A Study of Concept-Based Similarity Approaches for Recommending Program Examples

Roya Hosseini[a] and Peter Brusilovsky[b]

[a]Intelligent Systems Program, University of Pittsburgh, Pittsburgh, USA; [b]School of Information Sciences, University of Pittsburgh, Pittsburgh, USA

ABSTRACT

This paper investigates a range of concept-based example recommendation approaches that we developed to provide example-based problem-solving support in the domain of programming. The goal of these approaches is to offer students a set of most relevant remedial examples when they have trouble solving a code comprehension problem where students examine a program code to determine its output or the final value of a variable. In this paper, we use the ideas of semantic-level similarity-based linking developed in the area of intelligent hypertext to generate examples for the given problem. To determine the best-performing approach, we explored two groups of similarity approaches for selecting examples: non-structural approaches focusing on examples that are similar to the problem in terms of concept coverage and structural approaches focusing on examples that are similar to the problem by the structure of the content. We also explored the value of personalized example recommendation based on student's knowledge levels and learning goal of the exercise. The paper presents concept-based similarity approaches that we developed, explains the data collection studies and reports the result of comparative analysis. The results of our analysis showed better ranking performance of the personalized structural variant of cosine similarity approach.

KEYWORDS

Problem-solving support; program examples; concept-based similarity; code structure

1. Introduction

Example-based problem-solving is one of the efficient approaches used by intelligent tutoring systems (ITSs) in the programming domain (Brusilovsky and Peylo, 2003). With this approach, when the student has trouble solving a problem, the system tries to find relevant examples that might be helpful in solving the problem. A domain expert typically selects relevant examples; however, this selection is a time-consuming task that requires considerable amount of human effort and makes the process costly when the volume of learning content grows.

There are other possible approaches to linking problems and examples dynamically and without experts' help. Some of these approaches are based on analysis of the content, such as keyword-based approaches that measures the similarity of content on the level of keywords (Kibby and Mayes, 1989; Mayes et al., 1988). Although content-based approaches are simple, they have a great deal of surface-level similarity, which is

CONTACT Roya Hosseini. Email: roh38@pitt.edu

not good enough for identifying content that is truly similar. An alternative approach is to focus on knowledge behind the content (Koedinger et al., 2012). Knowledge-based approaches could be used to generate higher quality links between content items, since they use semantic similarity (Carr et al., 2001). An example of elaborated knowledge-based approach for connecting problems and examples can be found in (Weber and Mollenberg, 1994).

The work presented in this paper uses ontology *concepts* from the Java ontology developed by PAWS Lab[1] as elements of Java domain knowledge to build *concept-based similarity approaches* for learning content in the domain of Java programming. The goal of these approaches is to find the most relevant examples for Java programming problems. Our main innovation lies in analyzing domain concepts related to examples and problems, as well as using the underlying concepts to find similarity between them. We explored several novel concept-based similarity approaches: *structural* vs. *non-structural* approaches to investigate the impact of structure, and *personalized* vs. *non-personalized* approaches to investigate the impact of user information available from the user model. A non-structural concept-based similarity approach can be considered a bag-of-concept approach that determines whether two sets of concepts are more or less similar as a whole. On the other hand, a structural concept-based similarity approach determines whether the subsets of adjacent concepts that have appeared together in the same line or block are similar. Both approaches determine similarity solely based on a conceptual analysis of examples and problems, and are thus non-personalized, which means that they are independent from individual students. In personalized approaches, we studied the effect of personalization on structural and non-structural similarity approaches by accounting for information about each student's knowledge levels and/or learning goal.

We evaluated our approaches using data collected from studies where users and experts were asked to rate the helpfulness of examples selected for different problems. The user study was designed to collect ratings for answering the following questions: (Q1) which of two competing approaches - structural and non-structural - can select more relevant examples for problems? (Q2) Will personalization that is based on a student's knowledge levels and/or learning goal result in more relevant examples generated by concept-based similarity approaches? We selected a subset of approaches to use in the study with a goal of collecting a sufficient amount of rating data on relevant example and problem pairs. The expert study was carried out after the user study to collect judgments of domain experts for examples thats users have rated for each problem.

The rest of the paper starts with an overview of the related work, from how examples are used for problem-solving in the ITS domain to the most common approaches in information retrieval that could be used for finding similar examples and problems. We then introduce the context of our work by explaining the platform that we use for problem-solving support, namely jHelp. This platform was used in our online classroom study, as well as the study that was conducted to collect rating data. Next, we describe approaches that we used for finding similar examples for Java problems, followed by the studies that we conducted to evaluate the helpfulness of each approach in recommending examples for problems. After that, we present the results of our comparisons that were obtained using the rating data collected from the studies. Finally, we close with a discussion of the results and plans for future work.

[1] `http://www.sis.pitt.edu/~paws/ont/java.owl`

2. Related work

2.1. *Example-based problem-solving in intelligent tutoring systems*

One of the main challenges of intelligent tutoring systems (ITS) is to provide problem-solving support when a student fails to correctly solve a problem (for example, a programming problem). One way to provide this support is by using expert-crafted worked-out examples that are isomorphic to the problem. Each worked-out example consists of a problem formulation, solution steps, and a final solution. Studies have shown that such examples are beneficial for the initial acquisition of cognitive skills (Renkl et al., 2002), especially for novice learners (Kirschner et al., 2006). This finding encouraged a range of projects that explored strategies for providing problem-solving support by combining worked-out examples and problems (Gross et al., 2014; McLaren and Isotani, 2011; McLaren et al., 2008, 2014; Najar et al., 2014; van Gog, 2011). However, the number of problems with worked-out examples is usually limited, as creating worked-out examples for each problem requires a lot of authoring effort. Moreover, sometimes the concepts that lead to failure can be better illustrated by a worked-out example that is not isomorphic to the current problem. Whether one can find another approach that would overcome the aforementioned challenges remains an open and interesting question that our current work seeks to answer.

2.2. *Similarity approaches for content linking*

The idea to automatically connect two similar content items was first explored in the area of educational hypertext under the term similarity-based navigation in the pioneering work of Mayes and Kibby (Kibby and Mayes, 1989; Mayes et al., 1988). The idea of their approach was to create links between documents that are similar on the keyword level. Due to its low quality, keyword-level similarity-based navigation has been replaced by several types of more reliable semantic linking, which are frequently referred as intelligent linking in the area of hypertext.

The first type of semantic linking is known as metadata-based linking (Tudhope and Taylor, 1997). The core idea of similarity-based navigation is the same as in keyword-based linking: a similarity measure is computed between documents, and those with similarity above a certain threshold are connected by a link. The metadata similarity is calculated as a weighted measure of similarity along each facet of metadata. Since metadata expresses semantic similarity (in contrast to the surface similarity that is expressed by keywords) this approach typically produces higher quality links.

More recently, the focus of research on semantic linking has moved to ontology-based linking. Ontology-based linking is possible when documents are indexed with ontology terms, either manually or automatically. In this case, the process of finding similar documents is more challenging, since it has to account for the position and connections of ontological terms in the ontology. Hypermedia researchers extensively explored ontology-based linking in the early 2000s (Carr et al., 2001; Crampes and Ranwez, 2000) and the application of this technology for educational domain followed in just a few years (Dolog et al., 2003).

In the programming domain, similarity has been mostly investigated using content-level information. A recent example is the work of Gross et al. (2014), where Java programming contents are linked based on the similarity of their corresponding abstract syntax trees (ASTs). The syntax tree focuses on the whole body of examples and problems. However, this approach may not be the most effective, since better

similarity can be obtained by considering smaller fragments (structures) within the content of examples and problems (Weber, 1991). An early example of this approach was implemented in ELM-PE ITS for LISP programing (Weber and Mollenberg, 1994). While it is 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 expand to other programming domains and more complex problems.

Motivated by the approach introduced in ELM-PE, the present work continues our past attempt to explore concept-based similarity approaches (Hosseini and Brusilovsky, 2014), and was aimed at finding the best ways to calculate the ontology-based similarity of programming items (i.e. problems and examples). Here, we will refer to our similarity approaches as concept-based, since we treat terms of the ontology as programming concepts. Our similarity approaches are 'automatic', meaning that they do not need an expert for designing isomorphic problem-example pairs. Instead, relevant examples for a problem are found by searching the space of learning materials. At the same time, as compared to "episodic" approaches used in ELM-PE (Weber and Mollenberg, 1994), our approaches are flexible enough to be applied to different programming domains with relatively low effort. What makes this possible is that these approaches rely on information about concepts associated with lines of the code that are automatically extracted from the program code. We believe that these similarity approaches could work for a range of code-related programming problems. However, in our study, we focus solely on code comprehension problems (where students are presented with a program code and asked to predict its output or the final value of a variable) rather than on program composition problems (where students have to write a code that performs a specific task).

