Investigating How to Effectively Combine Static Concern Location Techniques *

Emily Hill
Dept. of Computer Science
Montclair State University
Montclair, NJ 07043
hillem@mail.montclair.edu

Lori Pollock and K. Vijay-Shanker
Dept. of Computer and Information Sciences
University of Delaware
Newark, DE 19716
{pollock, vijay}@cis.udel.edu

ABSTRACT

As software systems continue to grow and evolve, locating code for maintenance tasks becomes increasingly difficult. Studies have shown that combining static global concern location techniques like search with more structure-based local techniques can improve effectiveness. However, no studies have yet investigated *why* this occurs. In this paper, we investigate why combining global and local techniques improves effectiveness, and under what conditions. We explore such questions as: "What are the limits of lexical information in locating concerns?", "How far away does a local technique have to go to locate the remaining relevant elements?", and "How sensitive are these results to the query or scoring thresholds of the techniques?". The results of our study can inform design decisions to maximize effective global and local combinations in future concern location techniques.

Categories and Subject Descriptors: D.2.7 [Software Engineering]: Distribution, Maintenance, & Enhancement—*reverse engineering*

General Terms: Human Factors, Reliability

Keywords: Software maintenance, concern location, source code search, source code exploration

1. INTRODUCTION AND BACKGROUND

Research has shown that developers spend more time finding and understanding code than making modifications during maintenance [3]. Thus, we can reduce maintenance costs by helping developers to more effectively find the code relevant to their maintenance tasks. The code relevant to a maintenance task typically involves one or more *concerns*. A concern is anything stakeholders of the software consider to be a conceptual unit, such as features, requirements, design idioms, or implementation mechanisms [11]. Locating a concern in source code can be especially difficult if the relevant code is scattered throughout the system, rather than located in a single file [4].

Approaches to concern location typically rely on static and/or dynamic information. Dynamic information is collected by executing the source code, based on the idea that methods executed by a test suite are likely to be relevant to the concern [1, 9]. In contrast, static approaches locate program elements using only information from the source code. Because source code is consistently available for all software projects, even for legacy or incomplete codes that cannot be executed, we focus on static concern location approaches. Static concern location approaches can be further categorized by the type of source code information they use: global or local.

Global Techniques. Global approaches consider all possible source code elements, determining the most relevant by some criteria. Existing global static techniques include ranking each element by some structural criteria, such as the HITS web mining algorithm [9, 12], or search tools. Search tools predominantly use lexical information to determine relevance to a user-supplied query, often using sophisticated information retrieval techniques to improve results [6, 7].

Existing structural ranking criteria have been used in conjunction with other types of information, such as dynamic execution traces or search techniques [1, 9]. Since structural-based rankings are independent of the target concern (i.e., they use no query or test suite to focus on relevant source code), they are ill-suited to be a stand-alone global concern location technique. Thus, we focus our attention on lexical searching as our static global technique.

A global search technique can locate relevant elements across a code base, irrespective of dominant decomposition [14], and thus can perform well even when locating concerns that cut across file and package structures. However, search techniques are incapable of locating relevant code that does not contain lexical clues. Furthermore, global techniques require the user to enter a query describing the target concern, guessing what words the original developers used to implement the code. An ideal query will find many relevant results (i.e., have high *recall*), with few irrelevant results (i.e., with high *precision*). This can be challenging since the ideal query depends not only on finding the right words to describe the concern (and achieve high recall), but determining which of those words will discriminate between relevant and irrelevant results (thus increasing precision).

Local Techniques. In contrast to global approaches, local approaches start from a set of presumed relevant elements and recommend additional elements structurally lo-

*This material is based upon work supported by the National Science Foundation under a Graduate Research Fellowship and Grant No. CCF-0702401 and CCF-0915803.

