Facilitating End-User Developers by Estimating Time Cost of Foraging a Webpage

Xiaoyu Jin, Nan Niu
Department of Electrical Engineering and Computer Science
University of Cincinnati
Cincinnati, OH, 45221, USA
jinxu@mail.uc.edu, nan.niu@uc.edu

Michael Wagner
Division of Biomedical Informatics
Cincinnati Children's Hospital Medical Center
Cincinnati, OH, 45229, USA
michael.wagner@cchmc.org

***Abstract*—During programming, end-user developers constantly go to search engines to seek for information. The search engine is of significant help since it ranks the webpage links according to relevance. However, the time cost of foraging a webpage also affects if and how soon a developer can obtain a satisfying answer. In this paper, we use operationalizable constructs from Information Foraging Theory to identify two features: information accumulation and information amount for a webpage, which we hypothesize could assist developers in selecting appropriate webpages. We then invited 20 participants to perform a lab experiment of two software change tasks. The results supported our hypothesis by two findings. When having the tool support, the participants used less task completion time, and tended to visit more easy-to-forage webpages.**

I. Introduction

Developers, both novices and experts alike, constantly use web search engine as an opportunistic approach to programming, emphasizing speed and ease of development over code robustness and maintainability [3]. The search engine is of significant value since it not only filters oceans of information, but also ranks webpage links according to relevance. However, situation exists that a developer was sure that he selected a webpage containing the needed information but it turned out that he was overwhelmed by the large amount of info in the webpage. This situation reflects that having only relevancy may not be enough in identifying an optimal webpage link. The cost of foraging a webpage can provide supplementary information which could prevent developers from jumping into trouble such as being overwhelmed and lost focus.

Recently, Piorkowski *et al.* [21] used Information Foraging Theory [26] to study developers' ability to predict the value and cost of their investigation decisions. They found that over 50% of developers' navigation choices produced less value than they had predicted and nearly 40% cost more than they had predicted [21]. Their study revealed open problems in predicting the value and cost of navigation decisions. Related to our study, they pointed out the cost estimation problem regarding how to enable developers to more accurately predict the foraging cost they will incur before they incur them [21].

Motivated by Piorkowski *et al.*'s [21] work, we focus on analyzing time cost of foraging a webpage. We identify two factors of information accumulation and information amount for a webpage, which could assist developers in selecting appropriate webpage links. Our overall hypothesis is that by comprehensively considering the webpage's relevancy, information accumulation, and information amount, developers can find useful information easier and faster.

In this paper, we develop tool support to automatically extract two features for a webpage: information accumulation, and information amount. To evaluate our tool support, we invite 20 end-user developers to perform a lab experiment of two software change tasks. The results validated our hypothesis by two findings. When having the tool support, the participants used less task completion time, and tended to visit more easy-to-forage webpages. The key contribution of our work lies in the identification of two hints: information accumulation and information amount for a webpage, which could facilitate end-user developers' web search process.

II. Background and Related Work

In terms of web navigation and predication, the work by Chi and Priolli *et al.* [4] has significant contribution, especially the models of WUFIS (Web User Flow by Information Scent) [6] and SNIF-ACT (Scent-based Navigation and Information Foraging in the ACT architecture) [24]. They use information scent construct from Information Foraging Theory to perform rational analysis on users' navigational behavior [5], [26]. They develop architecture and system for the analysis and prediction of user behavior and web site usability, which provides significant value in applications of personalized web environments, web site design, and help identify parts of a web site as bad designs [5]. Our work in this paper is supplemental to their work in two perspectives: (1) we aim to serve web users to help them navigate efficiently, while their work mainly aim to assist web designers to identify bad design and develop better website; (2) we consider reducing cost to achieve higher efficiency, while their strategy mainly concerns about the value of contents and information scents on webpage.