3. The context of the work

The work on advanced similarity approaches presented in this paper was motivated by our experience with our example-based problem-solving support system jHelp, an online tool that provides students with an interface to access examples while they solve programming problems in a Java course. The problems are parameterized code evaluation exercises, which the subject could repeat several times with different parameters (Hsiao et al., 2010). The problems also include automatic evaluation and feedback for students that says whether the response was correct or incorrect. On the other hand, examples are snippets of code that are enriched with line-by-line annotations that are displayed when subjects click on each annotated line (Brusilovsky et al., 2009). When a student fails to correctly solve a problem, jHelp recommends examples that could be helpful for solving the problem. The example recommendation interface is shown in Figure 1, where the left panel shows two recommended examples to the student who failed to solve a problem. Student can select any of the recommended examples and explore them line by line (see Figure 2).

We used jHelp in a Java programming course for several semesters with a few relatively simple example recommendation approaches. To assess the quality of these approaches, we augmented jHelp with a feedback mechanism by adding buttons to rate the helpfulness of each recommended example for solving the original problem (see bottom of Figure 2). While the students extensively used the recommended examples, they rarely rated them. Since the volume of explicit feedback generated by the use of jHelp in real classes was insufficient to reliably evaluate example recommendation

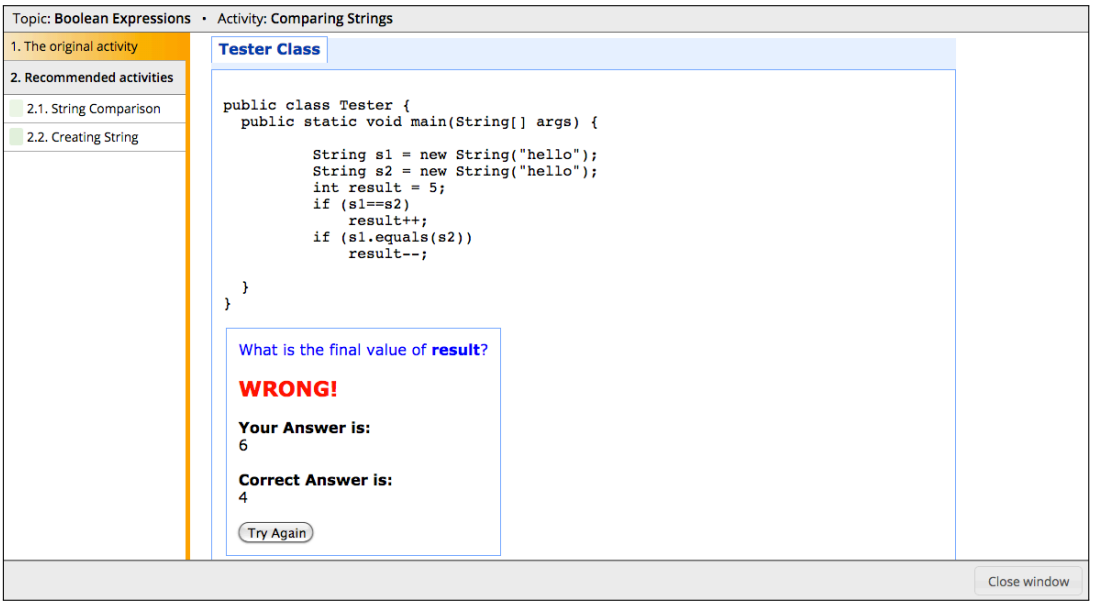


Figure 1. A typical case of problem-solving support in jHelp. When a student fails at the 'Comparing Strings' problem, the system recommends relevant examples for that problem. 'The original activity' and 'Recommended activities' on the left panel provide access to the problem and the list of recommended examples, respectively.

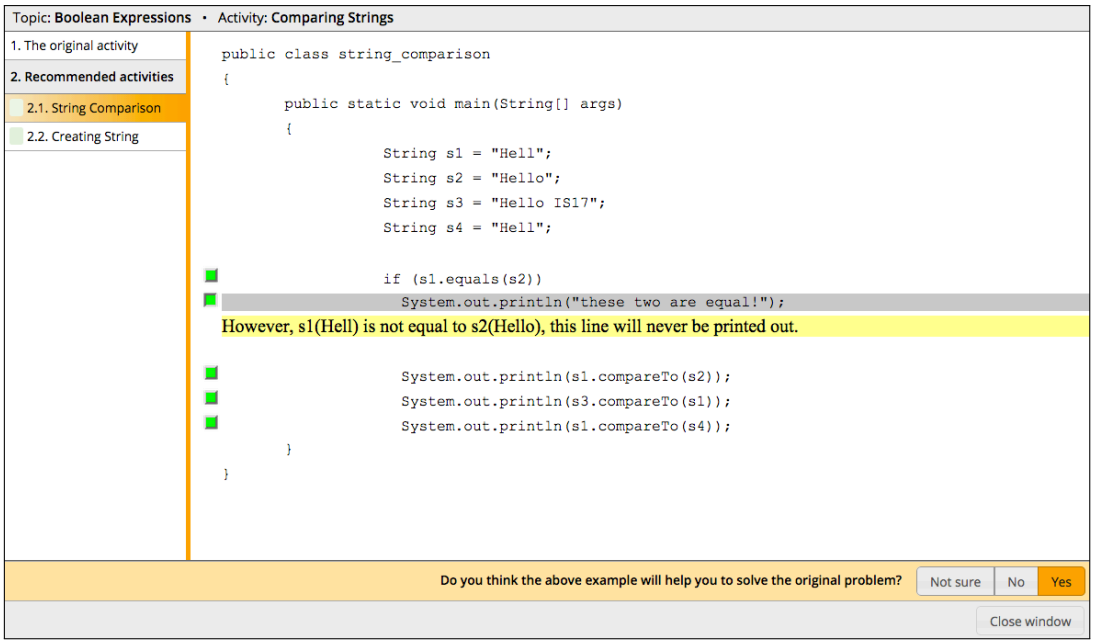


Figure 2. A recommended example selected by the student for the problem shown in Figure 1. A click on the bullet that is located next to some of the code lines shows the annotation for the corresponding line in the code. Here, the student confirmed that this example is helpful for solving the problem by clicking on the 'Yes' button.

approaches, we designed a lab study to collecting subjective ratings on how helpful examples are for the problems.

In the next section, we describe our proposed concept-based approaches in detail. In the first step, we describe the process required for extracting concepts from programming items. After that, we introduce how information about concept structures can be used to calculate similarity between programming problems and examples.

4. Similarity approaches for linking problems and examples

Our idea of similarity-based linking of Java programming learning contents is based on the ability of automatic extraction of ontology concepts from content items. Automatic concept extraction is an active research area, with the majority of work focused on concept extraction from text and Web pages (Carmel et al., 2012; Parameswaran et al., 2010). This approach is most frequently used to connect documents to ontologies and similar frameworks, like Wikipedia (Csomai and Mihalcea, 2008; Milne and Witten, 2008), but it could be equally useful for ontology-based linking between documents (Carr et al., 2001; Wang and Taylor, 2007). However, the domain of programming offers a more attractive source for concept extraction – programming code fragments exist in the majority of learning content in this area, from textbook chapters to examples to problems. The formal nature of programming code makes it easy to not only extract a list of programming concepts used in the code, but also recognize the structure of concepts used in the code; namely, which set of concepts are used together in each sub-fragment of the code. We used these structures with the hope of finding items with higher similarity.

4.1. *Concept extraction from programming items*

Our first step for identifying concept structures in programming items was to use our concept extraction tool, called JavaParser (Hosseini and Brusilovsky, 2013). This parser extracts a list of ontological concepts from the Java source code, using the PAWS Java ontology mentioned earlier. It provides us with not only the concepts, but also with detailed information about the location inside the code where the concept starts or ends. As a result, the conceptual structure can be identified by lines or blocks of code that have sets of adjacent concepts. Table 1 lists a set of concepts indexed by the parser for the *WhileDemo* content shown in Figure 3. The concept "While Statement" forms a block, and thus, has a different start and end line.

The parser helped us index a considerable volume of Java problems and examples. After this step, we had a good knowledge of the concept structure within programming items, and were ready to explore different ways of using this information in our main task, which is finding good examples for programming problems.

4.2. *Structural vs. non-structural approaches*

The extracted content-concept mappings enriched with line information can be used for similarity approaches in two different ways. One way is to consider the content as one big chunk and to represent it with a vector of concepts that have appeared inside it. Since this approach provides us with a big picture of the concepts, it will hereafter be referred to as the 'bag-of-concepts' approach. Another way is to make

```
1 public class WhileDemo {
2
3     public void demo() {
4         int count = 1;
5         while (count < 11) {
6             count++;
7         }
8         System.out.println("Count=" + count);
9     }
10 }
```

Figure 3. The WhileDemo content

Table 1. Concepts extracted by JavaParser for WhileDemo content

Line	Concept
1	Class Definition, Public Class Specifier
3	Method Definition, Public Method Specifier, Void Data Type
4	Integer Data Type, Simple Assignment Expression
5	Less Expression
5-7	While Statement
6	Post-Increment Expression
8	Java.lang.system.out.println, Actual Method Parameter, String Literal, String Addition

use of the structure of concepts in the content item to represent it with multiple structure subtrees that each contain distinct concepts that have appeared together in the same line or block. Since this approach captures existing concept structures in the programming item, we will hereafter refer to it as the 'concept-structure' approach. Figure 4 shows the concept-structure representation for the *WhileDemo* content, listing all of its subtrees. Note that the While block creates three subtrees: one for the whole block (including the line that includes the while statement), the content of the block, and the line of the while statement.

