we did not have any apriori hypotheses on the effect of interface on selection speed during the primary task. We did, however, have the following hypotheses with respect to recall selection speed, perceived workload and user preference:

H1 (Recall Speed): The Combined interface will have faster recall times than both the Basic interface and the Separated interface. We expect no difference between the Basic interface and the Separated interface.

H2 (Perceived Workload): Perceived workload will be lower with the Combined interface than with the Separated interface.

H3 (User Preference): Users will prefer the Combined interface over the Separated interface.

Results

We performed our analysis with a one-way RM-ANOVA with *Interface* (Basic, Combined, Separated) as the within-subjects factor. We used $p > 0.05$ as our threshold for significance, and Bonferroni corrections were applied to all post-hoc comparisons. Error bars in the figures represent Standard Error.

Primary Task Selection Time

As expected, we did not find a significant main effect of Interface on primary task selection time ($F_{2,34} = 1.064$, $p = .356$, $\eta^2 = .059$, Figure 10 left). The fact that the recommendations did not significantly improve immediate task performance is consistent with prior results on low accuracy predictors (e.g., [11]). Despite having only limited immediate accuracy, the recommendations did not hurt task performance, and perhaps even helped it slightly (as indicated by the means and effect size), when presented in-place.

Recall Speed (H1)

In the case of the incidental awareness task (i.e., the Recall test), we found the main effect of Interface was statistically significant ($F_{2,34}$ = 12.731, p < .001, η^2 = .428, Figure 10 right). Moreover, the post-hoc comparisons revealed significantly faster selections using the Combined interface (19.8s, se 1.3s) when compared to either the Basic (32.0s, se. 1.4s, p < 0.001) or Separated (32.0s, se. 2.7s, p = 0.005) conditions. The difference between the Basic and Separated conditions was not statistically significant (p = 1.00). Therefore, we find support for *H1*.

Perceived workload (H2)

As a measure of a perceived workload, we used the data from NASA-TLX questionnaires. Table 3 shows significant main effects of Interface on two of the NASA-TLK categories: "hard work" and "frustration". For these categories, participants reported experiencing lower workload with the Combined interface, however, the only significant pairwise difference revealed by the post-hoc comparisons was that of frustration for the Combined and Basic interface variants (p = 0.038). Therefore, we could not fully support *H2*.

User Preference (H3)

Regarding subjective preferences, in the post study questionnaire we asked our participants to rank the different interface types based on their overall performance. The analysis of results showed high inclination towards the Combined option, with 13 participants ranking it as their most preferred interface variation. For comparison, 3 users rated the Separated interface as their primary choice and only two preferred the Basic one (this difference was significant with $\chi^2 = 16.0$, p = 0.002). Therefore we find support for *H3*.

DISCUSSION

Combined, the results of our offline and laboratory evaluations suggest that recommendations generated based on information mined from web documentation is a promising approach to improving command awareness in feature-rich software. They also highlight a number of important considerations moving forward.

Assessing the Utility of Novel Recommendations

Our offline evaluation indicates that for our target application (GIMP), the QF-based approach was able to generate at least some contextually relevant recommendations. For example, when providing the user with 20 recommendations (which would be distributed throughout the entire menu hierarchy), previously collected GIMP usage data indicated that at least one recommended command would subsequently be used within a user's next 15 selections. Further work is required, however, to assess the relevance and utility of recommended commands that do not later appear in the user's command stream, since their omission does not imply a lack of utility. A potential first step in this direction would be to collect relevance ratings from application experts; however, a longer-term experiment is necessary to fully assess the value of these recommendations from the user's perspective. A longer-term experiment would also enable us to compare the QF-based approach to the collaborative filtering approaches explored in prior work [30, 35].

	Basic	Separated	Combined	F	Sig
Mental demand	13.1 (.78)	12.8 (.96)	11.3 (.94)	1.756	0.188
Physical demand	4.5 (1.0)	4.9 (.94)	4.0 (.81)	1.125	0.336
Temporal demand	10.8 (1.3)	10.8 (1.1)	9.7 (1.1)	0.98	0.386
Success	6.2 (1.1)	4.6 (.95)	4.8 (1.0)	1.717	0.195
Hard work	**14.3 (1.0)**	**12.9 (1.0)**	**10.4 (1.2)**	**3.754**	**0.034**
Frustration	**10.9 (.86)**	**9.4 (1.1)**	**7.1 (1.0)**	**5.251**	**0.01**

Table 3. Mean (st. err.) NASA-TLX values (1=low, 20=high). Rows in bold indicate significant differences.

Prototype Extensions and Improvements

Motivated by our initial feasibility study, there are a number of system-related improvements worth exploring. Aside from culling queries from the original QF-Graph that were clearly not representative of high-level tasks, we did little to optimize the graph's suitability to act as a recommender. More sophisticated lexical analysis or machine learning could enable the system to focus its recommendations on a more informative set of high-level tasks. It also possible that restricting the documentation set to specific tutorial repositories would improve the precision of the command-to-task mappings. Finally, there are numerous avenues that could be explored to improve the approach's predictive capabilities, such as incorporating command usage frequency.

