Characterizing the Usability of Interactive Applications Through Query Log Analysis

Adam Fourney
afourney@cs.uwaterloo.ca

Richard Mann
mannr@uwaterloo.ca

Michael Terry
mterry@cs.uwaterloo.ca

David R. Cheriton School of Computer Science
University of Waterloo
Technical Report CS-2010-18

ABSTRACT

People routinely rely on Internet search engines to support their use of interactive applications, making query logs a rich source of data cataloguing the day-to-day tasks and needs of a user base. In this paper, we introduce an automated process for harvesting, ordering, labeling, and grouping search queries related to any publicly available interactive system. The end result is a data set that can complement and augment data collected through traditional usability methods. We call this process *CUTS*—characterizing usability through search. The labeled, ordered data produced by CUTS can be assembled in minutes, is timely, has a high degree of ecological validity, and is arguably less prone to self-selection bias than traditional usability methods. We describe this process and demonstrate applications of its use with a handful of interactive systems.

Author Keywords

Query log analysis, Usability

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: Miscellaneous

INTRODUCTION

People rely on search engines (e.g., Google[1], Yahoo![2], Bing[3], etc.) to support their use of interactive systems [4, 6]. For example, users submit search queries to locate tutorials, troubleshoot problems, or learn how to use specific features of an application. Given this behavior, search engine query logs serve as centralized repositories cataloguing the day-to-day needs of the user base of an interactive system.

In this paper, we argue that search engine query logs can be filtered and transformed into forms that usefully complement and augment data collected via traditional usability methods, such as observational studies, instrumentation, and surveys. We demonstrate this potential by introducing an automated process for harvesting, ordering, labeling, and grouping search queries to understand the common tasks and needs of a user base (Figure 1). We call this process *CUTS*—characterizing usability through search. Importantly, the labeled, ordered data produced by CUTS can be assembled in minutes, is timely, has a high degree of ecological validity, and is arguably much less prone to self-selection bias than traditional means of collecting data from users.

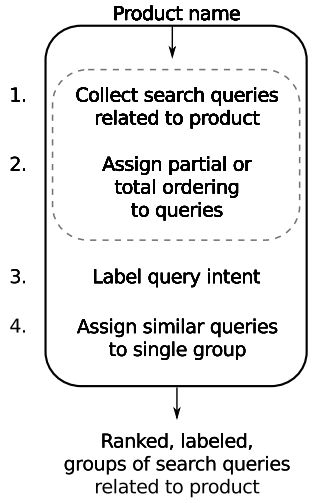

Figure 1. An overview of CUTS. Steps 1-2 are easily performed with access to raw query logs, but otherwise require approximation techniques. Step 3 utilizes our query taxonomy specialized for interactive systems.

As an example of the utility of this approach, an approximation of this process can be illustrated using Google Suggest, the service that provides query completion suggestions for a given input. Given the phrase "firefox how to", Google Suggest produces a list of 10 query completion suggestions (Figure 2). As we will show later, these suggestions closely correspond to the 10 most popular queries matching that input. Knowing this, by using the generic pattern "*product* how to" we can quickly obtain a ranked, ordered list of some the most common issues for any interactive system.

From the list of top 10 Firefox "how to" suggestions (Figure 2), it is immediately clear that users have a number of privacy and security concerns, as evidenced by their desire to clear their cache, history, and cookies. However, the eighth item ("get menu bar back") is particularly interesting. An inspection of the Firefox user interface (version 3.6 on Win-

[1] http://www.google.com
[2] http://www.yahoo.com
[3] http://www.bing.com

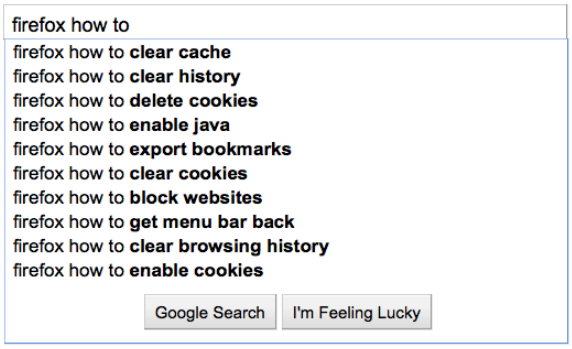

Figure 2. The top 10 suggestions provided by Google Suggest for the phrase "firefox how to".

dows), reveals that the top-level menu bar is easily hidden by deactivating the "Menu bar" item in Firefox's "View → Toolbars" sub-menu. However, once this action is taken, it is not easily reversed: the top-level menuing system is now hidden, removing the very means the user would employ to attempt to re-instate the menu bar. What is noteworthy about this example is that we quickly moved from data derived from query logs to a testable hypothesis regarding the usability of the software.

The contributions in this paper lie in expanding this manual process to the automated one shown in Figure 1. While seemingly straightforward, automating this process requires overcoming a number of challenges: Raw query logs are not made publicly available; there is a need to automatically determine query intent for the purposes of labeling and filtering queries (for example, to distinguish troubleshooting queries from those seeking to download the application); and differently phrased queries on the same topic must be reduced to a single query. Our specific contributions, outlined below, address these challenges.

To address the problems of obtaining and ranking search queries, we demonstrate how publicly available query suggestion services (e.g., Google Suggest) and web-based tools for advertisers can be employed to create reasonable approximations of raw query logs. This method also includes fallback strategies for ordering query data when precise frequency counts are not available for the queries.