In the following sections, we describe different structural and non-structural approaches and illustrate each of them with an example. In all of the cases, we consider the content space shown in Table 2 with one problem ($p1$) and two examples ($e1$, and $e2$). Problem $p1$ has two subtrees, and each example has only one subtree. Concepts inside a subtree are separated by comma. We will show how each approach can be used to find the similarity between problem $p1$ and example $e1$.

4.2.1. Non-structural similarity approaches

A non-structural concept-based similarity can be considered as a bag-of-concept approach that in its simple form could be identified by cosine similarity. Assuming that P and E are the vector of concepts for the problem and example, cosine similarity is defined as follows:

Table 2. Contents used as illustrative examples for non-structural and structural similarities

Content	Code	Concepts	Subtrees
Problem p1	x++; x--;	post increment; post decrement	(1) post increment (2) post decrement
Example e1	print (x++);	print; post increment	(1) print, post increment
Example e2	return null;	return; null	(1) return, null

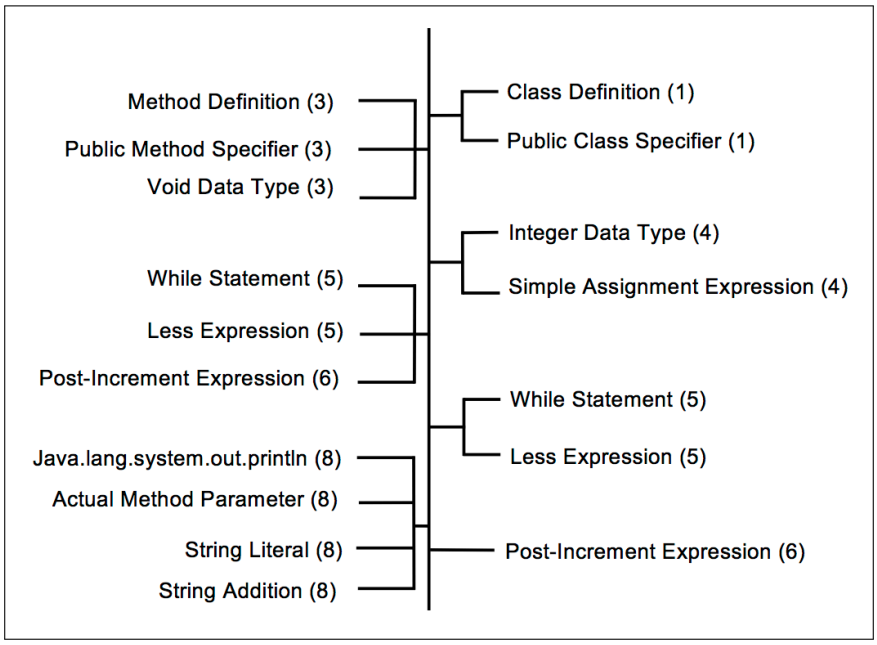


Figure 4. Concept-structure representation for *WhileDemo* Content. A number in parentheses indicates the concept's line number in the program, as shown in Figure 3

$$S_{pe} = \frac{P \cdot E}{\|P\|\|E\|} = \frac{\sum_{i=1}^{N} P_i \times E_i}{\sqrt{\sum_{i=1}^{N} {P_i}^2} \times \sqrt{\sum_{i=1}^{N} {E_i}^2}} \quad (1)$$

where S_{pe} is the similarity between the problem p and example e and ranges from 0 to 1, N is the length of the vectors, and where the concept vectors are weighted using the tf-idf values of the concepts in the problem and example.

For example, the cosine similarity between problem $p1$ and example $e1$ in Table 2 can be obtained as follows: In this case, the vector of concepts will be: $< post\ increment,\ post\ decrement,\ print >$. By calculating tf-idf weights in the given content space (3 contents) and weighting the vectors, we get the weighted vector $< 0.2, 0.5, 0 >$ for problem $p1$ and $< 0.2, 0, 0.5 >$ for example $e1$. The cosine similarity between $p1$ and $e1$ will be as follows:

$$S_{pe} = \frac{(0.2 \times 0.2) + (0.5 \times 0) + (0 \times 0.5)}{\sqrt{0.2^2 + 0.5^2 + 0^2} \times \sqrt{0.2^2 + 0^2 + 0.5^2}} = 0.14$$

An alternative approach is to determine the similarity based on the number of matching and mismatching concepts between two programming items. For this purpose, we use association coefficient, introduced in (Zhong et al., 1997). Assuming that a is the number of concepts that are common to the problem p and example e (number of matching concepts), and that b is the number of the concepts that are not common

to them (number of mismatching concepts), the association coefficient between the problem p and example e is defined as follows (ranging from -1 to $+1$):

$$S_{pe} = \frac{2a - b}{2a + b} \tag{2}$$

So, for problem $p1$ and example $e1$ (see Table 2), there is one matching ($a = 1$) and two mismatching concepts ($b = 2$). Thus, the association coefficient gives us the similarity value of 0 as shown below:

$$S_{pe} = \frac{2 \times 1 - 2}{2 \times 1 + 2} = 0$$

4.2.2. Structural similarity approaches

There are different ways to use information about concept structures. One approach is to represent structures as trees and then to compare them using tree edit distance (TED) (Zhang and Shasha, 1989). The goal of this approach is to minimize the distance between the subtrees in the two programming items; in this case, the problem and example. The overall distance between the problem and example is obtained as follows:

$$D_{pe} = \sum_{i=1}^{N} \min_{j \in M} \{ \frac{TED(i,j)}{w_{p,j}} \} + \sum_{j=1}^{M} \min_{i \in N} \{ \frac{TED(j,i)}{w_{e,i}} \} \tag{3}$$

where D_{pe} is the weighted distance between problem p and example e (ranging from 0 to 1); N and M represent the total number of the subtrees in p and e, respectively; TED is the tree edit distance between the given subtree i in the problem and j in the example; $w_{p,j}$ is the sum of tf-idf values of the concepts in the example subtree j for the problem p; and similarly, $w_{e,i}$ is the sum of tf-idf values of the concepts in the problem subtree i for the example e. The weighted distance D_{pe} is used to calculate the similarity between the example and the problem as follows:

$$S_{pe} = \exp(-\alpha D_{pe}) \tag{4}$$

where S_{pe} is the similarity between the example e and problem p; and α is a coefficient for the natural exponential function which we set on a trial-and-error basis to 0.01 as it generates more meaningful similarities.

For example, in this case, the similarity between problem $p1$ and example $e1$ can be obtained as follows: As shown in Table 2, $p1$ has two subtrees (1: post increment; 2: post decrement) and $e1$ has one subtree (1: print, post increment). Thus, we have $N = 2$ and $M = 1$, and Equation (3) can be rewritten as follows:

$$\begin{aligned} D_{pe} = & \frac{TED(i=1,j=1) + TED(i=2,j=1)}{w_{p,1}} + \\ & \min\{\frac{TED(j=1,i=1)}{w_{e,1}}, \frac{TED(j=1,i=2)}{w_{e,2}}\} \end{aligned} \tag{5}$$

The distance between the first subtree of $p1$ and the subtree in $e1$, $TED(i = 1, j = 1)$ or $TED(j = 1, i = 1)$, is 1, and the distance between the second subtree of $p1$ and

subtree of $e1$, $TED(i = 2, j = 1)$, or $TED(j = 1, i = 2)$, is 3. Also, the sum of the tf-idf values of concepts in subtree of $e1$ for problem $p1$, $w_{p,1}$ is 0.2. Similarly, the sum of tf-idf values of the concepts in subtree 1 and 2 of the $p1$ for the example is $w_{e,1} = 0.2$ and $w_{e,2} = 0$, respectively. By plugging these values into the equation, we obtain $D_{pe} = 25$ and $S_{pe} = e^{-0.25} = 0.78$.