Holmes *et al.* [8] described their approach to finding source code examples in which the structure of the source code that the developer is writing is matched heuristically to a repository of source code. Ten examples are returned to the users and ranked according to structural similarity based on four heuristics: inherits, calls, uses (a relaxation of the calls heuristic), and

978-1-5386-0443-4/17/$31.00

references. The ranking and the returned knowledge of graphical overview and textual rationale description can be used by developers to quickly decide whether the recommended example is worth examining more closely thereby achieving cost evaluation and reduction. A few previous studies consider humans as a valuable resource. Minto and Murphy [17] introduced Emergent Expertise Locator (EEL) that uses emergent team information to propose experts to a developer within their development environment as the developer works. DeLine *et al.* [7] introduced Team Tracks that rely on human further to use the combined data across all team member. While this tool is not to recommend members, it uses the team's navigation data to filter the typical hierarchical information about the program. To summarize, these studies mainly focus on the cost estimation and reduction for the source code level activities. Our study is supplementary to these efforts since our study focuses on the evaluation and reduction for the cost of foraging webpages, which contain highly diverse information.

III. Our Approach

Our overall hypothesis is that by comprehensively considering the webpage's relevancy, information accumulation, and information amount, developers can find useful information easier and faster. Since relevancy has already been provided by the search engines, our approach focuses on the remaining two features of information accumulation and information amount.

The first feature of information accumulation is adopted from Information Foraging Theory, which was originally inspired by appeals in the psychology literature for an ecological approach to understanding human information-gathering and sense-making [23]. Pirolli [23] laid out the basic analogies between food foraging and information seeking: predator (human in need of information) forages for prey (the information itself) along patches of resources and decides on a diet (what information to consume and what to ignore). The theory has been successfully applied in areas of software development [2], [13], [14], [15], [20], code navigation [9], [12], [18], [19], and website design and evaluation [6].

The patch model is a core component of information foraging theory. For instance, imagine a bird that forages for berries found in patches on berry bushes. A forager first needs to expend some between-patch time getting to the next food patch. Once in a patch, the forager needs within-patch time to forage food and also needs to decide when to stay or leave this patch for the next one [25]. A patch can be constructed differently such as a book, a webpage, a source code file, a code fragment, or even a line of code, etc. As shown in Fig. 1, the Charnov's marginal value theorem [25] depicts that as the foraging time within the patch increases, the cumulative amount of useful information (represented as $g(t_w)$) gained from the patch increases. The curve is increasing but with a decreasing speed based on the assumption that there will be diminishing valuable information gained in a patch as time progresses. The assumption is based on the observations that forager generally will prioritize to forage more valuable information and forager may get redundant information in later

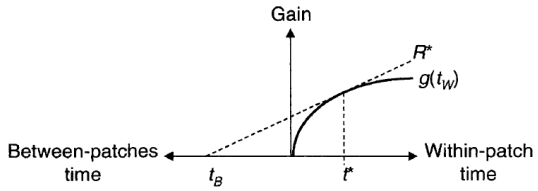

Fig. 1: Charnov's marginal value theorem adopted from [25]: the rate-maximizing time to spend in patch t^* occurs when the slope of the within-patch gain function, g, is equal to the average of gain, which is the slope of the tangent line R. "t_B" and "t_W" represent between patch time and within patch time respectively.

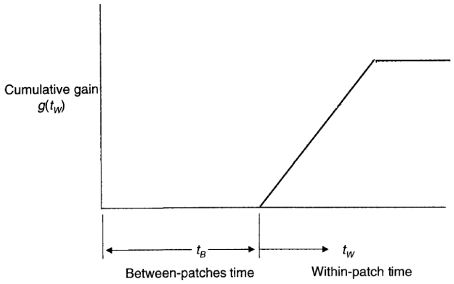

Fig. 2: Foraging curve adopted from [25]: A linear, finite cumulative within-patch gain function.

time which replicates information encountered earlier [25]. Further, Charnov's marginal value theorem was developed to predict that in order to achieve the maximum rate of foraging information, a forager should remain in a patch so long as the slope of $g(t_w)$ is greater than the average rate of gain, R, for the environment, as shown in Fig. 1.