SUITE '11, May 21–28, 2011, Waikiki, Honolulu, HI, USA

cated nearby that are likely to be relevant to the concern. Existing static local approaches include purely structural-based techniques [1, 10, 12] as well as lexical-based techniques [2]. Local techniques can potentially overcome some of the limitations of global search techniques. For example, structural-based local techniques can find relevant elements without lexical clues. Lexical-based local techniques can avoid the challenge of finding a query by leveraging words found in the initial set of program elements. However, local approaches also have limitations that global techniques do not. Specifically, local approaches are limited by the model of structural information used, i.e., local approaches will miss relevant results connected by relationships not modeled by the structural information. For example, if data dependencies are too expensive to calculate, a local approach will miss connections between elements that add and remove items from an event queue. Furthermore, local techniques require a starting set of relevant elements. If the developer is familiar enough with the code to create such a starting set, then the developer is unlikely to need the assistance of a combined concern location technique. During the concern location process, local techniques are useful as a developer refines his understanding of the code, rather than as he is just beginning to locate code.

2. COMBINING GLOBAL AND LOCAL

In a previous evaluation using search, dynamic information, and structural information for concern location, Eaddy et al. [1] demonstrated that combining global information from a search technique and a local, structure-based technique can improve effectiveness over search alone. Further concern location studies have shown that combining sources of information can be more effective, but depends on balancing precision and recall by selecting appropriate thresholds on the number of results returned [8]. In this paper, we present an empirical investigation of how global and local techniques can be effectively combined for concern location. We supplement Eaddy et al.'s existing work [1] by investigating trade-offs in precision and recall to inform design decisions in future concern location techniques. We extend Ratanotayanon et al.'s study [8] by showing that local and global thresholds can be selected to outperform global alone.

Open Research Questions

Rather than evaluate specific global and local techniques, our goal is to analyze the limits of the information predominantly used by each type of technique, since they are complementary. We investigate the following key open research questions:

- How do the trade-offs in terms of precision and recall at various global and local thresholds inform design decisions to maximize effective combinations?
- What is the role of lexical and structural information in achieving high recall and precision in the overall concern location process?

Specifically, we explore:

- Is combining local and global techniques more effective than global alone? Under what conditions? What are the trade-offs involved in terms of precision and recall?
- What are the limits of lexical information? How high a recall can we achieve with just lexical information and is structural information necessary to achieve high recall?

These are our primary questions of interest. However, to better interpret and generalize from these results, we must first investigate the following two preliminary questions:

- How sensitive are the results to the query?
- What are the trade-offs in terms of precision and recall at various global thresholds?

Data to Investigate Open Questions

Global Technique. As with prior work [1], we use the common tf-idf scoring function [5] to score an element's relevance to the query. Tf-idf multiplies two component scores together: term frequency (tf) and inverse document frequency (idf). The intuition behind tf is that the more frequently a word occurs in an element, the more relevant the element is to the query. In contrast, idf reduces the tf scores of words that frequently occur throughout the code base.

Local Technique. We use the call graph as the structural model, as it has been a component of many local techniques [1, 2, 8, 10, 12]. To study the effect of threshold on our local technique, we use a lexically-based tf-idf score to rank the call edges in terms of relevance, in addition to simply including all call edges. We used Eclipse's call hierarchy to generate call graph information.

Subject Concerns and Queries. We use 8 of 9 concerns and queries from a previous concern location study of 18 developers [13]. One of the techniques in the study, GES [7], uses keyword queries suitable for input to tf-idf. For one concern no subject was able to formulate a GES query that returned any relevant results, leaving us with 8 concerns. For each concern, 6 developers interacted with GES to formulate a query, resulting in a total of 48 queries.

Although the queries were formulated using GES, we used tf-idf as the global technique in our study for two reasons: (1) the behavior of GES is significantly affected by the number of query words; and (2) GES uses a proprietary search algorithm that is difficult to reason about when analyzing the results. In contrast, tf-idf is well-understood, thus facilitating research analysis, and has been used in prior work [1].

Methodology and Measures. Given a query, we score each program element using tf-idf. We select the top x for the global results. The local technique augments these x results using call edges, scores the results with tf-idf, and the top y become the local results. We then measure effectiveness in terms of precision (P), recall (R), and their combined harmonic mean, F measure (F) [5].