We also introduce two new query classification schemes to address the need to label queries. The first classification scheme is a taxonomy that extends previous search query taxonomies to include categories relevant to interactive systems. For example, this new taxonomy differentiates between queries issued to troubleshoot a problem and those seeking a tutorial or instructions. The second classification scheme considers *how* a query is phrased. We show that how a query is phrased closely corresponds to the categories of our specialized taxonomy. Since determining how a query is phrased is a relatively simple task, our method exploits the relationship between these two classification schemes to ascribe query intent from query phrasing.

Finally, we introduce a set of heuristics that enable differently phrased queries on the same topic to be represented by a single group. These heuristics transform a query into a canonical form using existing methods (such as word lemmatization, and stop word removal). We show that the highest ranking queries are also members of the largest query groups (by number of alternate phrasings).

The rest of this paper is structured as follows. We first present related work, then describe our method for harvesting and ranking search queries using publicly available services. We then introduce our two classification schemes and show how they can be used to label search queries. The final step of the process, grouping queries, is discussed and a set of strategies are introduced to assist with this process. Next, we present a series of examples illustrating the overall utility of this approach, followed by a discussion of the limitations of the technique. We conclude with directions for future work.

BACKGROUND & RELATED WORK

In recent years, researchers have demonstrated the potential for search engine query logs to model and predict real-world phenomena and events. For example, Jeremy Ginsberg *et al.* have demonstrated how query logs can be employed to help track the spread of influenza over time [10]. In this latter research, "health-seeking behaviour" is automatically detected by monitoring search terms associated with influenza (symptoms, medications, etc.). This allows the Google Flu Trends application[4] to estimate the prevalence of influenza infections on a week-to-week basis. The resultant models closely agree with data released by the Center for Disease Control (CDC), though they exhibit much less lag: Models built using query logs show a 24 hour lag in tracking flu trends, compared to the week lag of the CDC.

More generally, Richardson [20] argues that query log analysis could quickly become an indispensable tool for researchers working in such human-centric fields as anthropology, sociology, psychology, medicine, economics, and political science. He notes that query logs function as if "a survey were sent to millions of people, asking them to, every day, write down what they were interested in, thinking about, planning, and doing." Accordingly, he argues that "taken as a whole, across millions of users, ... queries constitute a measurement of the world and humanity through time" [20]. To demonstrate his point, Richardson describes a common search pattern that unfolds over the course of three to six months, starting with a user's search for "mortgage calculators." Within a week, these same users search for "realtors." About one month later, they search for legal services (e.g., "notary"), and three months later, their searches include those for home furnishing (e.g., "pottery barn"). As with Google Flu Trends, this latter example shows the potential for query logs to discover and model real-world phenomena.

Within the realm of interactive systems, the research literature contains many accounts of search query logs being used to improve *information interfaces*—interfaces in which find-

[4] `http://www.google.org/flutrends/`

ing or accessing information is a user's primary task. For example, Zhicheng Dou *et al.* demonstrate how query logs can be used to improve personalized search [8]. It is also common for website designers to use query logs to help determine what links should be provided in their site's top-level navigation [19]. Our work broadens and generalizes these previous uses of query logs, demonstrating their utility in understanding users' needs with any interactive system.

While prior work in interactive systems has focused on using query logs to improve information interfaces, work by Brandt *et al.* have demonstrated how software developers employ search engines when creating software [6]. Through studies of developer practices, Brand *et al.* show that software developers make extensive use of Google for a variety of programming-related tasks, such as using search to translate knowledge from one domain to another (e.g., from one programming language to another). In subsequent work, Brandt *et al.* streamlined this common strategy by building web search tools directly into integrated development environments (IDEs) [5]. While this prior work validated their initial study findings through an analysis of query logs, our work more generally considers query logs and how they can be employed to learn the specific deficiencies and limitations of any interactive system.

Importantly, all of the aforementioned research has been conducted by institutions or companies with direct access to search query logs. Access to these query logs is highly guarded, especially after the privacy problems encountered after AOL released (what they felt was) an anonymized sample of their query logs [24]. Lacking direct access to raw query logs, Bar-Yossef and Gurevich have demonstrated how statistics of these logs can be approximated using an importance sampling technique [3]. This technique estimates the popularity of certain keywords by using parameterized models (derived from the aforementioned AOL search query logs) and by sampling query completions provided by query completion suggestion services. Our work is inspired by this research, as we also use query completion suggestion services. However, in contrast to this previous work, we perform exhaustive searches of the query suggestions for a specific topic (namely, a specific interactive system), as a means of sampling the raw query logs.

In summary, previous work in other domains demonstrates the overall utility of query log analysis: It yields timely, highly ecologically valid data that can quickly lead to significant insights when studying a wide range of phenomena. In the rest of this paper, we demonstrate its specific utility in the realm of interactive systems by detailing each step of the CUTS process.

QUERY HARVESTING

When access to raw query logs is not possible, search queries can be harvested using publicly accessible interfaces: Modern search engines provide indirect and privacy-preserving access to their logs through their query completion suggestion services [3]. In this section, we describe a process for systematically harvesting all the queries related to a particular interactive system using these services. We also provide evidence that the results of this method can be considered a representative sampling of the raw query logs.