However, when applying the patch model to the webpages in our study, we found a problem that the foraging curves for webpages are not always like the shape in Fig. 1, which increases with a decreasing speed, because the structures and organizations of webpages could vary significantly. The patch model assumed that the forms of patches are more or less similar to each other, which is not the case for webpages. Our observation is validated in Pirolli and Card's paper [25] that they described an example whose foraging curve is linearly increasing as shown in Fig. 2. The example depicted that an information forager who collects relevant citations from a finite list of citations returned by a search engine, where the relevant items occur randomly in the list.

Following the observation, we further analyzed more webpages and found two additional shapes of foraging curve and summarized the foraging curves into four categories. The first category was depicted in Fig. 1, meaning that the contents of a webpage is ranked according to certain standard. Q&A website is a typical example such as Stack Overflow, Quora, Answers.com, and Yahoo Answers, because these webpages ranked the answers according to users' voting. We name the first category as Ranked Foraging. The second category was depicted in Fig. 2, indicating that the forager can linearly get information from a webpage. The webpages containing listed items belong to the second category. For instance, an API

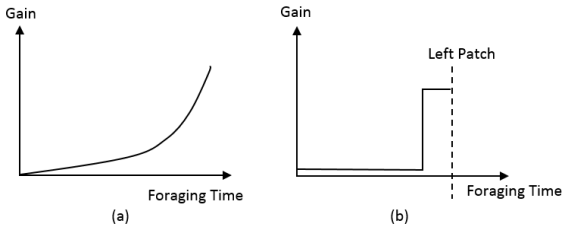

Fig. 3: Foraging curves: (a) Conceptual Foraging; (b) Answer Seeking Foraging.

website for a software system, which lists the functions can be used. The number of functions learned is roughly linear to the time spent. We name the second category as Linear Foraging. The third and fourth categories are shown in Fig. 3-(a) and Fig. 3-(b) respectively. The third category represents the situation when the goal of the forager is not to seek for information, but to study certain concept. The knowledge learned is accumulated gradually and exponentially, and forager may get to a tipping point to achieve an in-depth understanding. Wiki and blog websites could be examples for such foraging goal because they aims to convey knowledge comprehensively. We name this category as Conceptual Foraging. The fourth category represents the situation that the forager is seeking a piece of specific information from a webpage. Before he found the answer, the gain was always near to zero. For instance, in a forum webpage, there are posts from many people discussing a topic. The forager scans the webpage and skips the unrelated discussions to his needs, and when he finds the right answer and solves his problem, he would leave the webpage without continuing reading the remaining contents. We name this category as Answer Seeking Foraging. We summarized the four categories of foraging curves as a feature named information accumulation.

Another tool feature is to quantify the amount of information in a webpage. We use number of words on a webpage to estimate the amount of information since word is the most important carrier for information. People can generally read about 300 words in a minute [1]. Dividing the word count by this speed, we obtain the approximate time needed to forage a webpage. Having the two features, we then developed tool support which will extract and return features of information accumulation and information amount as shown in Fig. 4. Fig. 4 displays the first five result links returned by Google search engine when searching "java double to string". From the information accumulation and information amount information in Fig. 4, first link and third link are two good options to forage because the first link contains ranked information and the third link contained the least information, which requires only 0.5 minutes approximately to forage. This example can somewhat reflect the value of our tool support, which is to have another layer of information filtering besides relevancy.

IV. Evaluation Design

To evaluate our tool support and hypothesis, we performed a lab experiment. Our independent variable was our designed