An alternate structural similarity approach lies in applying a tree-comparison method based on subtree similarity (Zhong et al., 1997) to our task (i.e., finding relevant examples for a problem). This approach is originally known as the webbing matrix method (WMM). Assuming that a problem has N and an example has M subtrees, a matrix of length $N \times M$ can be created with N and M subtrees as the row and column headings, respectively. The value of each element, S_{ij}, represents the similarity between subtree p_i and e_j. The similarity of 1 means that the problem and example subtrees match exactly. When 1 appears in any element, asterisks will be placed in all other elements of its column and row in the matrix (see Figure 5). This means that if two subtrees, p_i and e_j, are exactly matched, other subtrees do not need to be compared with either one of the two matching subtrees. An overall similarity measure is defined by summing up all non-asterisked elements inside this matrix, as follows:

	e_1	e_2	…	e_j	…	e_M
p_1	S_{11}	S_{12}	…	*	…	S_{1N}
p_2	S_{21}	S_{22}	…	*	…	S_{2N}
…	…	…	…	*	…	…
p_i	*	*	*	1	*	*
…	…	…	…	*	…	…
p_N	S_{M1}	S_{M2}	…	*	…	S_{MN}

Figure 5. An illustration of the webbing matrix method

$$S(p, e) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} \alpha(S_{ij}) S_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{M} \alpha(S_{ij})} \tag{6}$$

$$\alpha(S_{ij}) = \begin{cases} 1 & \text{if } S_{iy} \neq 1 \text{ and } S_{xj} \neq 1 \\ 0 & \text{if } S_{iy} = 1 \text{ and } j \neq y \text{ or } S_{xj} = 1 \text{ and } i \neq x \end{cases} \tag{7}$$

$$x = 1, 2, \ldots, N \; y = 1, 2, \ldots, M$$

where $S(p, e)$ is the similarity between problem p and example e, and $\alpha(S_{ij})$ is a weight function related to S_{ij}. Asterisks are assigned a weight of 0 and the rest of elements have a weight of 1. Here, S_{ij} is based on the similarity of concepts in the two subtrees p_i and e_j, and is obtained by using cosine similarity (Equation 1) or the association coefficient (Equation 2).

To illustrate this further, consider the problem $p1$ and example $e1$ (see Table 2). The WMM has one column that represents the subtree in $e1$, and two rows that represent the two subtrees in $p1$. Assuming that association coefficient is used to find the similarity between two subtrees, the similarity between the subtree 1 in $p1$ and subtree of $e1$, (S_{11}), will be 0.33, and the similarity between subtree 2 in $p1$ and subtree of $e1$, (S_{21}), will be -1. By applying Equation 6 we get a similarity value of $(0.33 + (-1))/(1 + 1) = -0.34$.

4.3. *Personalized approaches*

Thus far, we have discussed similarity approaches that provide the same relevant examples for problems for all students. However, this approach might not produce good examples, since students have different levels of domain knowledge and different learning goals. One of the questions that we would like to answer at this time is whether personalizing the example selection based on student's knowledge levels can recommend more relevant examples than the corresponding non-personalized approaches.

Our first hypothesis is that it might be good to *re-rank* the list of examples selected by a non-personalized approach, and to prioritize examples that will be easier for the target students to understand because the concepts that they illustrate are better known. An example to consider would be one in which the similarity approach returns a ranked list of $< e1, e2, e3 >$ that is ordered by a decreasing degree of similarity of the example to the problem. If we re-order these examples according to the student's level of knowledge in the example's concept, we may get a different order $< e2, e1, e3 >$, meaning that the student has more knowledge in concepts of $e2$ and less knowledge in concepts of $e3$. We assume that this would reduce the overall cognitive load and thus lead to greater learning from examples.

An alternative to the *re-ranking* approach, which we refer to as the *average* approach, is to combine personalized ranking with a non-personalized ranking that averages the similarity value returned by each of the two approaches. For example, when similarity values between examples and a problem is $< e1 = 0.9, e2 = 0.8, e3 = 0.6 >$ and student's knowledge levels in the examples is $< e1 = 0.5, e2 = 0.2, e3 = 0.6 >$, the final rank of the examples using *average* approach will be $< e1 = 0.7, e2 = 0.5, e3 = 0.6 >$.

Our second hypothesis is that examples are helpful when they target the particular concepts within a problem that the student does not know. We wanted our approaches to prioritize examples for a given problem, based on the level that each example could contribute to learning the less-known concepts within the problem. We tried to integrate this hypothesis into our previously introduced non-structural and structural similarity approaches. Remember that we used either cosine similarity or the association coefficient for comparing bag-of-concepts (in the non-structural approach), or subtrees of concepts (in the structural approach). Here, we need to modify these similarity metrics to consider not only information about concepts, but also each student's knowledge levels.

In particular, in cosine similarity scenarios (Equation 1), instead of using tf-idf values for weighting the concepts, we define the weight of each concept to be the amount that student can learn, such that lower knowledge level for a concept makes the weight of that concept larger. In this case, the weight for the concept i is $1 - k_i$ where k_i is the student's knowledge level of concept i, has the minimum value of 0 (no knowledge) and asymptotically reaches 1 (maximum knowledge) (Yudelson et al., 2007). This means that similarity between an example and a problem increases as the number of matching concepts for which the student still lacks knowledge grows. In other words, it favors examples that have more concepts that are similar to the problem, but that have not yet been sufficiently learned.

When the association coefficient is used (Equation 2), instead of using the number of concepts that are *common* and *not common* to the example and problem, we use the degree to which these concepts can be learned. This way, examples are favored when there are no mismatched concepts or when such concepts are already learned, and there is more to be learned from the matched concepts. In this way, the number of the mismatching concepts in the problems and examples that student does not know is kept to a minimum. For this personalized approach, the two parameters a and b in the association coefficient (Equation 2) are calculated as follows:

$$a = \sum_{i \in C_p \cap C_e} (1 - k_i) \qquad b = \sum_{i \in C_p \triangle C_e} (1 - k_i) \tag{8}$$

where $C_p \cap C_e$ and $C_p \triangle C_e$ is the set of concepts *common* and *not common* to the problem p and example e, respectively; and k_i is the knowledge level of student in concept i, has the minimum value of 0 (no knowledge) and asymptotically reaches 1 (maximum knowledge). The term $1 - k_i$ is the amount that is not learned in the concept i.

Besides using information about a student's knowledge levels, we attempted to consider the learning goal of the problem; namely, the course concepts that the problem should support, according to the teacher's intention. As a proxy for the learning goal, we use all targeted concepts of the course topic to which the problem belongs (targeted concepts of a topic are concepts that are introduced and learned within the topic). Subtree similarity approaches (cosine similarity and association coefficient) use this goal set for determining whether or not a concept should increase the similarity. In cosine similarity approaches, concepts that are not in the goal set are assigned a weight of zero. The association coefficient, on the other hand, treats concepts that are not in the goal set the same way as the mismatching concepts.

To illustrate this further, consider a case where a personalized goal-based approach is used for calculating the similarity value between a problem with concepts $\{x, y, w\}$ and an example with concepts $\{x, z, w\}$. Also, assume that the goal set contains only one concept and is equal to $\{w\}$. In the case that association coefficient is used for calculating the similarity value, the matching set of concepts will contain concepts that appear both in the problem and example and are also in the goal set, which in this case is $\{w\}$. On the other hand, the mismatching set of concepts will contain concepts that did not appear in both problem and example or they were not in the goal set, here $\{x, y, z\}$. Then, Equation 8 should be used to calculate a and b with $C_p \cap C_e = \{w\}$ and $C_p \triangle C_e = \{x, y, z\}$. Similarity value can be obtained by using value of a and b in Equation 2.

When cosine similarity is used for calculating the similarity value, the concept vector will be $< I_x, I_y, I_w, I_z >$, with each element I_j being an indicator for concept j and

$j \in \{x, y, w, z\}$. The indicator I_j is zero if the concept j is not in the goal set, otherwise it is equal to student's lack of knowledge in concept j, i.e. $1 - k_j$ (k_j is student's knowledge level in concept j). Here, since the goal set contains only concept w, the vectors of the problem and example will be $< 0, 0, I_w, 0 >$ and assuming that student's knowledge level in concept w is 0.4, we get $< 0, 0, 0.6, 0 >$. This vector will be used as vectors of P and E in Equation 1 to get the similarity value.

All of the aforementioned similarity approaches and summary of their characteristics are presented in Table 3. For the sake of simplicity in the rest of the paper, we will refer to each approach by its mnemonic.

Table 3. Characteristics and mnemonics of similarity approaches for finding relevant examples for problems

Approach (Mnemonic)	Non-Structural	Structural	Personalized	
			Goal-free	Goal-based
Association coefficient (A)	✓			
Cosine similarity (C)	✓			
WMM with association coefficient (SA)		✓		
WMM with cosine similarity (SC)		✓		
Tree edit distance (ST)		✓		
Re-ranking approach (PR)			✓	
Average approach (PAVG)			✓	
Personalized association coefficient (PA)	✓		✓	
Personalized cosine similarity (PC)	✓		✓	
Personalized WMM with association coefficient (PSA)		✓	✓	
Personalized WMM with cosine similarity (PSC)		✓	✓	
Personalized goal-based association coefficient (PGA)	✓		✓	✓
Personalized goal-based cosine similarity (PGC)	✓		✓	✓
Personalized goal-based WMM with association coefficient (PGSA)		✓	✓	✓
Personalized goal-based WMM with cosine similarity (PGSC)		✓	✓	✓

4.4. *Correlation analysis of the suggested approaches*

While we suggested a range of different similarity approaches above, the extent to which they truly differ is unclear. It could be that some pairs of the approaches, while looking different, in reality produce almost equivalent results. In this case, one of these approaches could be excluded from further investigation. To see whether the proposed approaches are similar, we measured (a) Kendall's τ and (b) the overlap ratio between the lists of the top five examples that are generated by each pair of approaches for the questions that each subject attempted during the study. Figure 6 summarizes the similarities between the recommendations that are made by each pair of approaches. The value in each cell represents the mean±SD of the Kendall's τ (Figure 6(a)) and the mean±SD of the overlap ratio (Figure 6(b)) between the corresponding approaches in the row and column. A light gray color denotes a lower value and a dark gray color denotes a higher value.