Harvesting from auto-completion services

Query completion suggestion services operate as if backed by a prefix tree [3]. When viewed in this way, the characters making up a partially entered query define a path through the tree starting at the root, passing through numerous nodes. Each node contains a listing of popular queries whose prefix matches the path taken thus far. Query completion services follow the paths prescribed by partially entered queries, and return the suggestions listed at the end of these paths.

Given the tree-like structure of these services, a standard depth-first or breadth-first tree traversal can be performed by expanding partial queries one character at a time, starting with the name of the system under investigation (figure 3). A leaf (or external node) is reached when the completion service returns no suggestions for the given prefix.

```
firefox,
firefox a, firefox aa, firefox aaa, …
…
firefox z, firefox za, firefox zaa, …
```

Figure 3. Input sequence representing a depth-first traversal of Google's prefix tree rooted at "firefox".

Mining additional queries

Some search providers, such as Google, vary their query suggestions depending on the position of the caret in the search query input box (Figure 4). More specifically, Google provides a list of the top 10 completions that either begin or end with the phrases on the left or right side of the cursor. Given this behaviour, the whole tree traversal procedure can be repeated to uncover query suggestions that end with a particular suffix, providing a more complete sampling of the query logs.

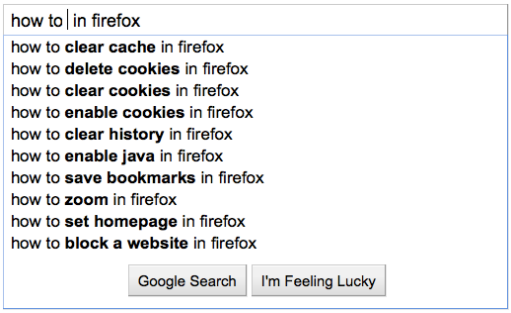

Figure 4. Google's "Suggest" auto-completion service varies its suggestions based on the caret position.

By executing a systematic search of the query completion tree, a significant number of queries can be collected for a given topic. For example, a systematic search of query suggestions incorporating the term "Firefox" uncovers 74,795 unique queries. Similar results were obtained for other systems for which we collected data (Table 1).

Application	Description	# of Query Suggestions
ubuntu	A Linux distribution	122,242
photoshop	An image editor	119,791
firefox	A web browser	74,795
kindle	An eBook reader	21,621
gimp	An image editor	14,569
nook	An eBook reader	8,985
audacity	An audio editor	6,517
kobo	An eBook reader	2,680
inkscape	A vector graphics editor	2,501

Table 1. Number of unique query suggestions provided by Google for a number of interactive systems.

Representativeness and Timeliness of Auto-Completions

In harvesting these queries, our working assumptions are that (1) query completion services are derived from the raw query logs, (2) a given query's prevalence in these logs will have some bearing on its ranking in the list of suggestions, and (3), the suggested completions are *timely*. That is, we assume that query completion services assign more weight to queries performed within a recent window of time. In the following subsections, we briefly provide evidence that these assumptions are sufficiently valid for our purposes.

Representativeness of Query Completion Suggestions

In the case of Google, some information about their query suggestion service has been published [11]. Specifically, Google's documentation notes that "All of the queries shown in (Google) Suggest have been typed previously by other Google users". Google also states:

> *Our algorithms use a wide range of information to predict the queries users are most likely to want to see. For example, Google Suggest uses data about the overall popularity of various searches to help rank the refinements it offers.* [14]

Later, when discussing the use of marketing tools to rank queries, we will provide further evidence that auto-complete suggestions have a close correspondence to the most frequently, recently performed queries.

Timeliness of Query Suggestions

To identify trends and new issues as they arise, it is desirable that query suggestion services emphasize recent searches over those performed in the more distant past. To study the timeliness of Google's query suggestion service, we monitored the query completion suggestions for a range of products and software applications for a period of approximately 3 months (June 2010 through August 2010, inclusive). Suggestions were sampled on Monday, Wednesday and Friday on each week during this timeframe. An analysis of the collected data reveals that Google updates its auto-completion database approximately once every 14 days. These results indicate that Google is actively maintaining its query suggestion database.

Knowing the frequency with which these services are updated is advantageous, but it is not sufficient for determining the extent to which current search trends are represented in query suggestions. To investigate this question, we can examine when a noteworthy event begins to appear in query suggestions. A prime candidate for exploring this question is provided by the release of the iPhone 4 on June 24th, 2010. Almost immediately, there were reports of significant signal degradation when the phone was held in a certain way [9]. The first evidence of this issue was spotted in the query suggestions on July 14th. On this date, the partial query "iphone d" resulted in Google suggesting "iphone death grip", while "iphone a" yielded "iphone antenna", and "iphone how to h" yielded "iphone how to hold". None of these queries appear in the suggestions sampled on previous dates. This corresponds to a lag of about 20 days, suggesting that the query completion services place sufficient weight on recent queries.

RANKING QUERIES

After harvesting queries, the next step is to assign an importance rank to each query. When queries are sampled from query suggestion services, detailed ranking information is not made available (current services do not report the frequency of each query returned). We substitute this missing data in two ways. First, we complement our data set with data from advertising and market research tools, such as Google AdWords [12]. Second, we examine the structure of the synthesized prefix tree to obtain a *partial ordering* of the queries not covered by the market research tools. We describe each technique in turn.

