Example-Based Problem Solving Support Using Concept Analysis of Programming Content

Roya Hosseini and Peter Brusilovsky

University of Pittsburgh - Pittsburgh, PA
{roh38,peterb}@pitt.edu

Abstract. This paper presents two different approaches to example-based problem solving support in the domain of programming based on concept analysis of the learning content. The goal of these approaches is to offer students a set of most relevant remedial examples when they have trouble solving a problem. The paper reviews earlier work and introduces a global and a local approach for selecting examples that are similar to the problem in terms of concept coverage and structure of the content, respectively. It also reports results of a lab study conducted to explore the effectiveness of each approach.

Keywords: concept-based similarity, problem solving support, remediation.

1 Introduction

Example-based problem solving is one of the efficient approaches used by Intelligent Tutoring Systems (ITSs) in the programming domain [1]. In this approach, when the student has trouble solving a problem, the system tries to find the relevant examples which might be helpful to solve the problem. The approach has been used, for example, in ELM-ART ITS for LISP programing [2]. While known to be efficient, this approach remains one of the least explored since the original LISP research was based on advanced episodic learner modeling which is difficult to build for other programming domains.

Our goal is to create a different version of the example-based problem solving support for Java programming which is generalizable in multiple different programming languages without too many effort that is required for advance analysis of content in a system like ELM-ART. The main innovation is in analyzing domain concepts related to programming problems and examples and using the underlying concept structure to find similarity between examples and problems. The similarity can be obtained using number of complicated approaches, but our first challenge was to choose between global and local similarity approaches. A global concept-based similarity considers whether two sets of concepts are more or less similar as a whole and in its simple form could be identified by cosine similarity of vectors of concepts. A local concept-based similarity considers similarity on the structure level where detailed level of similarity can be identified by structure of blocks and adjacent concepts.

To explore and compare these two approaches, we developed a specialized concept analysis tool, JavaParser, which can extract not only the list of concepts but also concept structure [3]. The parser provides a fine-grained level of indexing per line of code which helps identifying blocks of code that have sets of adjacent concepts.

S. Trausan-Matu et al. (Eds.): ITS 2014, LNCS 8474, pp. 683–685, 2014.

The parser helped us index a considerable volume of Java programming problems and examples and we could then start a study comparing the approaches. In the present work, we introduce global and local concept-based approaches for finding similar examples for Java problems and present preliminary results of a lab study that aims to compare the effectiveness of these approaches.

2 Method

We propose global and local concept-based approaches for example-based problem solving. The conventional global concept-based similarity approach is based on cosine similarity with TF-IDF weighting for vectors of concepts. The local concept-based similarity approach is based on selecting sets of examples that have the closest blocks of code to the question that the user failed in. The main idea of this approach is to build subtrees of concepts that have appeared together as blocks in each of the contents. As a result, each subset of concepts that are either in the same line or in the same block, will be merged together to create a subtree for the content. Having created the subtrees, we can find the similarity of a question and example by comparing their corresponding subtrees. Several methods have been suggested to compare trees, among which Tree Edit Distance (TED) is quite well known and has been widely used in other studies for similar purposes [4]. The distance between the question and the example is obtained using Eq. (1):

$$WD_{Q,E} = \sum_{q=1}^{N} Arg\min_{e \in M}(TED(s_q, s_e) \times \frac{1}{w_{Qe}}) + \sum_{e=1}^{M} Arg\min_{q \in N}(TED(s_e, s_q) \times \frac{1}{w_{Eq}}) \quad (1)$$

where WD_{QE} is the weighted distance between question Q and example E; N and M represent the total number of the subtrees in Q and E, respectively; TED is the Tree Edit Distance between the given subtrees s_q and s_e; W_{Qe} is the sum of Term Frequency–Inverse Document Frequency (TF-IDF) values of the concepts in the subtree s_e for the question Q; Similarly, W_{Eq} is the sum of TF-IDF values of the concepts in the subtree s_q for the example E. Finally, the weighted distance WD_{QE} is used in Eq. (2) to determine the similarity between example E and question Q:

$$Sim = \frac{1}{e^{\alpha WD_{Q,E}}} \quad (2)$$

where Sim is the local concept-based similarity between the example E and question Q; and α is a coefficient for the exponential function which is set to 0.01 in the present study. Finally, the top five examples with the highest similarity value can be selected for presenting to the student as remedial support.

3 Preliminary Evaluation

We conducted a lab study to investigate the effectiveness of local and global concept-based similarity approaches. We hypothesized that the local approach might outperform

the global one since it considers the closeness of chunks in determining the similarity in a somewhat similar way to ELM-PE that was known as a very successful example recommendation approach. The study started in January 2014 and is ongoing. To date, we have had 12 subjects. The learning materials are organized into 6 Java topics, 83 annotated Java examples, and 24 parametric Java questions. Subjects were assigned to three topics based on their pretest scores and had to solve 4 questions in each of those topics. We selected relevant examples for the questions and asked the student to rate their helpfulness in two different conditions: one at the time that student's answer to question was incorrect and one after the student finished solving a question. The rating ranges from 0 representing 'Not helpful at all' to 3 representing 'Very helpful'. An analysis of the results showed that the average rating of students for the examples selected by the global concept-based approach (1.95) was higher than the local concept-based approach (1.49). Contrary to what we expected, closeness of the concepts did not help select more relevant examples. However, for a much more detailed discussion of results, we need to look into different conditions under which each approach performs better.

4 Conclusion

We proposed global and local concept-based approaches that provide remedial recommendations of examples for Java programming problem solving. These approaches can be leveraged to address the problem solving support in different programming languages with little effort. The result of the ongoing lab study showed that global similarity is almost always helpful for students while local similarity is not. The collected rating data can be useful for cross validation of multiple numbers of similarity approaches and this data is open for other researchers as well. For future work, we would like to perform detailed analysis on the results and also explore whether the approaches can be improved by taking into account the user knowledge.

References

1. Brusilovsky, P., Peylo, C.: Adaptive and intelligent Web-based educational systems. International Journal of Artificial Intelligence in Education 13(2-4), 159–172 (2003)
2. Weber, G.: Individual selection of examples in an intelligent learning environment. Journal of Artificial Intelligence in Education 7(1), 3–31 (1996)
3. Hosseini, R., Brusilovsky, P.: JavaParser: A Fine-Grain Concept Indexing Tool for Java Problems. In: The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013), pp. 60–63 (2013)
4. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing 18(6), 1245–1262 (1989)