Apparently, as we can see from either plot, each approach has the highest correlation/overlap with itself and a considerably lower correlation with other approaches[2]. All in all, Figure 6 suggests that all of the approaches in our study generate sufficiently different lists of recommendations and could be considered to be distinct approaches for selection of relevant examples for problems. Therefore, all of them are considered in the comparative analysis.

[2]The personalized approaches depend on the student's level of knowledge for generating ranked list of examples for a problem; thus, their corresponding value on the diagonal is not equal to one.

5. The study

To evaluate our recommendation approaches, we collected data in two different ways. First, we ran a user study to collect student data that could be used to compare similarity approaches. Second, we collected similar kind of data from experts. This section explains how the data was collected, and presents results of an inter-rater agreement and reliability analysis that was carried out to check the quality of the data and ensure the validity of our evaluation.

5.1. *User data collection study*

5.1.1. Subjects

Participants in data collection were graduate and undergraduate students at the University of Pittsburgh and Carnegie Mellon University from several disciplines, including engineering, information sciences, physics, and arts and sciences. Participants were paid $12 per hour for their participation and the whole task lasted about 75 minutes on average, with an approximate range of 30 to 150 minutes. During the study, from January 2014 to March 2014, we recruited 42 participants and 2 pilot subjects.

5.1.2. The materials

Data collection used learning content for six Java topics: Variables and Operations, Decisions and Boolean Expressions, Loops, Arrays, Classes and Objects, and Inheritance. Each topic contained four problems and a set of examples. In total, there were 82 Java examples and 24 Java problems associated with these topics. Table 4 provides details of the complexity of both problems and examples. We divided the problems into two complexity categories (low-difficulty and high-difficulty) by comparing the number of distinct concepts in a problem into a median of distinct concepts, which was 15.5. A problem was labeled as a low-difficulty problem when it had 15 or fewer concepts, and as a high-difficulty problem otherwise. Figure 7 illustrates instances of a low- and high-difficulty problem. The problem code in Figure 7(b) depends on an additional class, which is shown as a second tab in the problem interface. The study also included a pretest with the goal of assigning students to topics of proper complexity. The pretest consisted of eight questions on the target topics and was crafted by a domain expert.

Table 4. Complexity of problems and examples in the study. In problems or examples that had additional classes, the value of each variable reports the variable summed over all of the classes.

	Problems (N = 24)				Examples (N=82)			
	Mean (SD)	Min	Max	Median	Mean (SD)	Min	Max	Median
Distinct concepts	21 (13.09)	10	52	15.5	16.12 (3.26)	8	25	16
Total concepts	55.92 (73.56)	10	223	22	26.6 (13.22)	13	68	21.5
Total lines	32.21 (38.28)	4	110	10	12.95 (10.64)	4	60	10

5.1.3. The procedure

The procedure for data collection was as follows: first, the subject had to read and sign the consent form. Then, they were given the pretest and after that a brief description

about the experiment was provided. In the next step, the subjects performed the main task – problem answering and example rating. Each participant was asked to work with three topics, based on their pretest scores; they had to solve all problems allocated to these topics and rate the examples that jHelp recommended for each of these problems.

Examples were presented to subjects in two contexts: context 1, after the subject failed to solve a problem during the *solving phase*; and context 2, during the *rating phase* at the end of the work with each problem before moving to the next problem. In each phase, we asked users to react to the following statement: "The above example is helpful for me to solve the exercise". We used a four-point rating scale: not helpful at all, not helpful, helpful, and very helpful, coded from zero to three, respectively.

The *solving phase* and the *rating phase* differed in two ways: first, in the number of similarity approaches that were used to recommend examples for the problem, and second, in whether the rating was required or not. The first *solving phase* simulated the natural use of examples for help in problem-solving. To support this natural use, subjects were not required to check more examples that they needed for problem-solving, and even rating the examined example was optional. In this phase, if the subject's answer was incorrect, one similarity approach was randomly chosen and used to select five relevant examples for the problem. The recommended examples were presented to the subject in order of decreasing relevance, in a similar way as Figure 2. The second "rating phase" started when the subject clicked the 'Finish Solving' button in the problem window (See Figure 8). In this phase, the subject had to rate the top two relevant examples selected by four approaches, with respect to their helpfulness for the problem that was being solved. Overlapping examples were removed in this context. The merged set of recommended examples was presented in a random order to minimize any possible learning effects. Unlike the "solving phase", rating was mandatory in this phase to make sure that ratings were collected from all subjects. However, the examples that were voluntary rated during the problem-solving process were shown as already rated, and re-rating these was not required. We used four essentially different similarity approaches (one approach from each of the similarity categories: *cosine similarity* in the non-structural, *tree edit distance* in the structural, and *re-ranking* and *average* approach in the personalized category) to assure the collection of a sufficient number of rated examples for each problem that we could later use in the evaluation.

5.2. Expert data collection study

We conducted a subsequent study to collect judgments of domain experts for evaluation of similarity approaches from the prospects of experts. Three experts with experience in teaching Java participated in the study. To make the expert dataset consistent with the student dataset, expert data collection was set up to collect ratings of experts for examples that subjects have rated for each problem. There were a total of 217 examples that were rated by subjects for the 24 problems. Each expert had to rate the helpfulness of the examples for solving problems, using an interface similar to Figure 8. The number of examples that experts had to rate for each problem ranged from 6 to 13 with median at 8.5.

5.3. *Inter-rater agreement and reliability analysis*

We explored agreement among raters and data reliability using a quadratic weighted Kappa coefficient (κ) and Cronbach's alpha (α), respectively. We started by calculating the aforementioned statistics for all possible pairs of subjects. The mean quadratic weighted Kappa between subject pairs was low ($\kappa = 0.20$), as was the mean Cronbach's alpha ($\alpha = 0.28$).

To further explore whether agreement or reliability level relied on the user's overall knowledge of the domain, we classified subjects into two main groups by comparing the pretest scores with the median score of 5 (see Table 5). Subjects with pretest scores ranging from 0 to 4 were assigned to the low pretest group, and subjects with score ranging from 5 to 8 were assigned to the high pretest group. The mean quadratic weighted Kappa and Cronbach's alpha between subject pairs were found to be low, even as we looked at subjects with low and high pretest scores separately. Yet, high-pretest students ($\kappa = 0.24, \alpha = 0.38$) showed higher agreement and reliability, as compared with low-pretest students ($\kappa = 0.14, \alpha = 0.18$).

Table 5. Summary statistics for pretests

	Min-Max	Median	Mean (SD)
Low (n=17)	0-4	2	1.65 (1.50)
High (n=25)	5-8	6	6 (0.82)
TOTAL (n=42)	0-8	5	4.24 (2.44)

To investigate the extent to which each subject agreed with the "crowd wisdom", we measured the agreement level and reliability between each subject's ratings with a majority vote of all subjects over all items (i.e., problem-example pairs) that the subject has rated. The majority vote was the most frequent rating given to each item. In cases of ties between ratings, the one with the lower relevance score was used. Figure 9 illustrates the level of agreement and reliability between each subject and the overall crowd wisdom. Each point in the plot represents one subject. We colored the points in the figure based on the pretest group of subjects, with the dark gray point representing a subject with a high pretest score and the light gray point representing a subject with a low pretest score.

As the figure shows, the agreement between each subject and overall crowd wisdom changes from poor (less than 0.2) to good (0.61–0.8). The reliability values differs across subjects too, ranging from unacceptable (less than 0.5) to good (0.81–0.9). While students in the high-pretest group tended to have a generally higher agreement with overall crowd wisdom, the agreement and reliability level ranges from poor or unacceptable to good in both the low- and high-pretest groups. We found no significant difference either in the mean agreement or in the mean reliability between these two groups.

This data points to the fact that some subjects rated the helpfulness of the same example for a problem very differently from others and from the majority vote. This is an important observation that suggests that the judgment of an example's relevance for a problem is a challenging task for students, and that it might also be challenging to use student data as the key source for the evaluation of recommendation approaches. This observation was one of the reasons for collecting and using expert data, in addition to student data, in the evaluation process.

Inter-expert agreement was assessed by calculating quadratic weighted Kappa statistics for all possible expert pairs and was expressed as the mean average. The mean quadratic weighted Kappa showed a fair to moderate agreement of 0.51. Cronbach's

alpha indicated an acceptable inter-expert reliability as well ($\alpha = 0.78$). This shows that expert data is a more reliable source for evaluating example recommendation approaches.