Using Marketing Tools to Assign Ranks

Google provides a set of tools that can be directly applied to the problem of ranking queries. Specifically, Google Insight [13], and the Google AdWords Keyword Tool [12] can both be applied to this problem. While both tools are intended to help marketers valuate keywords for advertising purposes, we can instead use them to "valuate" queries related to interactive systems.

Of the two Google tools, Google AdWords provides the most information, and can be configured to report the estimated average monthly global search volume for any exact phrase provided by a user. As such, it is possible to directly rank many query suggestions using this tool alone. In doing so, we again find that there is good correspondence between the harvested queries and their estimated search volume.

While many queries can be ranked directly using the Google AdWords tool, not all queries can be ranked in this way; Google AdWords provides no data for queries whose monthly search volume is below some threshold, and this threshold is reached well before the list of query suggestions has been exhausted. For example, on May 22nd, 2010, we harvested 2501 unique query suggestions for the Inkscape vector graphics program. However, Google AdWords provides search volume data for only 597 of those queries. In short, about three quarters of the data cannot be ranked using this method, so we must employ another means of ranking queries, as described next.

Generating a Partial Ordering

While query suggestion services do not return the frequency with which each suggested query is performed, we have shown that they operate by returning the most popular queries for a given input. We can use this behaviour to derive a *partial ordering* of the query suggestions. The key insight is this: For a given prefix, we know that *the 10 query suggestions returned for that exact prefix are deemed more relevant than all other queries later harvested that begin with that same prefix*. An example illustrates this point.

Returning to the earlier Firefox example, the suggestion "firefox menu bar missing" appears in the top 10 suggestions for the prefix "firefox m". Thus, we can infer that the "firefox menu bar missing" query is more relevant than the 2362 other suggestions occurring in the data set that also share the prefix of "firefox m". We say that this query has 2362 *subordinates* in order to convey this relationship.

This ranking technique provides only a partial ordering because we can only perform comparisons of a node with its ancestors and descendants in the prefix tree. We *cannot* directly compare suggestions occupying separate branches of the tree.

The partial ordering can be extended to a total ordering that *approximates* the true ranking of queries based on search volume. This is done by simply sorting all remaining queries according to their number of subordinates. When queries are ranked using this metric, the pairwise relationships established by the partial ordering are preserved. As an example, if query B is the subordinate of query A, then query A's subordinate count will necessarily be greater than that of B (subordinates of B are also subordinates of A). As such, B will appears lower in the rankings.

These first two steps of harvesting and ranking queries provide us with a suitable, privacy-preserving, publicly accessible replacement for raw query logs. In the remainder of the paper, our technique assumes only that one has access to a ranked list of search queries relating to the interactive system of interest.

QUERY CLASSIFICATION AND QUERY TAXONOMIES

Given a ranked list of queries, the next step is to label and classify them according to the intent of the individual submitting the query. However, we first need to understand the various types of queries users submit to search engines to support their use of interactive systems. While previous work has developed a number of taxonomies for general classification of search queries (e.g., to distinguish between navigational and information-seeking queries) [7, 16, 21], we found these too broad for our purposes. Instead, a classification scheme specialized for this domain is needed. Additionally, we need understand what features of a query can be used to support automatic labeling of the queries.

In this section, we address both of these needs: We introduce a taxonomy of *query intent* specialized for interactive systems, and a second classification scheme that describes how a query is *phrased*. As we will show, query phrasing is strongly related to query intent.

Query intent taxonomy for interactive systems searches

To develop a taxonomy of queries related to interactive systems, we performed open coding of 200 randomly sampled queries related to the GIMP software application. From this initial coding, we identified a set of common, higher-level themes, which led to our taxonomy. The resultant taxonomy includes six separate classes of interactive system queries, synthesized from the perspective of query intent:

QUERY INTENT:

- **Operation Instruction**
  Would the query be used to find instructions for performing a specific operation?
- **Troubleshooting**
  Would the query be used for troubleshooting a bug or error?
- **Reference**
  Would the query be used to find reference material? (e.g., a list of keyboard shortcuts)
- **Download**
  Would the query be used acquire, download, or install something?
- **General Information**
  Would the query be used to find general information about the application? (e.g., product reviews or comparisons)
- **Off-topic**
  Is the query unrelated to the software / project?

Query phrasing classification scheme

In parallel with developing the former taxonomy, we also developed a classification scheme that describes how individual queries are phrased. The motivation for developing this scheme arose during our open coding sessions: In considering the range of queries, it appeared that *how* a query was phrased was very much related to the *intent* of the user. As we will show, there is indeed a relationship between the two.

Based on the open coding of the queries, the following high-level categories of query phrasing were identified:

QUERY PHRASING:

- **Noun phrase** (e.g., gimp brushes)
- **Imperative statement** (e.g., gimp rotate text)
- **Question** (e.g., how to draw a line in gimp)
- **Statement of fact** (e.g., gimp won't start)
- **Present participle** (e.g., rotating text in gimp)
- **Other**

In contrast to the former query intent taxonomy, the above query phrasing classification scheme is not specific to interactive systems and is thus applicable to query analysis in other domains.