In addition improving the algorithmic component, there is also the potential to make the interaction between the system and the user more of a mixed-initiative one [21]. In particular, the system could leverage the human-readable format of the high-level tasks to display its task assessments to the user. The user could then refine these assessments to obtain more tailored recommendations.

Presentation Techniques

Our laboratory evaluation is one of few systematic explorations of how to present command recommendations designed to promote command awareness (as opposed to short-term efficiency). Our results indicate that presenting these types of recommendations in-place can significantly improve incidental command awareness over a palette-based approach. This is perhaps not surprising given that this presentation technique is more obtrusive. What is perhaps more surprising is that the extra visual complexity introduced into the main interface did not appear to negatively impact short-term task efficiency. Users also preferred this in-place presentation strategy and reported

lower levels of frustration. Further exploration is needed to determine the sensitivity of these results to factors such as the number of recommendations, their distribution across the menus, etc.

While our results show initial promise for an in-place presentation technique, there are a number of open questions concerning how to present recommendations in a way that will eventually lead to their adoption. For example, with the palette approach, it would be easier to provide rich supplemental information on why the command is recommended and how it might be used in practice. In a palette, the system could display its confidence in each recommended command, the list of tasks to which the command relates, and links to documentation that illustrate how to use the command. Such information could also potentially be integrated within the main interface (available, for example, on mouse over), but at the risk of impacting immediate task performance. A palette-based approach could also potentially do more to promote command location awareness than simply displaying the command path on mouse over. For example, users could be provided with an animated location cue when they select a command in the palette.

Understanding the above types of presentation-level tradeoffs will be important to the ultimate success of all approaches to command recommendation, not just ones based on Web documentation. It is also possible that a more static presentation technique is desirable. For example, the system could recommend entire task-centric interface that corresponds to the user's most probable high-level tasks [28].

Generalizability

Finally, it would be interesting to explore the generalizability of the QF-based approach to command recommendations to feature-rich applications other than GIMP. Fourney *et al.*'s original QF-Graph graph results suggest that the technique will extend to other applications with a large Web presence [13]. Exploring generalizability to other applications, however, could provide insight on how properties of the graphs themselves affect their abilities to generate useful recommendations, such as the range of high-level tasks present, and the connectedness of the graphs.

In addition to exploring the generalizability of our approach to other feature-rich applications, it will also be important moving forward to explore the generalizability of our study results to a more diverse user population.

SUMMARY

In this paper we have created a prototype recommender system that leverages a new form of information on command relevance: command-to-task mappings mined from Web documentation. Our offline evaluation suggests that this technique has the potential to expose users to a number of new and relevant commands, while our online evaluation suggests value in integrating the recommendations within the main interface. Our work also suggests a number of promising avenues for future research, included developing a more detailed understanding of the design space for recommendation presentation and exploring ways to tailor the system's recommendations according to user feedback. Long-term evaluations of deployed technologies are also needed to assess the impact of these document-based recommendations on command use.

ACKNOWLEDGEMENTS

This work was supported by the GRAND Network Centre of Excellence and the Natural Sciences and Engineering Research Council (NSERC). We also thank Adam Fourney for his invaluable help in providing data for building the QF graph.

REFERENCES

1. Adar, E., Dontcheva, M., and Laput, G. CommandSpace : Modeling the Relationships Between Tasks , Descriptions and Features.In *Proc. UIST 2014*, 167-176.
2. Bunt, A., Conati, C. and McGrenere, J. Supporting Interface Customization Using a Mixed-Initiative Approach. In *Proc. IUI 2007*, 92-10.
3. Bunt, A., Dubois, P., Lafreniere, B., Terry, M. and Cormack, D. TaggedComments: Promoting and Integrating User Comments in Online Application Tutorials. In *Proc. CHI 2014*, 4037-4046.
4. Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. Exploiting redundancy in question answering. In Proc. *SIGIR* Research and Development in Information Retrieval 2001, 358-365.
5. Chi, P.P., Ahn, S., Ren, A., Dontcheva, M., Li, W., and Hartmann, B. MixT: Automatic Generation of Step-by-Step Mixed Media Tutorials.In *Proc. UIST 2012*, 93–102
6. Dong, T., Dontcheva, M., Joseph, D., Karahalios, K., Newman, M.W., and Ackerman, M.S. Discovery-Based Games for Learning Software. In *Proc. CHI 2012,* 2083–2086.
7. Ekstrand, M., Li, W., Grossman, T., Matejka, J., and Fitzmaurice, G. Searching for Software Learning Resources Using Application Context. In *Proc. UIST 2011*, 195–204.
8. Fernquist, J., Grossman, T., and Fitzmaurice, G. Sketch-Sketch Revolution: An Engaging Tutorial System for Guided Sketching and Application Learning.In *Proc. UIST 2011*, 373–382.
9. Findlater, L. and McGrenere, J. A Comparison of Static, Adaptive, and Adaptable Menus. In Proc. *CHI 2004*, 89-96.

