Leveraging Crowdsourced Technical Documentation: Building a Command Thesaurus

Adam Fourney
afourney@cs.uwaterloo.ca

Michael Terry
mterry@cs.uwaterloo.ca

David R. Cheriton School of Computer Science
University of Waterloo

ABSTRACT
Since its inception, the Internet has enabled motivated members of an application's user base to compose and self-publish technical documentation, manuals and tutorials. These distributed acts of self-publishing can be thought of as the implicit crowdsourcing of technical support. In this paper, we leverage user-generated documentation to construct what we call a "command thesaurus". A command thesaurus groups together semantically related words, bridging the gap between the vocabulary expressed by users and the (sometimes highly technical) terminology employed by software applications. In this work, we outline one potential approach for the automatic generation of a command thesaurus, and we present some initial experiments suggesting that the proposed approach is feasible. We then conclude by describing various compelling applications of these newly generated resources. In particular, command thesauri may find use in search-driven interfaces, and in tools that translate tutorials from one application to another.

Author Keywords
Crowdsourcing, Technical Documentation, Command Thesaurus

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous

General Terms
Human Factors

INTRODUCTION

In a very real sense, application help and support has been implicitly crowdsourced on the web since the web's inception: People routinely make tutorials and how-to videos available online, and they ask and answer questions on product-specific forums. The need for this organic, community-driven support is clear: Software producers cannot anticipate the wide range of problems users will attempt to solve with their software, or all of the different types of users who may wish to use it. Thus, user-generated help, tutorials, and support fill an important niche in the software ecosystem.

While user-generated help and support address crucial needs, they are not without their problems. There is considerable variability in vocabularies and terminology between users, there is no centralized organizational structure to this user-generated help, and user-generated help is not integrated with the software itself (though recent research efforts are moving in this direction [3]). User-generated help and support can thus be considered a crowdsourcing success, but a success with very clear limitations.

The phenomenon of user-generated help and support is important because it provides a glimpse of the types of problems and opportunities that are likely to arise as other forms of crowdsourcing and human computation become more commonplace. In this paper, we focus on the specific problem of variable vocabularies between parties, and the overhead this variability introduces in crowdsourcing or human computation contexts. More specifically, we are interested in reducing the time and effort it takes for users to express their needs in ways that third parties can clearly understand.

To address the problem of bridging disparate vocabularies in crowdsourcing and human computation contexts, we introduce the notion of a *Command Thesaurus*, a graph that connects alternate representations of the same concept with one another. In our initial formulation of this concept, we focus on the problem of matching users' vocabularies with that of a given application (making the software the "third party"). For example, users of the Inkscape vector graphics editor often want to "crop" their images. However, Inkscape offers this functionality via the "clip" command. A Command Thesaurus would provide a bridge between the user's conceptualization of their goal, and the actual command set provided by the application itself. Importantly, while we ground this paper in this specific problem space (the user communicating with the application), the concepts put forth should be equally applicable to other crowdsourced and human computation tasks in which people are connecting with other people.

In the rest of this paper, we develop this notion of a Command Thesaurus, provide examples of its application in in-

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

stitching images in gimp	gimp invisible color	gimp text along path color
gimp stitch photos	gimp make area transparent	gimp curving text
stitching panoramas in gimp	gimp translucency	curve text in gimp
...	...	...
how to make panoramic photos in gimp	gimp how to use alpha channel	how to bend text in gimp

Table 1. Three search query clusters related to the GIMP software application. Despite the wide range of terminology expressed in these queries, the clustering algorithm is able to identify common themes.

terface design, and argue for its overall utility in other contexts.

THE COMMAND THESAURUS

In [4] we argued that users routinely rely on Internet search engines to support their use of interactive systems, and we demonstrated that an analysis of search query logs can reveal the primary tasks and needs of a product's user population. In this work, we found that two of the most prolific categories of user queries involve issues related to feature discoverability, and problems of mismatched vocabulary (the interface using terminology that the user is not familiar with). Importantly, query logs provide an excellent view of the vocabulary and terminology with which users conceive their use of interactive systems. An example illustrates this point.

On May 23rd, 2010, we used the CUTS technique [4] to collect samples from Google's query logs, revealing 14,559 distinct queries relating to the GNU Image Manipulation Program (GIMP). Analysis of those queries reveals the search "gimp how to make black and white" is quite common. GIMP (version 2.6) offers at least three alternative methods for converting a color image to black and white. These methods are labeled as "grayscale", "desaturate", and "channel mixer". Such technical terms may not be familiar to a sizeable portion of GIMP's user base, as evidenced by the vocabulary used in search queries.

Importantly, users performing the search "gimp how to make black and white" will find that Google returns webpages which mention the term "grayscale" at a rate which is much higher than can be attributed to chance alone. As such, we may associate "make black and white" and "grayscale" to one another, and record this relationship in GIMP's command thesaurus. Newer version of GIMP could then leverage this thesaurus, perhaps augmenting the tooltip for the grayscale command to include a list of related terminology including "make black and white". This would help users to better identify the commands needed to perform their desired tasks.

In the remainder of this section, we formalize this approach to constructing command thesauri, we present results from early experiments, and we discuss how a command thesaurus might be leveraged from within an application.