3. PRELIMINARY QUESTIONS

Since the input to a local technique is the result of a global technique, we first need to understand the performance and variability of our global technique, tf-idf, before we address our primary research questions. There are two inputs to a global search technique: query and threshold.

Queries. For each concern, we have 6 queries [13]. To evaluate combining local and global techniques, should we select the most successful queries in terms of one of our measures (P, R, F), or use all of the queries? The difference tells us how effective the combinations are under ideal conditions as compared to the average case. In the average case, our results depend in part on the developer's ability to construct a query using the search tool.

Thresholds. A global technique such as tf-idf will rank the relevance of each element to the query. But to calculate our measures and provide a set of elements as input for the

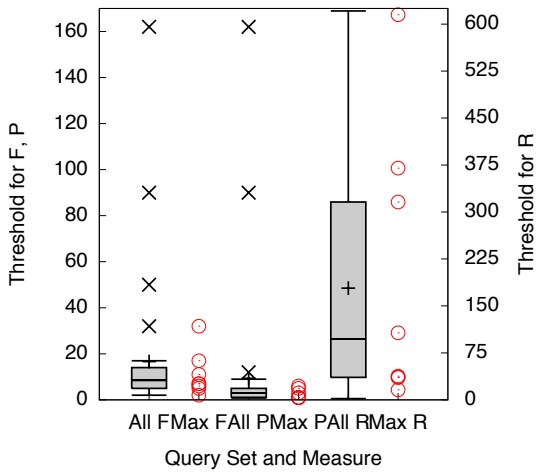

Figure 1: Threshold for the best F, P, and R for every query (all) and the best queries (max).

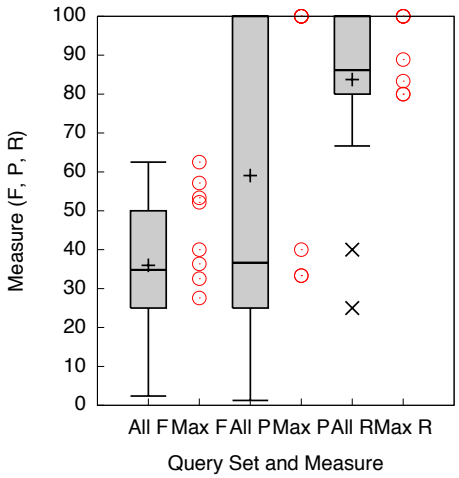

Figure 2: Best F, P, and R measures for every query (all) and the best queries (max).

local technique, we need to select a threshold to determine when an element is considered relevant.

To investigate these two issues, we compared the success of each query in terms of the threshold of its highest P, R, or F. We analyzed the difference between the average case and the best case by comparing all the query results on 3 subsets for each concern: the queries with the best overall P, R, or F. Figure 1 shows box plots for the thresholds at which each query had the highest F, P and R. The shaded box represents the middle 50% of the data, the middle horizontal line is the median, + is the mean, and × indicates an outlier. The thresholds for the best queries in terms of F, P, and R are plotted to the right of each box plot. Figure 1 shows that the best P occurs below a threshold of 20. In contrast, the best R occurs at thresholds of 36 or more for 75% of the queries, with a median threshold of 97. This points to the practical limits of achieving high recall with a global search alone. Figure 2 shows the value of each measure at the best F, P, and R thresholds plotted in Figure 1.

Figures 1 and 2 show that there is no significant difference in F, P, or R between all queries and the set of the best, or max, queries. We used a Bonferroni mean separation test to confirm that the results for all queries versus the best (max) queries for each measure was not significantly different in terms of the threshold or the measure's value. The fact that the best queries in terms of F, P and R are not significantly different from the average case means we should be able to generalize our results from a subset (i.e., the best queries) to the average case.