To explore the extent to which experts and subjects agree on their ratings, we calculated the agreement between experts' wisdom (majority vote) and subjects' wisdom (majority vote). The quadratic weighted Kappa in this case was 0.49, which indicates a moderate agreement between experts and subjects' wisdom. Interestingly, agreement was good for the high-pretest subjects ($\kappa = 0.61$) while it only reached a fair level for the low-pretest subjects ($\kappa = 0.31$). Similarly, when we looked at the overall reliability of ratings, the reliability was acceptable only with high-pretest subjects ($\alpha = 0.76$), and was low when we considered all subjects together ($\alpha = 0.66$) or subjects with low pretests ($\alpha = 0.48$). These observations demonstrate that collective opinion of subjects with high pretest scores was closer to experts' wisdom than the opinion of low-pretest subjects. In other words, as also indicated in Figure 9, high-pretest students might be more reliable raters as a group for our study.

6. Results of the comparative analysis

In total, we collected 2541 rated problem-example pairs for the 24 problems rated by subjects in the user study. Table 6 provides a summary of the collected ratings from the subjects. For the expert study, we collected a total of 651 ratings, which consisted of 217 ratings from each of the three experts for the problem-example pairs that were rated by subjects. To evaluate the helpfulness of each approach, we first merged ratings that pointed to examples *not helpful at all* with *not helpful* examples and then aggregated multiple judgments that were available for each problem-example pair using the majority rule approach (Kazai et al., 2011, 2009; Voorhees and Tice, 2000). In cases of ties between two judgments for the same pair, the one with the lower relevance score was used; and in cases of ties between three judgments, the middle rating (*helpful*) was selected as the majority vote.

Table 6. Summary of the collected ratings from subjects, by problem complexity level

Problems	#Raters	#Distinct problem-examples pairs	#Total problem-examples pairs
Low-Difficulty (N=12)	42	108	1660
High-Difficulty (N=12)	42	109	881
TOTAL	42	217	2541

To analyze the approaches from the prospects of subjects and experts, we considered users' and experts' feedback for the top–3 example recommendations generated by each approach for each problem used in the study. The choice of the top–3 was motivated by our observations of student difficulties in distinguishing the most useful examples. In this situation, recommending the top–3 examples might be the most appropriate use of recommendation approaches and the most meaningful to evaluate. As explained above, we treated the majority vote for each top–3 recommended examples as the ground truth for the quality evaluation. Table 7 shows the number of user ratings available for the top–3 recommended examples, generated by each similarity approach. As the table data shows, a sufficient number of ratings were available to evaluate each similarity approach. The number of distinct problem-example pairs available for the evaluation of each approach ranged between 43 and 89 (with an average of 63.6). Note that the number of pairs supported by experts' ratings is the same as in the case of

subjects' ratings, since the experts rated the same problem-example pairs that were rated by subjects.

Table 7. Number of ratings available for the top–3 examples selected by each approach for the problems in the study

Approach	#Problems	#Distinct problem-example pairs
A	24	58
C	24	70
SA	22	43
SC	23	45
ST	22	49
PR	20	56
PAVG	20	60
PA	24	85
PC	24	89
PSA	23	62
PSC	24	77
PGA	24	71
PGC	23	55
PGSA	24	69
PGSC	24	65

Since the collected data did not contain the relevance information for all possible problem-example pairs, there were cases in which one or more examples among the top–3 list selected by an approach for a problem had not been rated by any subjects. It has been shown that metrics that can be reliably used for performance evaluation when relevance data is incomplete include normalized discounted cumulative gain (*nDCG*), *Q-measure*, and *AP* (Sakai, 2007; Sakai and Kando, 2008). Among these metrics, we selected nDCG, one of the most popular evaluation metrics in the field of recommender systems, to evaluate our approaches[3].

We evaluated the proposed approaches using the data collected during both the user and expert studies. We started by comparing the approaches that we used for data collection. As mentioned earlier, we used *cosine similarity* in the non-structural, *tree edit distance* in the structural, and the *re-ranking* and *average* approach in the personalized category. The preliminary evaluation of these approaches using the nDCG metric showed that personalized approaches in either the *re-ranking* or *average* form performed worse than the other two approaches, and were therefore excluded from later evaluations.

6.1. *Expert evaluation*

6.1.1. Majority vote analysis

We started the expert evaluation by comparing the performance of similarity approaches, based on the majority vote of the experts. We ran a mixed model to examine the impact of each similarity approach on the performance, assessed by nDCG. The model included *nDCG* as a response variable and had two factors: factor *approach*, and factor *problem complexity* to control for the nDCG variations caused by problem complexity. The baseline of the *approach* factor, against which other approaches were compared, was the association coefficient approach (A), as it was the simplest approach among the similarity approaches and only considered the frequency of concepts in similarity calculations.

We found that the *approach* factor had a significant effect on nDCG ($p < .0001$). Among the similarity approaches, only two personalized approaches that were variants

[3]Using AP and Q-measure for the evaluation did not substantially change the results.

of cosine similarity, namely approach PC ($p = .018$) and $PGSC$ ($p = .001$), had a significantly higher nDCG than the baseline approach, association coefficient (A). The positive effect of the personalized approach PA, a personalized variant of the association coefficient, reached only marginal significance ($p = .076$). Results from this analysis are plotted in Figure 10 to show the predicted marginal means of nDCG for each approach. This analysis, as illustrated in Figure 10, indicated that the majority of experts gave the highest relevance ratings to the examples selected by the personalized goal-based structural variant of cosine similarity ($PGSC$). The second and third best-performing approaches were personalized variant of cosine similarity (PC) and personalized variant of association coefficient (PA), respectively.

6.1.2. Distance analysis

We postulated that a good approach should generate a ranking of examples that is comparable with a single expert's view. The majority vote of three experts is usually more reliable than an opinion of a single expert, but it is also a very high target to reach. For example, it's probably too much to demand that a recommendation approach should be comparable to the majority vote of experts (namely, that it is *better* than a single expert). We can argue that an approach could be considered to be very good if it is at least as good as a single expert. To compare an approach to a single expert, we select two of the experts as *reference points*, and then calculate the distance of the third expert and each similarity approach (i.e., *source points*), from the reference points. If the cumulative distance of an approach to reference experts is less than the cumulative distance of source expert to reference experts, we can consider this approach to generate rankings that is at least as good as a single expert's judgment.

We define the distance between a reference expert and a ranking approach or a reference expert and a source expert as an inverse function of the approach nDCG. The distance between a reference expert and a ranking approach is calculated using the expert's ratings for example list generated by the approach for each problem. The distance between a reference expert and a source expert is calculated in a similar way, by considering the source expert to act like an approach. That is, we use the ratings of the source expert to generate a ranked list of examples for each problem in descending order, sorted by example helpfulness. Then, we use ratings of the reference expert to evaluate that ranked list in terms of nDCG.

We compared the cumulative distances of different source points from two experts by running mixed models with *cumulative distance* as the response variable, and *source points* and *problem complexity* as the fixed factors. The latter factor was included in the model to control for distance variations caused by problem complexity. The factor *source points* had 14 levels, including one level for each similarity approach and one level to represent the source expert. The base level in *source points* factor against which the other categories were compared was the source expert. The effect of the source point was found to be significant on cumulative distance ($p < .0001$). The only similarity approach that had a significantly lower cumulative distance from the experts, as compared to the baseline (i.e., source expert) was the personalized goal-based structural variant of cosine similarity, namely approach $PGSC$ (p=.020) (See Table 8). On the other hand, the association coefficient (A), cosine similarity (C), personalized association coefficient (PA), personalized WMM with association coefficient (PSA), personalized WMM with cosine similarity (PSC), personalized goal-based association coefficient (PGA), personalized goal-based cosine similarity (PGC), and personalized goal-based WMM with association coefficient ($PGSA$) were all significantly worse

than a single expert (i.e., had a higher cumulative distance from the reference experts compared to the source expert). The WMM with cosine similarity (SC) was also worse than an expert, although the difference was only marginally significant. Therefore, the approach that used student's knowledge levels, the problem's goal, the problem's structure, and applied cosine similarity for the calculation of subtree similarity worked better than a single expert, in terms of inter-expert similarity. Similar results were obtained when we used different pairs of experts as reference points.

Table 8. Parameter estimates and standard errors in the mixed models for examining the relationship between cumulative distance of levels of the source points from the reference experts, the base level was the source expert.

Approach	Coef.	SE	P-value
A	0.035	0.011	**
C	0.053	0.011	***
SA	0.010	0.011	
SC	0.020	0.011	.
ST	0.015	0.011	
PA	0.013	0.003	***
PC	0.003	0.003	
PSA	0.020	0.003	***
PSC	0.016	0.003	***
PGA	0.024	0.003	***
PGC	0.014	0.003	***
PGSA	0.038	0.003	***
PGSC	-0.006	0.003	*

. :< .1; ∗ :< .05; ∗∗ :< .01; ∗∗∗ :< .001

6.2. *User evaluation*

We conducted a mixed-model analysis to see the impact of *approach* on the *nDCG* from the majority of the subjects' point of view. Similar to the previous models, we controlled for the nDCG variations caused by problem complexity, and the base level in the approach factor was association coefficient approach (A).