Constructing a command thesaurus

To build a command thesaurus, we must link different vocabularies describing the same concept. In our specific instantiation of this problem, we are interested in associating the vocabulary of search queries with the vocabulary expressed by interface constructs. As noted earlier, the vocabulary expressed in searches can be extracted directly from search query logs, while the vocabulary expressed by a GUI can be extracted from the software localization dictionaries which are used by software to support multiple languages. Building a command thesaurus involves discovering how the terms of each vocabulary are related.

Previous work by Baeza-Yates *et al.* describes how semantic relations between search terms can be inferred by relating search queries to one another through observations of which pages users visit after performing searches [1]. The idea is that, if a user elects to visit a search result after performing a query, they are implicitly indicating that the chosen document is relevant to the original query. In fact, search engines often use this feedback to actively refine their relevance rankings of documents. The approach taken by Baeza-Yates *et al.* is to express these query-document relations in a weighted bipartite graph (queries occupy one partition, while documents occupy the other). The structure of this graph is then analyzed to infer various relationships between search terms.

Building upon these ideas, we can relate query terms to software commands as follows: We again create a weighted bipartite graph; but, in this graph, queries occupy one partition, and *commands* occupy the other. When a user submits a query to the search engine and elects to visit a particular search result, we associate the query terms with the commands referenced in the visited webpage. As with the work of Baeza-Yates, we may then analyze the structure of the resulting graph to reveal clusters of tightly related terminology and commands. These clusters can populate the "command thesaurus".

We have already begun performing experiments following an approximation of the above process. We describe the promising results of these early experiments in the next section.

Early experiments and results

In order to explore the possibility of using the proposed approach for constructing a command thesaurus, we performed a number of experiments using a simplified version of the protocol. Specifically, for input we used the aforementioned May 2010 GIMP query data set, which does not yield information regarding which pages users visit after performing searches. To compensate for this missing data, we simply assume that each user eventually visits the top 3 search results for each query. We note that the strategy is surprisingly ef-

fective, and has been used extensively in past research (e.g., [2, 6]). When we apply this technique to the GIMP data set, we find many interesting clusters of related terminology. Three example clusters are listed in Table 1.

The initial results are promising, and suggest that the process of generating command thesauri can be heavily automated. Automation enables command thesauri to be updated continuously as new user-generated documentation is made available online, and as trends in application usage vary with time. This enables a wide range of applications which we describe in the next section.

Applications

Once a command thesaurus is available, it enables a wide range of novel affordances and interactions which may improve feature discoverability in software applications. The example noted earlier in this document, where an application's tooltips are automatically augmented with related terminology, is perhaps the simplest use of a command thesaurus. We believe that such tooltips would allow users to identify commands they might be interested in using, even if they do not recognize those commands by name. We note that the notion of augmenting tooltips with community generated data is not new (e.g., [5]); but, we believe our approach benefits from its use of timely data mined from thousands of web pages and search queries.

Augmented tooltips are a passive means for helping users identify the commands they might be interested in performing. Command thesauri also enable far more active assistance. One particularly compelling vision of an active approach is the use of command thesauri in search-driven interfaces. In a search-driven interface, users have the option of typing in the description of the task they would like to perform, and the interface responds by presenting the user with a list of relevant commands, options, or tool pallets. Such search algorithms could be directly backed by the command thesaurus, or at the very least, the command thesaurus could be leveraged by the search algorithms to improve overall result rankings.

Finally, it may be possible to relate the command thesaurus of one application to the command thesaurus of a similar application in order to help translate commands from one application to the other. For example, this could help users of Photoshop complete tasks using GIMP, and vice versa. A more intriguing possibility is that such associations could enable translations of tutorials created for one application so that they can be performed in the other. Importantly, creating these associations between command thesauri might be as simple as identifying commands in each thesaurus that give rise to similar user queries.

GENERALIZING TO OTHER CROWDSOURCED AND HUMAN COMPUTATION APPLICATIONS

While our initial experiments involve search queries and application commands, it should be obvious that the technique is likely to generalize to crowdsourced and human computation applications. In these cases, there is no real "command set," but, instead, a set of natural language tasks and communications between a user and a worker. With suitable instrumentation and logging of these processes, one should be able to achieve similar results when linking users' conceptualizations of tasks with those of the workers.

REFERENCES

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

2. Bernstein, M. S., Suh, B., Hong, L., Chen, J., Kairam, S., and Chi, E. H. Eddi: interactive topic-based browsing of social status streams. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, ACM (New York, NY, USA, 2010), 303–312.

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

4. Fourney, A., Mann, R., and Terry, M. Characterizing the usability of interactive applications through query log analysis. In *Proc CHI '11*, ACM (New York, NY, USA, 2011).

5. Matejka, J., Li, W., Grossman, T., and Fitzmaurice, G. Communitycommands: command recommendations for software applications. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, ACM (New York, NY, USA, 2009), 193–202.

6. Shen, D., Pan, R., Sun, J.-T., Pan, J. J., Wu, K., Yin, J., and Yang, Q. Q2c@ust: our winning solution to query classification in kddcup 2005. *SIGKDD Explor. Newsl. 7* (December 2005), 100–110.