To summarize, we conclude from our preliminary results that: (1) there is no significant difference between results for the ideal queries and the average case (all queries); and (2) achieving consistently high recall using a global search alone requires the developer to look through as many as 300 results, providing further motivation that global search needs help in improving recall, such as by using local information.

4. PRIMARY QUESTIONS

Our primary research questions are (1) what are the trade-offs in terms of precision (P) and recall (R) at different global and local thresholds, and (2) to what extent are lexical and structural information necessary to achieve high R and P?

Threshold trade-offs. To analyze these trade-offs, we compared 9 global thresholds (1, 3, 5, 7, 10, 15, 20, 25, all) with 12 local (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, all tf-idf > 0, all), traversing 0, 1 and 2 call graph edges away. Because our goal is to balance P and R, we focus our analysis on the set of queries with the best F measure.

As expected, we observe that as the global and local thresholds increase, R increases and P decreases. Interestingly, P does not decrease as quickly as R increases. We confirmed this observation using multiple linear regression. For instance, increasing the local threshold by 1 approximately translates to +2.3 R, −1.1 P, and 4.4 more results returned.

When exploring 1–2 edges away, 5 and 7 appear to be the best global thresholds in terms of F, along with local thresholds of 2–4. Based on this information, we selected a subset of combinations to analyze further, shown in Figures 3 and 4. As can be seen in Figure 3, exploring one call edge away increases R by almost 16 on average, whereas exploring two edges away increases R by just 3. In addition, Figure 4 shows that exploring a second edge away increases the number of results, especially for higher global thresholds.

Finally, we address the open question in [8] as to whether global and local combined outperforms global alone when given appropriate thresholds. We compared our most successful combination in terms of F (a global threshold of 7, local threshold of 2, and exploring 1 call edge) with the global results at threshold 7 with no additional edges. A paired-difference t-test using Wilcoxon signed rank confirms that combining local and global techniques results in significantly higher R, without a significant difference in terms of P ($\alpha = 0.05$). Adding local edges resulted in 1.875 more true positives on average, with just 6 more returned results.

The role of lexical and structural information. As we saw in Figure 2, not all queries achieve 100% recall, regardless of threshold. However, we do achieve 100% recall on our data set within two structural call edges. In fact, we reach 100% recall on 6/8 concerns with just a single edge. The fact that the remaining relevant elements are just 1-2 edges away from the global results means that there is potential to achieve excellent recall and keep precision in balance with an effectively-designed local technique.

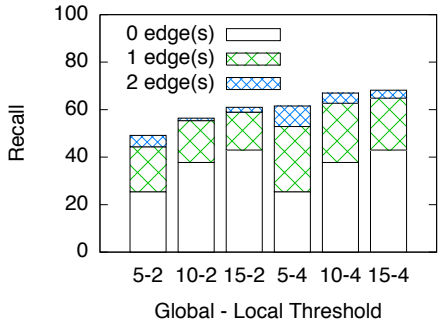

Figure 3: Recall at various global and local thresholds for 0, 1, and 2 structural edges away.

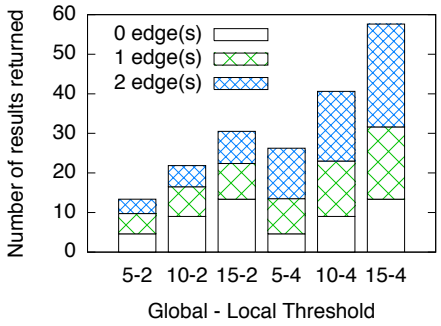

Figure 4: Number of results returned at various thresholds for 0, 1, and 2 structural edges away.

To further investigate how such global and local techniques could be constructed, we analyzed the results missed by the best approach in terms of F measure: a global threshold of 7 with a local threshold of 2, exploring just 1 edge away. For each result missed, we counted (with duplicates) what possible solutions would have resulted in the element being included in the result set. Of the 29 relevant results missed, we observed 5 broad categories of potential solutions: improving the global technique's accuracy (21), using a higher local threshold (15), exploring two or more edges away (14), utilizing structural information to overcome 0 tf-idf scores (9), and augmenting the query to increase recall after global search but before applying the local technique (7). Both a higher local threshold and exploring more edges away require improved precision of the local technique. Researchers can prioritize their efforts based on which solution categories have the biggest potential benefits.