10. Findlater, L., Mcgrenere, J., Evaluating Reduced-Functionality Interfaces According to Feature Findability and Awareness. In *Proc. Interact 2007*, 592-605.
11. Findlater, L., Moffatt, K., Mcgrenere, J., and Dawson, J. Ephemeral Adaptation: The Use of Gradual Onset to Improve Menu Selection Performance. In *Proc. CHI 2009*, 1655–1664.
12. Fourney, A., Mann, R., and Terry, M. Characterizing the Usability of Interactive Applications through Query Log Analysis.In *Proc. CHI 2011*, 1817–1826.
13. Fourney, A., Mann, R., and Terry, M. Query-feature graphs: bridging user vocabulary and system functionality. In *Proc. UIST 2011*, 207-216.
14. Greenberg, S. The Computer User As Toolsmith: The Use, Reuse, and Organization of Computer-based Tools. Cambridge University Press, 1993.
15. Gajos, K., Czerwinski, M., Tan, D. and Weld, D. Exploring the design space for adaptive graphical user interfaces. In *Proc. AVI 2006*, 201-208.
16. Grabler, F., Agrawala, M., and Erato, J.S.T. Generating Photo Manipulation Tutorials by Demonstration. In *Proc. SIGGRAPH* 2009, 1–9.
17. Grossman, T.; Fitzmaurice, G. and Attar, R. A survey of software learnability: metrics, methodologies and guidelines. In *Proc. CHI* 2009, 649-658.
18. Grossman, T., Matejka, J., and Fitzmaurice, G. Chronicle: Capture, Exploration, and Playback of Document Workflow Histories.In *Proc. CHI* 2010, 143–152.
19. Grossman, T. and Fitzmaurice, G. ToolClips: An investigation of contextual video assistance for functionality understanding. In *Proc. CHI* 2010, 1515-1524.
20. Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proc. UAI* 1998, 256–265.
21. Horvitz, E. Principles of mixed-initiative user interfaces. In *Proc CHI 99*. 159-166.
22. Hart, S.G., Staveland, L.E.: Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. Human Mental Workload 1, 139–183 (1988)
23. Kim, J., Nguyen, P.T., Weir, S., Guo, P.J., Miller, R.C., and Gajos, K.Z. Crowdsourcing Step-by-Step Information Extraction to Enhance Existing How-to Videos.In *Proc. CHI 2014*, 4017–4026.
24. Kong, N., Grossman, T., Hartmann, B., Fitzmaurice, G., and Agrawala, M. Delta: A Tool For Representing and Comparing Workflows.In *Proc. CHI 2012*, 1027–1036.
25. Lafreniere, B., Bunt, A., Lount, M. and Terry, M. Understanding the Roles and Uses of Web Tutorials. In *Proc ICWSM 2013*, 303-310.
26. Lafreniere, B., Bunt, A., Whissell, J., Clarke, C., and Terry, M. Characterizing Large-Scale Use of a Direct Manipulation Application in the Wild. In *Proc GI 2010*, 11-17.
27. Lafreniere, B., Grossman, T., and Fitzmaurice, G. Community Enhanced Tutorials: Improving Tutorials with Multiple Demonstrations. *In Proc. CHI 2013*, 1779–1788.
28. Lafreniere, B., Krynicki, F., Terry, M., Bunt, A. and Lount, M. AdaptableGIMP: Designing a Socially-Adaptable Interface, In *Proc. UIST 2011*, 89-90.
29. Li, W., Grossman, T., and Fitzmaurice, G. GamiCAD: A Gamified Tutorial System For First Time AutoCAD Users.In *Proc. UIST 2012*, 103–112.
30. Li, W., Matejka, J., Grossman, T., Konstan, J. A. and Fitzmaurice, G. Design and Evaluation of a Command Recommendation System for Software Applications. In ACM Trans. Comput.-Hum. Interact., ACM, 2011, 18, 6:1-6:35.
31. Linton, F. and Schaefer, H.-P. Recommender Systems for Learning: Building User and Expert Models Through Long-Term Observation of Application Use. User Modeling and User-Adapted Interaction, Kluwer Academic Publishers (2000), 181-208.
32. Lount, M., and Bunt. A., Characterizing web-based tutorials: Exploring quality, community, and showcasing strategies. In *Proc. SIGDOC 2014*.
33. Matejka, J., Grossman, T., and Fitzmaurice, G. Patina: Smart Heatmaps for Visualizing Application Usage.In *Proc. CHI 2013*, 3227-3236.
34. Matejka, J., Grossman, T., and Fitzmaurice, G. IP-QAT: In-Product Questions, Answers & Tips. In *Proc. UIST 2011*, 175–184.
35. Matejka, J., Li, W., Grossman, T., and Fitzmaurice, G. CommunityCommands: Command Recommendations for Software Applications. In *Proc. UIST 2009*, 193–202.
36. McGrenere, J., Baecker, R.M., and Booth, K.S. An Evaluation of a Multiple Interface Design Solution for Bloated Software. In *Proc. CHI 2002*, ACM Press (2002), 163–170
37. Terry, M., Kay, M., Van Vugt, B., Slack, B., and Park, T. 2008. Ingimp: introducing instrumentation to an end-user open source application. In Proc. *CHI 2008.* 607-616.