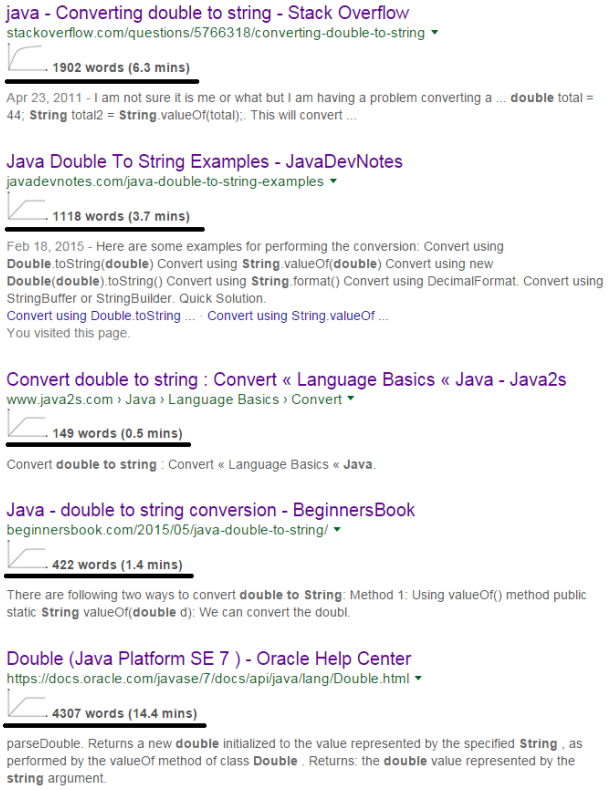

Fig. 4: An example result with our tool support, which shows the first five results links when searching "java double to string" in Google.

tool support that we wanted to test in a controlled manner. We named our tool as CostEstimator. We select bioinformatics researchers who develop biomedical software as target group of end-user developer [11]. Twenty participants took part in our experiment. These participants were recruited from the local community via email invitations. To be eligible to participant in our experiment, each individual had to consider writing software as an essential (as opposed to accidental) part of their work. Our participants had a varied background: 13 had no professional software development experience, 1 had less than a year professional experience, 3 had 1-5 years, and 3 had more than 5 years.

The participants were asked to perform two software change tasks with direct biomedical relevance. The reason for using software change tasks is because participants mainly perform software change tasks in their daily work. The tasks were designed based on the intent to best simulate bioinformatics researchers' actual programming tasks. For each task, an open-source software acted as the target system where the software change was expected to take place. The tasks were named after the software as ImageJ and StochKit respectively [11]. The participants worked individually in a lab and began by signing the consent form and completing a background survey. Each participant performed one task with CostEstimator and the other without it. We counterbalanced both CostEstimator-treatment order and the task order.

TABLE I: Comparison of task completion time: "Median" means the median task completion time, "SD" represents standard deviation.

Task	Without CostEstimator Median (SD)	With CostEstimator Median (SD)
ImageJ	60 min (3.4)	47.5 min (2.7)
StochKit	74.8 min (4.5)	68.3 min (4.7)

TABLE II: Distribution of webpages according to the four categories of foraging curves. The data are averaged from 20 participants.

Categories	ImageJ-without	ImageJ-with	StochKit-without	StochKit-with
Total	32.6	25.8	20.6	16.7
Ranked	10.7 (33%)	10.9 (42%)	6.9 (33%)	7.4 (44%)
Linear	9.2 (28%)	5.6 (22%)	5.3 (26%)	3.8 (23%)
Answer Seeking	8.8 (27%)	6.2 (24%)	6.1 (30%)	3.4 (20%)
Conceptual	3.9 (12%)	3.1 (12%)	2.3 (17%)	2.1 (13%)

V. Results and Analysis

When performing the software change tasks, the computer screen was recorded as a video for each participant. Our results are generated by analyzing these videos. The result of task completion time for the 20 participants is presented in Table I. Generally speaking, developers spent less time on the ImageJ task than the StochKit task. When the median completion time is compared, the tool support of CostEstimator facilitated both tasks to be finished faster. For ImageJ task, it is a 20.8% reduction from 60 to 47.5 minutes, and for StochKit task, it is an 8.7% reduction from 74.8 to 68.3 minutes. However, the effect is statistically significant only on the ImageJ task (Wilcoxon signed rank test: p=0.0020, α=0.05) but not on the StochKit task (Wilcoxon test: p=0.1235, α=0.05). We speculate this is because the complexity of StochKit task lies more in the original source code comprehension, and the function that the participants need to write is relatively simpler than ImageJ task. For the StochKit task, the participants need less external information from websites.