Query Source	κ intent	κ phrasing
Firefox, top 50	0.74 (substantial)	0.80 (substantial)
Firefox, random 50	0.86 (near perfect)	0.80 (substantial)
GIMP, top 47	0.66 (substantial)	0.72 (substantial)
GIMP, random 48	0.66 (substantial)	0.81 (near perfect)

Table 2. Inter-rater reliability for each of the ≈50 queries in each sample set.

Validating the classification schemes

To validate these two classification schemes, two researchers applied both schemes to 195 sampled queries. For this experiment, harvested queries for GIMP and Firefox were used, with queries selected as follows: For each application, the top 50 queries (by search volume) were selected, followed by an additional 50 randomly selected queries. The resulting set of 200 samples shared 5 queries in common with the set of queries used for the initial open coding and were thus excluded from our validation process.

In labeling this data set, we achieved an overall inter-rater reliability rate of $\kappa_{\text{intent}} = 0.76$ for query intent, and $\kappa_{\text{phrasing}} = 0.79$ for query phrasing, using the Cohen's kappa measure of rater agreement. Inter-rater reliability across the 4 sources of queries is listed in table 2. The observed agreements are considered to be substantial [17], suggesting their overall utility as instruments for labeling search queries.

Before describing how this query phrasing classification scheme can be used to identify query intent, we first show how queries are distributed across these two classification schemes. These query distributions lend additional arguments for the overall utility of this approach.

Characterizing query data

The classification of the 195 labeled queries is listed in table 3. The categories we find interesting for usability analysis coincide with the first two listed in the table and taxonomy: "Operating Instruction", and "Troubleshooting". In our sample, about half of all query suggestions fall within categories that are of interest to HCI researchers and practitioners, demonstrating the overall richness of query logs when studying interactive systems.

Relationship between query phrasing and intent

If we compare how a query is classified in each scheme, we find that how a query is phrased strongly correlates with query intent. These findings are summarized in table 4.

There are a few noteworthy observations to make in this table. As can be seen, in our sample set, if a query is phrased as an imperative statement, there is a 91% chance that the query is seeking operating instructions. A similar probability (87%) applies if the query is phrased as a question. Finally, if a query is phrased as a statement of fact, then it is almost certainly being used for troubleshooting. These relationships provide us with a set of strategies for labeling queries, which we describe next.

	Rater 1		Rater 2	
Query Intent	Freq.	%	Freq.	%
Op Instr.	84	43%	80	41%
Troubleshooting	15	8%	17	9%
Reference	21	11%	19	10%
Download	55	28%	62	32%
General	12	6%	12	6%
Off topic	8	4%	5	2%

Table 3. Frequencies of query intent labels as assigned by two raters.

Query Intent	Noun	Imperative	Question	Fact	P. Participle	Other
Opr. Inst.	0.303	0.909	0.867		1.000	
Troubleshooting	0.045			1.000		
Reference	0.136	0.030	0.133			
Download	0.409	0.061				
General:	0.106					
Off topic						1.000

Table 4. Probability of query intent given its phrasing type.

Labeling heuristics

As we have shown in the previous section, queries useful to the study of interactive systems can be selected according to how they are phrased. Through further inspection of the data, we have also found that certain keywords or patterns are highly indicative of each of the different phrasing types. For example, queries containing the phrase "how to" are labeled as "Questions" and thus also labeled with the "Operating Instructions" category. A full list of phrasing patterns are listed in table 5, and serve as basic heuristics for labeling different types of queries. For each pattern in the table, all queries matching that pattern are assigned the labels as prescribed by the left column.

Many queries will not match any pattern, and will thus go unlabled at this stage of processing. In the next section we describe a technique for grouping related queries. When queries are grouped, labels for the individual queries can be extended to the group, increasing the coverage of the heuristic labeling.

Labels		Pattern	Example Query
Operating Instructions	Question	**how to** ___ **in** *SystemName*	how to delete history in firefox
		SystemName **how to** ___	firefox how to clear cache
		can *SystemName* ___	can firefox block websites
		does *SystemName* ___	does firefox have private browsing
	Imperative	**use** *SystemName* ___	use firefox for windows update
		make *SystemName* ___	make firefox default browser
		SystemName **set** ___	firefox set default zoom
		create ___ **in** *SystemName*	create a new profile in firefox
		SystemName **create** ___	firefox create pdf
Troubleshooting	Fact	firefox **is / isn't** ___	firefox is starting slow
		SystemName **can / can't** ___	firefox can't add bookmarks
		SystemName **will / won't** ___	firefox won't open pdf
		SystemName **does / doesn't** ___	firefox doesn't play sound
		SystemName **has / hasn't** ___	firefox has no address bar

Table 5. Filtering templates for labeling the phrasing and likely intent of queries.

"firefox lost toolbar"	
lost my toolbar firefox	firefox toolbars lost
lost firefox toolbar	firefox lost my toolbar
lost all toolbars in firefox	firefox lost all toolbars
lost toolbar in firefox	firefox lost toolbar
lost my toolbar in firefox	lost my firefox toolbar
firefox toolbar lost	

Table 6. 11 distinct queries which share the canonical representation "firefox lost toolbar".

GROUPING SIMILAR QUERIES

The final step in CUTS is to reduce the variability with which queries are expressed in the data set. In query logs, common questions or issues are expressed using a number of different query phrasings. As an example, GIMP users may search "how to draw a circle in gimp", or they may simply type "gimp draw circle". Given this variability, it is desirable that similar queries be grouped, and their weights or rankings combined, in order to better estimate the prevalence of a given issue.