These results also suggest potential solutions for future concern location technique designers. For instance, structural information could be used to overcome poor tf-idf scores if all callers of an element are already in the concern. Alternately, the global search could include information beyond the element's declaration, such as class or package names. Given these results, it appears that using edges from a simple structural model such as the call graph can be effective in general, but more work is needed in creating higher precision global and local ranking techniques.

5. CONCLUSION

We presented an empirical investigation of how global and local techniques can be effectively combined for concern location. We observed that as global and local thresholds increase, recall increases more than precision decreases. However, the raw number of returned results is still a practical consideration for users. A regression model could approximate how many additional results are expected so users can select a local threshold based on global technique success.

We have confirmed the prior foundations of techniques combining local and global information [1, 8] by observing that 100% recall can be achieved within 1–2 structural edges from the global search results. While this result indicates the potential success of combining global and local techniques, further work is needed in designing global and local techniques that keep precision at acceptable levels.

We base our conclusions on the results of 48 queries for 8 concerns in Java. Other developers, concerns, and languages may yield different conclusions. Given the statistical significance of the results, we expect the trends to be similar.

6. REFERENCES

[1] M. Eaddy, A. V. Aho, G. Antoniol, and Y.-G. Gueheneuc. Cerberus: Tracing requirements to source code using information retrieval, dynamic analysis, and program analysis. In *IEEE Int'l Conf. Prog. Comprehension*, 2008.

[2] E. Hill, L. Pollock, and K. Vijay-Shanker. Exploring the neighborhood with Dora to expedite software maintenance. In *Int'l Conf. Auto. Soft. Eng.*, 2007.

[3] A. J. Ko, B. A. Myers, M. J. Coblenz, and H. H. Aung. An exploratory study how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Trans. Soft. Eng.*, 32(12):971–987, 2006.

[4] S. Letovsky and E. Soloway. Delocalized plans and program comprehension. *IEEE Soft.*, 3(3):41–49, 1986.

[5] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge Univ. Press, New York, NY, USA, 2008.

[6] A. Marcus, A. Sergeyev, V. Rajlich, and J. I. Maletic. An information retrieval approach to concept location in source code. In *Work. Conf. Reverse Eng.*, 2004.

[7] D. Poshyvanyk, M. Petrenko, A. Marcus, X. Xie, and D. Liu. Source code exploration with Google. In *IEEE Int'l Conf. Soft. Maintenance*, 2006.

[8] S. Ratanotayanon, H. J. Choi, and S. E. Sim. My repository runneth over: An empirical study diversifying data sources to improve feature search. In *IEEE Int'l Conf. Prog. Comprehension*, 2010.

[9] M. Revelle, B. Dit, and D. Poshyvanyk. Using data fusion and web mining to support feature location in software. In *Int'l Conf. Prog. Comprehension*, 2010.

[10] M. P. Robillard. Automatic generation suggestions for program investigation. In *ESEC/FSE*, 2005.

[11] M. P. Robillard and G. C. Murphy. Representing concerns in source code. *Trans. Soft. Eng & Meth.*, 16(1):3, 2007.

[12] Z. M. Saul, V. Filkov, P. Devanbu, and C. Bird. Recommending random walks. In *ESEC/FSE*, 2007.

[13] D. Shepherd, Z. P. Fry, E. Hill, L. Pollock, and K. Vijay-Shanker. Using natural language program analysis to locate and understand action-oriented concerns. In *Conf. Aspect-Oriented Soft. Devel.*, 2007.

[14] P. Tarr, H. Ossher, W. Harrison, and J. Stanley M. Sutton. N degrees separation: multi-dimensional separation concerns. In *Int'l Conf. Soft. Eng.*, 1999.