To study what changes our tool has brought about, we calculated the distribution of webpages according to the four categories of foraging curves. In specific, for each participant, we analyzed all the webpages that were accessed. We then partitioned the webpages into two groups by checking whether a webpage is found with or without tool support. Then we classified these webpages according to the four categories of foraging curves. We count the number of webpages for each category per participant, and then calculated the average for each category. Table II summarizes the results, which are the distribution of the number of webpages in the four categories of foraging curves. We can see a general trend that when performing tasks with the tool CostEstimator, the number of visited webpages is reduced. Our further analysis found that this is because participants go to fewer webpages that are less useful to their task with the tool support.

In order to compare the data generated having or not having tool support, we transformed the data into percentage data (Table II and generated Fig. 5. From Fig. 5, for both ImageJ and StochKit tasks, when having tool support from CostEstimator,

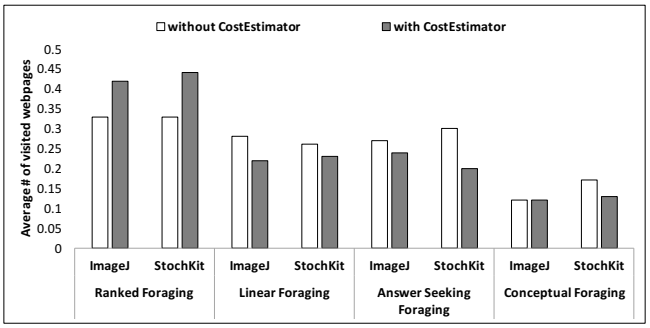

Fig. 5: A trend shifting to the Ranked Foraging category when having tool support.

participants used more webpages from the Ranked Foraging category, and used less webpages from categories of Linear Foraging and Answer Seeking Foraging. Our further analysis reveals the reason. We found that participants have more questions about the details of implementations than the conceptual level question, which is also reflected in Fig. 5 that only a few webpages used are in category of Conceptual Foraging. Therefore, having the tool support will lead participants to access more webpages from Ranked Foraging, from which answers can be easily found. This observation explains how our tool helps improve the efficiency from one perspective.

VI. Discussion

We identified the foraging cost as an important supplement to relevancy regarding the selection of webpages. However, this observation is not limited to selection of webpages. Other kinds of information patches, such as source code files in IDE, tutorial documents, books, etc., may also require cost estimation. There should be multiple ways to model and estimate the cost of foraging an information patch. Our approach of foraging curve and information amount is only one attempt to respond to the call for actions in [21]. We also found that developers prefer to access webpages belonging to Ranked Foraging category. From this viewpoint, we can collectively consider the information needs [10], [11] with the information accumulation to provide better suggestions to developers.

From the search engine perspective, our study helps generate an idea of advanced search engine: (1) search engine should not limit to the accuracy as the only ranking factor, but should consider other factors such as cost, authors, ratings, and recency, etc. [16]; (2) user should have the power to select which ranking strategy to use according to their specific foraging goal. We speculate that different foraging goals can alter the forager's strategies and behavior: a forager wanting a best answer probably values ratings; a forager wanting only a working solution probably values minimizing the cost; a forager wanting an authoritative answer may value who wrote the content.

Acknowledgment

This research is supported by the U.S. National Science Foundation (Award CCF-1350487).

REFERENCES

[1] T. Bell, 2001. Extensive reading: Speed and comprehension. *The Reading Matrix*, 1(1), pp.1-13.

[2] T. Bhowmik, N. Niu, W. Wang, J.R.C. Cheng, L. Li, and X. Cao, 2016. Optimal group size for software change tasks: A social information foraging perspective. *IEEE transactions on cybernetics*, 46(8), pp.1784-1795.

[3] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S.R. Klemmer, 2009, April. Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1589-1598). ACM.

[4] E.H. Chi, P. Pirolli, and J. Pitkow, 2000, April. The scent of a site: A system for analyzing and predicting information scent, usage, and usability of a web site. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 161-168). ACM.

[5] E.H. Chi, P. Pirolli, K. Chen, and J. Pitkow, 2001, March. Using information scent to model user information needs and actions and the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 490-497). ACM.

[6] E.H. Chi, A. Rosien, G. Supattanasiri, A. Williams, C. Royer, C. Chow, E. Robles, B. Dalal, J. Chen, and S. Cousins, 2003, April. The bloodhound project: Automating discovery of web usability issues using the InfoScentπ simulator. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 505-512). ACM.