To group similar queries, we transform queries to a canonical form where inconsequential differences are ignored (e.g., See table 6). This transformation applies the following rules:

- Convert inflected word forms to common word lemmas. (e.g., "deleting cookies" becomes "delete cookie")
- Remove all instances of stop words. (e.g., "and", "the", "to", "but", etc.)
- Remove words devoid of alphabetic letters. (e.g.: "3.6.10", and other non-english strings)
- Sort the query terms alphabetically.

Using this technique, it is possible to achieve a modest reduction of the data set to a smaller set of ranked queries. As an example, the Firefox data set of 74,795 unique queries is represented by 39,435 canonical query groups (53% of the original size). More importantly, this procedure preferentially groups popular queries, and yields a second and independent method for identifying the most common queries in the query logs. To illustrate this point, table 7 lists the cardinality of the canonical groups associated with the top 10 "firefox how to" queries already mentioned in the Introduction. All but the last of these queries fall within the top 99.6^{th} percentile of canonical group sizes, thus reinforcing the popularity of these concerns.

OUTPUT OF CUTS

At this stage of the process, queries have been sampled, ranked, labeled, and grouped by canonical form. The output of the process is a categorized and ranked list of query groups relating to the system under investigation; each query group is labeled with the text of its highest ranking member (i.e., the member whose search volume is greatest). A sample of this output, for the "firefox" application, is presented in Table 8.

"Firefox how to" ...	Canonical Form	Cardinality of group
clear cache	cache clear	110
delete cookies	cookie delete	60
clear cookies	clear cookie	44
enable java	enable java	41
export bookmark	bookmark export	40
enable cookies	cookie enable	32
clear history	clear history	30
block websites	block website	29
get menu bar back	back bar get menu	16
clear browsing history	browse clear history	5

Table 7. Canonical groups associated with the top 10 "firefox how to" queries.

Note that ranked output can still contain thousands of entries. It is thus useful to employ standard text visualization techniques to further summarize the data. As an example, the visualization in figure 5 was constructed from the data processed for the GIMP image manipulation software. From this visualization it is clear that GIMP users perform many searches relating to transparency, layers, and drawing primitives (e.g., straight lines, rectangles, circles, etc.). This latter problem (how to draw geometric primitives) is noteworthy because GIMP provides few tools for drawing simple shapes. We describe this latter issue in more detail in the Applications section that follows.

Operating Instruction (Question / Imperative Stmt.)	**Troubleshooting** (Statement of Fact)
clear cache	not responding
clear cookies	can not open pdf
delete cookies	slow
block websites	crashing
enable cookies	in safe mode
proxy	not checking my spelling
delete history	constantly crashing
speed up	lagging
remove persona	zoomed in
save bookmarks	not remembering passwords
...	...

Table 8. Query groups, related to "Firefox", output after query harvesting, ranking, labeling and grouping.

gimp	
image	transparent, cut, out, two, change, size
use	tool, brush, clone, path, scissor, heal
layer	open, get, select, transparent, add, multiple
transparent	background, image, layer, white, color, erase
brush	install, add, use, download, get, load
background	transparent, change, remove, white, delete, add
change	color, background, size, font, image, text
text	change, curve, edit, rotate, bend, remove
color	change, eye, replace, hair, another, splash
draw	line, rectangle, curve, circle, arrow, shape
add	brush, font, border, background, layer, watermark
picture	cut, look, out, put, background, change
install	brush, plugin, font, script, gap
font	install, add, change, download, into, put
photo	black, edit, white, use, color, collage

Figure 5. A visualization of the query suggestions related to *GIMP*. The leftmost column lists, in descending order, the words that occur most frequently in the query suggestions. Similarly, each row lists the words that co-occur most frequently with those listed in the left column.

APPLICATIONS

In this section, we apply our technique to a number of different interactive systems. Our goal here is to demonstrate the wide range of insights that can be gained using this approach. We structure this section by showing how issues related to language, desired functionality, and poor affordances can all be detected using this technique.

"Speak the User's Language"

Query logs provide an excellent view of the vocabulary and terminology with which users conceive their use of interactive systems. However, this terminology does not always match that which is used by their systems. When such discrepancies arise, the associated systems can be considered to be in violation of Jakob Nielsen's "Speak the User's Language" usability heuristic [18]. We provide two examples of this problem that we identified using our technique.

Black and White, but not Grayscale

The GNU Image Manipulation Program (GIMP) is an open source raster graphics editor, offering similar functionality to Adobe's Photoshop application. On May 23rd 2010, we harvested 14,559 queries relating to this software application. Analysis of the GIMP data set reveals that the terms "black" and "white" co-occur in 93 distinct queries, and in each case, the queries inquire about converting color images to black and white. According to the Google AdWords tool, the query "gimp black and white" is searched an average of 590 times a month, or about once every 74 minutes.

Inspecting GIMP's interface (version 2.6) reveals that the commands and menus relevant to performing this operation are labeled as "grayscale", "desaturate" or "channel mixer". These technical terms may not be familiar to GIMP's user base, as evidenced by the vocabulary used in the harvested queries. To provide points of comparison, the queries "gimp grayscale" and "gimp desaturate" are performed 260 and 58 times a month respectively, each less than half the number of "black and white" queries. Given this finding, one could create a "black and white" command that aggregates into one command the many methods of transforming an image into a grayscale image, perhaps using a rich previewing technique such as Side Views [23].