The model showed a significant effect for approach on nDCG ($p < .0001$). The predicted marginal means of nDCG for each approach are shown in Figure 11. As the figure shows, some approaches performed as well as association coefficient approach (A) and some performed worse than it. More specifically, WMM with association coefficient (SA) ($p = .022$), WMM with cosine similarity (SC) ($p = .014$), personalized association coefficient (PA) ($p = .007$), personalized cosine similarity (PC) ($p < .001$), and personalized goal-based cosine similarity (PGC) ($p = .065$) were the approaches that performed worse than the association coefficient (A). The difference between the association coefficient (A) and other approaches was not significant. These results indicate that subjects' wisdom led to the selection of a different set of best-performing approaches. Unlike experts who picked personalized cosine similarity (PC) and the personalized association coefficient (PA) as second and third best and better than the baseline association coefficient (A), subjects favored personalized cosine similarity (PC) and the personalized association coefficient (PA) less than the association coefficient (A). However, the experts' top performing approach, personalized goal-based

WMM with cosine similarity ($PGSC$), was still among the top performing approaches from the subjects' perspective.

In addition to comparing the performance of different approaches, it is interesting to compare their stability reflected by the confidence intervals. As could be observed in Figure 11, the 95% confidence intervals are much smaller for the personalized than for the non-personalized approaches. The expert evaluation data shows the same trend. This gives us some evidence that personalized approaches accounting for student's position and needs within the current topic offer a more stable performance (from the prospect of human judges) across a set of problems than non-personalized approaches.

7. Discussion

The current study investigated different approaches for recommending examples that might be helpful for students who failed to solve a programming problem. We treated the example recommendation problem as a problem of intelligent linking between problems and examples and used the ideas of semantic-level similarity-based linking to generate links between the target problem and different examples. To determine the approach that offers the best recommendation performance, we explored a range of concept-level similarity approaches that assess the similarity of problems and examples in terms of programming concepts measured within small fragments (structures), as well as within the content as a whole. In addition, we explored the value of considering student factors, such as expected student's knowledge levels and learning goal, as defined by the student's position in a course. Although our focus was on the Java domain, our proposed approaches are applicable to other programming domains, as long as fine-grained concepts are obtained for content items.

We evaluated the performance of various similarity approaches using relevance data collected from users and experts. Analysis of expert data revealed that two personalized variants of the cosine similarity approach, personalized cosine similarity (PC) and personalized goal-based WMM with cosine similarity ($PGSC$), performed significantly better than the simplest non-personalized approach, namely the association coefficient (A). Among these two top-performing approaches, only approach $PGSC$ that considered both expected student's knowledge levels and goal was found to generate a ranking of examples that is as good as a single expert's judgment. Analysis of user data, on the other hand, showed the approaches of cosine similarity (C), tree edit distance (ST), personalized WMM with association coefficient (PSA), personalized WMM with cosine similarity (PSC), personalized goal-based association coefficient (PGA), personalized goal-based WMM with association coefficient ($PGSA$), and personalized goal-based WMM with cosine similarity ($PGSC$) achieving comparable performance to the simplest non-personalized approach, that of the association coefficient (A), and other approaches as being inferior.

To sum up, the results of the user and expert study showed that the approach that selects the most helpful examples for a problem is that of a personalized goal-based WMM with cosine similarity; namely, approach $PGSC$, which calculates the similarity of examples to a problem by (a) focusing on the examples that are similar to the problem by the structure of the concepts rather than concept coverage (i.e., being a structural similarity approach), (b) personalizing example selection to select examples that match student's knowledge levels and learning goal (i.e., being a personalized similarity approach), and (c) calculating the concept-level similarity of the examples to a problem using a cosine metric. In addition, our data showed that personalized

approaches that selected remedial examples by considering student's expected level of knowledge and/or learning goal delivered more stable performance across a set of problems than non-personalized approaches. In other words, non-personalized approaches could frequently produce poor recommendations. This is a serious concern in an educational recommendation context where users are frequently not able to recognize such poor recommendations.

While the analysis of both user and expert data points to personalized goal-based WMM with cosine similarity ($PGSC$) as the best approach to use for remedial example recommendation, it also stresses that the students' majority vote was considerably different from the experts' vote. This is an important finding. The analysis of data collected for our study shows that students' opinion about example relevance should be considered with caution. In particular, both the agreement among students and the agreement between students and experts were considerably low. Given this data, it was not evident to what extent students were able to distinguish the value of different examples in helping to solve target problems. Our data also showed that as a group, high-pretest students were much closer in their judgment to experts, although some students in this group still had a considerably low agreement with the group as a whole. This indicates that high-pretest students could be considered as a better source of data for a study like ours than low-pretest students, yet not replacing the experts. The fact that students are poor judges of what instruction is helpful is also mentioned in (Bjork et al., 2013; Clark and Mayer, 2011).

We hope that the results of our study expand existing work on example-based problem-solving for learning programming Gross et al. (2014); Weber (1991); Weber and Mollenberg (1994). Adding to the earlier results reporting the benefit of using examples as help in problem-solving, it provides some guidance in selecting or developing example recommendation approaches and offers specific solutions. Specifically, our data points that it might be wise to prefer personalized approaches for remedial example recommendations. Among the approaches that we explored in our study, the personalized goal-based WMM with cosine similarity ($PGSC$) approach that accounts for both expected student's knowledge levels and learning goal delivered excellent performance in expert evaluation and was comparable to other top approaches in user evaluation. Thus, we believe that using personalized goal-based remedial example recommendations to help with problem-solving could improve computer science education.

While our study helped us to select the best-performing approach, the study was limited to a subset of possible ways to personalize similarity-based example recommendations. Furthermore, the assumption that the usefulness of an example depends on the similarity of the example code to the problem code is, to some degree, a simplistic assumption. Although, the present work examined personalized similarity approaches to account for differences in student's knowledge needs, further studies of personalized remedial recommendations should be performed to explore other personalization ideas. Another limitation of our study is its lab-based nature. To properly explore personalized recommendation approaches, we need data that has been collected in a real classroom context. As mentioned above, jHelp has been used in real classrooms for several semesters; however, the volume of collected relevance feedback is too low to run a thorough evaluation. We do plan to continue data collection in the context of regular classrooms and MOOCs, and hope to use this data in future attempts to examine the influence of the top-performing recommendation approaches on students' learning (rather than their acceptance of recommendations). Finally, we would like to

investigate the extent to which the findings of this study can be generalized to more complex coding assignments as well as other programming domains.

Acknowledgement

This work has been partially supported by the Advanced Distributed Learning Initiative under the contract W911QY13C0032.

References

Bjork, R. A., Dunlosky, J., and Kornell, N. (2013). Self-regulated learning: Beliefs, techniques, and illusions. *Annual Review of Psychology*, 64:417–444.

Brusilovsky, P. and Peylo, C. (2003). Adaptive and intelligent web-based educational systems. *International Journal of Artificial Intelligence in Education*, 13(2):159–172.

Brusilovsky, P., Yudelson, M., and Hsiao, I.-H. (2009). Problem solving examples as first class objects in educational digital libraries: Three obstacles to overcome. *Journal of Educational Multimedia and Hypermedia*, 18(3):267–288.

Carmel, D., Uziel, E., Guy, I., Mass, Y., and Roitman, H. (2012). Folksonomy-based term extraction for word cloud generation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4):60.

Carr, L., Hall, W., Bechhofer, S., and Goble, C. (2001). Conceptual linking: ontology-based open hypermedia. In *Proceedings of the 10th international conference on World Wide Web*, pages 334–342. ACM.

Clark, R. C. and Mayer, R. E. (2011). *E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning*. John Wiley & Sons.

Crampes, M. and Ranwez, S. (2000). Ontology-supported and ontology-driven conceptual navigation on the world wide web. In *Proceedings of the eleventh ACM on Hypertext and hypermedia*, pages 191–199. ACM.

Csomai, A. and Mihalcea, R. (2008). Linking documents to encyclopedic knowledge. *Intelligent Systems, IEEE*, 23(5):34–41.

Dolog, P., Henze, N., and Nejdl, W. (2003). Logic-based open hypermedia for the semantic web. In *Proceedings of the Int. Workshop on Hypermedia and the Semantic Web, Hypertext 2003 Conference, Nottinghan, UK*. Citeseer.

Gross, S., Mokbel, B., Hammer, B., and Pinkwart, N. (2014). How to select an example? a comparison of selection strategies in example-based learning. In *Intelligent Tutoring Systems*, pages 340–347. Springer.

Hosseini, R. and Brusilovsky, P. (2013). Javaparser: A fine-grain concept indexing tool for java problems. In *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, pages 60–63.

Hosseini, R. and Brusilovsky, P. (2014). Example-based problem solving support using concept analysis of programming content. In *Intelligent Tutoring Systems*, pages 683–685. Springer.

Hsiao, I.-H., Sosnovsky, S., and Brusilovsky, P. (2010). Guiding students to the right questions: adaptive navigation support in an e-learning system for java programming. *Journal of Computer Assisted Learning*, 26(4):270–283.

Kazai, G., Koolen, M., Kamps, J., Doucet, A., and Landoni, M. (2011). Overview of the inex 2010 book track: Scaling up the evaluation using crowdsourcing. In *Comparative Evaluation of Focused Retrieval*, pages 98–117. Springer.