[7] R. DeLine, M. Czerwinski, and G. Robertson, 2005, September. Easing program comprehension by sharing navigation data. In *IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 241-248). IEEE.

[8] R. Holmes, R.J. Walker, and G.C. Murphy, 2006. Approximate structural context matching: An approach to recommend relevant examples. *IEEE Transactions on Software Engineering*, 32(12), pp. 952-970.

[9] X. Jin and N. Niu, 2017, May. Short-term revisit during programming tasks. In *Proceedings of the 39th International Conference on Software Engineering Companion* (pp. 322-324). IEEE Press.

[10] X. Jin, N. Niu, and M. Wagner, 2016, November. On the impact of social network information diversity on end-user programming productivity: a foraging-theoretic study. In *Proceedings of the 8th International Workshop on Social Software Engineering* (pp. 15-21). ACM.

[11] X. Jin, C. Khatwani, N. Niu, M. Wagner, and J. Savolainen, 2016, June. Pragmatic software reuse in bioinformatics: How can social network information help?. In *International Conference on Software Reuse* (pp. 247-264). Springer International Publishing.

[12] A.J. Ko, B.A. Myers, M.J. Coblenz, and H.H. Aung, 2006. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Transactions on software engineering*, 32(12).

[13] J. Lawrance, R. Bellamy, M. Burnett, and K. Rector, 2008, April. Using information scent to model the dynamic foraging behavior of programmers in maintenance tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1323-1332). ACM.

[14] J. Lawrance, C. Bogart, M. Burnett, R. Bellamy, K. Rector, and S.D. Fleming, 2013. How programmers debug, revisited: An information foraging theory perspective. *IEEE Transactions on Software Engineering*, 39(2), pp.197-215.

[15] J. Lawrance, M. Burnett, R. Bellamy, C. Bogart, and C. Swart, 2010, April. Reactive information foraging for evolving goals. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 25-34). ACM.

[16] C. Martos, S.Y. Kim, and S.K. Kuttal, 2016, September. Reuse of variants in online repositories: Foraging for the fittest. In *IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 124-128). IEEE.

[17] S. Minto and G.C. Murphy, 2007, May. Recommending emergent teams. In *Proceedings of the Fourth International Workshop on Mining Software Repositories* (pp. 33-40). IEEE.

[18] N. Niu, X. Jin, Z. Niu, J.R.C. Cheng, L. Li, and M.Y. Kataev, 2016. A clustering-based approach to enriching code foraging environment. *IEEE transactions on cybernetics*, 46(9), pp.1962-1973.

[19] N. Niu, A. Mahmoud, and G. Bradshaw, 2011, May. Information foraging as a foundation for code navigation (NIER track). In *Proceedings of the 33rd International Conference on Software Engineering* (pp. 816-819). ACM.

[20] N. Niu, A. Mahmoud, Z. Chen, and G. Bradshaw, 2013, May. Departures from optimality: Understanding human analyst's information foraging in assisted requirements tracing. In *Proceedings of the 2013 International Conference on Software Engineering*(pp. 572-581). IEEE Press.

[21] D. Piorkowski, A.Z. Henley, T. Nabi, S.D. Fleming, C. Scaffidi, and M. Burnett, 2016, November. Foraging and navigations, fundamentally: Developers' predictions of value and cost. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 97-108). ACM.

[22] P. Pirolli, P., 2005. Rational analyses of information foraging on the web. *Cognitive science*, 29(3), pp.343-373.

[23] P. Pirolli, 2007. *Information foraging theory: Adaptive interaction with information.* Oxford University Press.

[24] P. Pirolli, W.T. Fu, E. Chi, and A. Farahat, 2005, July. Information scent and web navigation: Theory, models and automated usability evaluation. In *Proceedings of HCI International*.

[25] P. Pirolli and S. Card, 1999. Information foraging. *Psychological Review,* 106(4), pp. 643-675.

[26] P. Pirolli, and S. Card, 1995, May. Information foraging in information access environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 51-58). ACM Press/Addison-Wesley Publishing Co..