Clip, but not Crop

Inkscape is an open source vector graphics editor similar to Adobe's Illustrator program. On May 22nd 2010, we harvested 2,501 queries relating to Inkscape. Interestingly, the 8th highest volume query was "inkscape crop", with an average of 480 searches performed each month. However, being a vector graphics application, Inkscape does not have a "cropping" tool; cropping is specific to raster graphics. The equivalent operation for vector graphics is to "clip". This very popular query suggests that new Inkscape users are relying on Google to translate knowledge from one domain (i.e., raster graphics) to another domain (i.e., vector graphics). This behaviour closely resembles similar behaviour exhibited by programmers' use of Google [6]. Recognizing this issue, Inkscape could provide a "crop" command or help entry that assists users in setting the clip for their document.

Desired functionality

In addition to identifying potential usability issues relating to terminology, we found query log analysis to be an excellent source for understanding desired functionality.

Blocking unwanted calls

One popular class of queries relating to Apple's iPhone product inquires about the possibility of selectively blocking unwanted calls from specific telephone numbers. While this feature is not currently supported by the device, users search for information on performing this task at least 5,800 times a month, or once every 7.5 minutes. The consensus among the user community is that the issue can be resolved by associating a silent audio clip as the ringtone of unwanted telephone numbers. That this issue is so popular suggests users would be well-served if provided with a sanctioned means of achieving this same behaviour.

Changing screen savers

Another example of identifying desired functionality emerges when analyzing the searches specific to Amazon's Kindle eBook reader. Specifically, query log analysis reveals 89 distinct phrasings of the query "how to change your kindle screensaver". It turns out that the device ships with a few dozen stock images that are displayed by the device when not in use. However, these images cannot be customized by the end user. Again, the popularity of these searches suggest that such a feature would welcomed.

Draw shapes in GIMP

Finally, an analysis of the GIMP query data set reveals many queries relating to drawing primitive shapes: roughly 130 unique queries enquire about drawing various types of lines, 80 unique queries inquire about drawing circles, 40 queries inquire about drawing rectangles, 20 queries inquire about drawing squares, and 14 queries inquire about drawing ellipses. Moreover, the suggestions "gimp how to draw a line", and "gimp how to draw a circle" appear in the top 10 suggestions for the prefix "gimp how to", and the Google AdWords tool reports that the query "gimp draw circle" is performed an average of once an hour, each and every day. These queries are noteworthy because GIMP provides no explicit tools for drawing simple shapes. Dedicated tools for these functions would likely find great use by GIMP users.

The usability cost of poor affordances

As a final example of the types of problems that can be uncovered using query log analyses, we consider the usability cost of poor affordances and uninformative error messages.

Ubuntu's "authentication failure"

Ubuntu is currently one the most popular GNU/Linux distributions. For reasons of security, Ubuntu disables the "root" superuser account by default, requiring users to issue the "sudo" command to gain superuser privileges. The root account has otherwise been present and used in UNIX and UNIX-like systems for decades.

While Ubuntu's policy is arguably a positive change for security, the operating system may not be adequately com-

municating this policy to new users: attempts to log in as the root user (in Ubuntu version 10.04) simply result in an "authentication failure" error message. An analysis of the queries related to Ubuntu reveals nearly 130 distinct query phrasings all asking about how to access the root user account. The specific query "ubuntu login as root" is performed 720 times a month, or about once an hour. Similarly, the query "ubuntu root login" is searched 880 times monthly. This finding suggests that users would be well served by a more helpful or detailed error message which could communicate the proper course of action when attempting to login as the root user.

DISCUSSION

In this section, we more broadly discuss issues related to using query logs to understand the needs of users of interactive systems. We begin with a discussion of how query log analysis can factor into existing usability practices, then describe and address various issues that may affect the rankings produced by our method.

Integrating query log analysis in usability practices

Throughout the paper, we have been careful to note that query logs can be used to identify *potential* usability problems of interactive systems. While a query may *suggest* that users are experiencing difficulties with a particular aspect of the system (e.g., "gimp how to draw a circle"), further details and context are required before one can conclude the nature and severity of the issue. This additional information can be obtained using standard evaluation techniques involving users or expert evaluators. Since many methods (e.g., cognitive walkthrough) require representative tasks to be identified for evaluative purposes, query analysis can assist by supplying a ranked list of common tasks and needs.

A ranked list of common queries can also be used to assign importance to existing lists of known usability issues. The benefit of using the results of CUTS is that this ranking derives from the search behaviour of thousands, if not millions, of users. This ranked list may also be more exhaustive than existing lists tracking usability issues. Software producers with limited resources, including volunteer-driven open source products, could thus benefit from this additional means of identifying potential usability issues.

Finally, the data derived from this process can provide "hard evidence" for motivating product engineering teams to address issues relating to the usability of a device or software application.

Factors that may impact or compromise query ranking

To effectively use query analysis, it is also important that one understand and consider the various factors that can impact the weighting and ranking that such analysis produces. In this section, we discuss various effects that influence how often various searches are performed.

User search behaviours and query reformulation

A growing body of research (e.g., [15, 1]) examines user search behaviour. One of the practices observed is that people reformulate their queries when search results do not match their expectations or needs. As a result of this query reformulation strategy, it is conceivable that the analysis proposed in this paper artificially inflates the importance of particular issues. However, we note that query popularity has been observed to follow a Zipf's law distribution [22, 3]. As a result of this exponential relationship between queries and their frequency, it unlikely that any reformulation behaviours would grossly distort the query rankings. However, the effect may be more pronounced among queries with lower search volume, suggesting a need for more work in this space.

Products with generic names

A number of products have relatively generic names (e.g., Microsoft Word, Adobe Illustrator, etc.), which can cause many irrelevant or off-topic queries to appear in query logs. A similar problem is encountered by products whose names are now synonymous with a class of operations or applications. For example, an altered digital image is often described as being "Photoshopped", regardless of which software application was used for image manipulation.

In these problematic cases, we have found our filtering techniques (e.g., "how to ___ in photoshop") are often enough to filter out the less desirable, off-topic queries.

We also suspect that it is possible to differentiate between the uses of a word by analyzing the results that search engines return for those queries. Search engines are designed to return relevant documents, and often refine their relevance rankings by observing which pages users visit after performing searches [2]. The query-document associations recorded by search engines provide a wealth of untapped information that can further guide analysis of query logs. Use of these associations constitutes a promising area of future research.

CONCLUSION

When faced with difficulties or questions relating to the use of interactive systems, many people routinely turn to Internet search engines as a first line of support. In this paper, we have introduced the CUTS process (characterizing usability through search). This process takes as input the name of an interactive system and produces as output a ranked and categorized list of potential issues that users encounter with that system. These data are assembled by sampling from the query logs of top-tier Internet search engines. Importantly, the results of this process have a high degree of ecological validity, and can directly inform more formal evaluation methods by suggesting particular tasks or issues to test.

REFERENCES

1. A. Aula, R. M. Khan, and Z. Guan. How does search behavior change as search becomes more difficult? In *Proc CHI '10*, pages 35–44, New York, NY, USA, 2010. ACM.

2. R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *Proc KDD '07*, pages 76–85, New York, NY, USA, 2007. ACM.

3. Z. Bar-Yossef and M. Gurevich. Mining search engine query logs via suggestion sampling. *Proc. VLDB Endow.*, 1(1):54–65, 2008.

4. R. Barrett, E. Kandogan, P. P. Maglio, E. M. Haber, L. A. Takayama, and M. Prabaker. Field studies of computer system administrators: analysis of system management tools and practices. In *Proc CSCW '04*, pages 388–395, New York, NY, USA, 2004. ACM.

5. J. Brandt, M. Dontcheva, M. Weskamp, and S. R. Klemmer. Example-centric programming: integrating web search into the development environment. In *Proc CHI '10*, pages 513–522, New York, NY, USA, 2010. ACM.

6. J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proc CHI '09*, pages 1589–1598, New York, NY, USA, 2009. ACM.

7. A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

8. Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *Proc WWW '07*, pages 581–590, New York, NY, USA, 2007. ACM.

9. M. Gikas. Lab tests: Why Consumer Reports can't recommend the iPhone 4. *Consumer Reports*, July 2010.

10. J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457:1012–1014, February 2009.

11. Google Corperation. Features: Google suggest. `http://www.google.com/support/websearch/bin/answer.py?hl=en\&answer=106230`, 2010.

12. Google Corperation. Google AdWords keyword tool. `https://adwords.google.com/select/KeywordToolExternal?forceLegacy=true`, 2010.

13. Google Corperation. Google insight for search. `http://www.google.com/insights/search/`, 2010.

14. Google Corperation. Google suggest : Frequently asked questions. `http://labs.google.com/intl/en/suggestfaq.html`, 2010.

15. J. Huang and E. N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *Proc CIKM '09*, pages 77–86, New York, NY, USA, 2009. ACM.

16. M. Kellar, C. Watters, and M. Shepherd. A field study characterizing web-based information-seeking tasks. *J. Am. Soc. Inf. Sci. Technol.*, 58(7):999–1018, 2007.

17. J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):pp. 159–174, 1977.

18. J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proc CHI '90*, pages 249–256, New York, NY, USA, 1990. ACM.

19. W. Quesenbery, C. Jarrett, I. Roddis, S. Allen, and V. Stirling. Search Is Now Normal Behavior. What Do We Do about That. In *UPA 2008*, Baltimore, Maryland, USA, June 2008.

20. M. Richardson. Learning about the world through long-term query logs. *ACM Trans. Web*, 2(4):1–27, 2008.

21. D. E. Rose and D. Levinson. Understanding user goals in web search. In *Proc WWW '04*, pages 13–19, New York, NY, USA, 2004. ACM.

22. P. C. Saraiva, E. Silva de Moura, N. Ziviani, W. Meira, R. Fonseca, and B. Riberio-Neto. Rank-preserving two-level caching for scalable search engines. In *Proc SIGIR '01*, pages 51–58, New York, NY, USA, 2001. ACM.

23. M. Terry and E. D. Mynatt. Supporting experimentation with side-views. *Commun. ACM*, 45(10):106–108, 2002.

24. T. Zeller. AOL technology chief quits after data release. *The New York Times*, August 2006.