Kazai, G., Milic-Frayling, N., and Costello, J. (2009). Towards methods for the collective gathering and quality control of relevance assessments. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 452–459. ACM.

Kibby, M. and Mayes, J. (1989). Towards intelligent hypertext. *Hypertext: theory into practice*, pages 164–172.

Kirschner, P. A., Sweller, J., and Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational psychologist*, 41(2):75–86.

Koedinger, K. R., Corbett, A. T., and Perfetti, C. (2012). The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science*, 36(5):757798.

Mayes, J. T., Kibby, M. R., and Watson, H. (1988). Strathtutor©: The development and evaluation of a learning-by-browsing system on the macintosh. *Computers & Education*, 12(1):221–229.

McLaren, B. M. and Isotani, S. (2011). When is it best to learn with all worked examples? In *Artificial Intelligence in Education*, pages 222–229. Springer.

McLaren, B. M., Lim, S.-J., and Koedinger, K. R. (2008). When and how often should worked examples be given to students? new results and a summary of the current state of research. In *Proceedings of the 30th annual conference of the cognitive science society*, pages 2176–2181.

McLaren, B. M., van Gog, T., Ganoe, C., Yaron, D., and Karabinos, M. (2014). Exploring the assistance dilemma: Comparing instructional support in examples and problems. In *Intelligent Tutoring Systems*, pages 354–361. Springer.

Milne, D. and Witten, I. H. (2008). Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

Najar, A. S., Mitrovic, A., and McLaren, B. M. (2014). Adaptive support versus alternating worked examples and tutored problems: Which leads to better learning? In *User Modeling, Adaptation, and Personalization*, pages 171–182. Springer.

Parameswaran, A., Garcia-Molina, H., and Rajaraman, A. (2010). Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment*, 3(1-2):566–577.

Renkl, A., Atkinson, R. K., Maier, U. H., and Staley, R. (2002). From example study to problem solving: Smooth transitions help learning. *The Journal of Experimental Education*, 70(4):293–315.

Sakai, T. (2007). Alternatives to bpref. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 71–78. ACM.

Sakai, T. and Kando, N. (2008). On information retrieval metrics designed for evaluation with incomplete relevance assessments. *Information Retrieval*, 11(5):447–470.

Tudhope, D. and Taylor, C. (1997). Navigation via similarity: automatic linking based on semantic closeness. *Information Processing & Management*, 33(2):233–242.

van Gog, T. (2011). Effects of identical example–problem and problem–example pairs on learning. *Computers & Education*, 57(2):1775–1779.

Voorhees, E. M. and Tice, D. M. (2000). Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207. ACM.

Wang, J. Z. and Taylor, W. (2007). Concept forest: A new ontology-assisted text document similarity measurement method. In *Web Intelligence, IEEE/WIC/ACM International Conference on*, pages 395–401. IEEE.

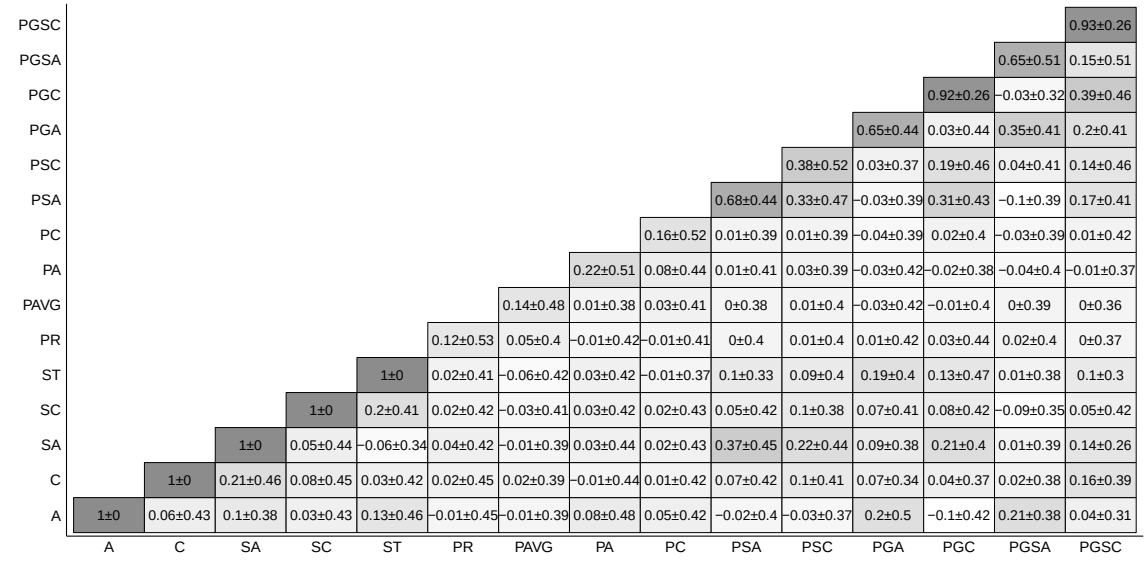
Weber, G. (1991). Explanation-based retrieval in a case-based learning model. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 522–527.

Weber, G. and Mollenberg, A. (1994). Elm-pe: A knowledge-based programming environment for learning lisp.

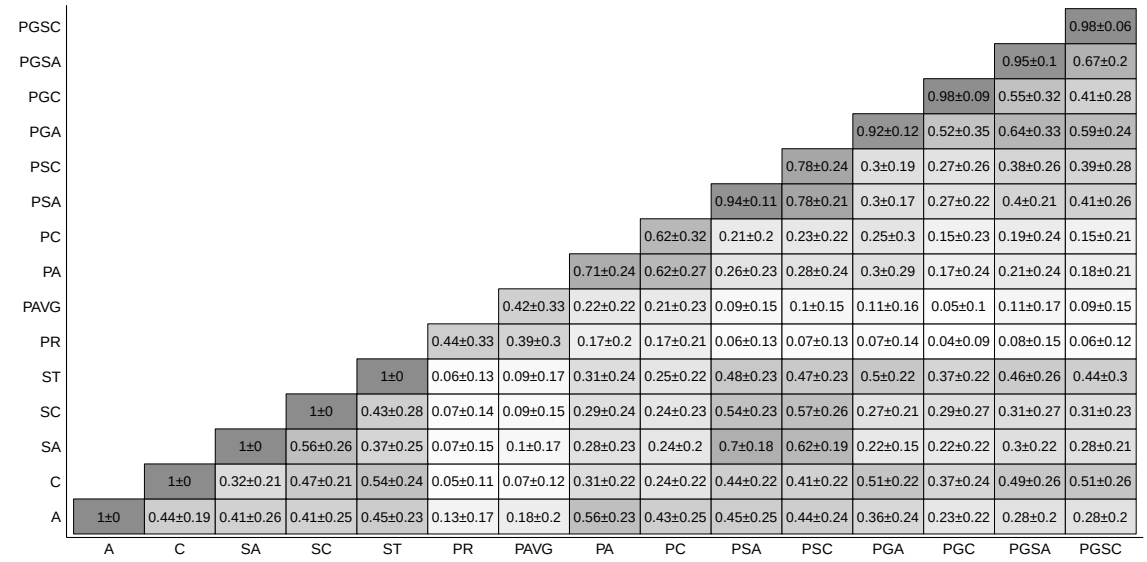
Yudelson, M., Brusilovsky, P., and Zadorozhny, V. (2007). A user modeling server for contemporary adaptive hypermedia: An evaluation of the push approach to evidence propagation. In *User Modeling 2007*, pages 27–36. Springer.

Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262.

Zhong, Y., Meacham, C. A., and Pramanik, S. (1997). A general method for tree-comparison based on subtree similarity and its use in a taxonomic database. *Biosystems*, 42(1):1–8.

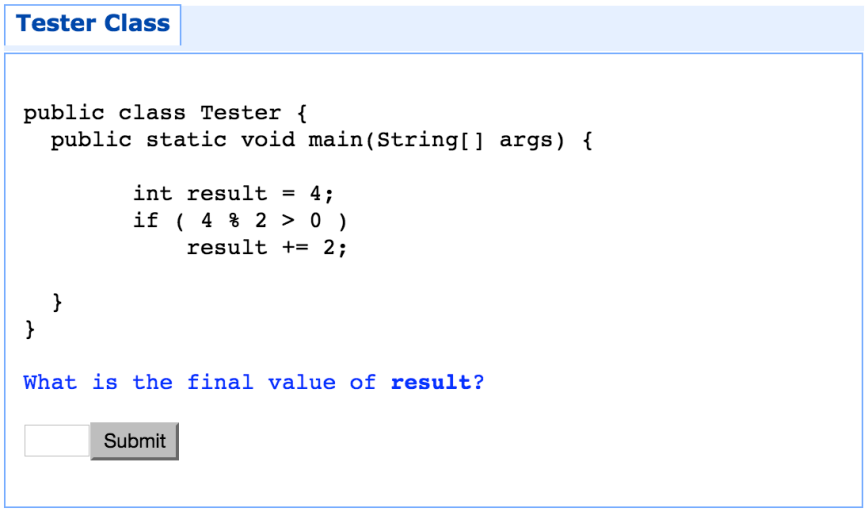


(a)

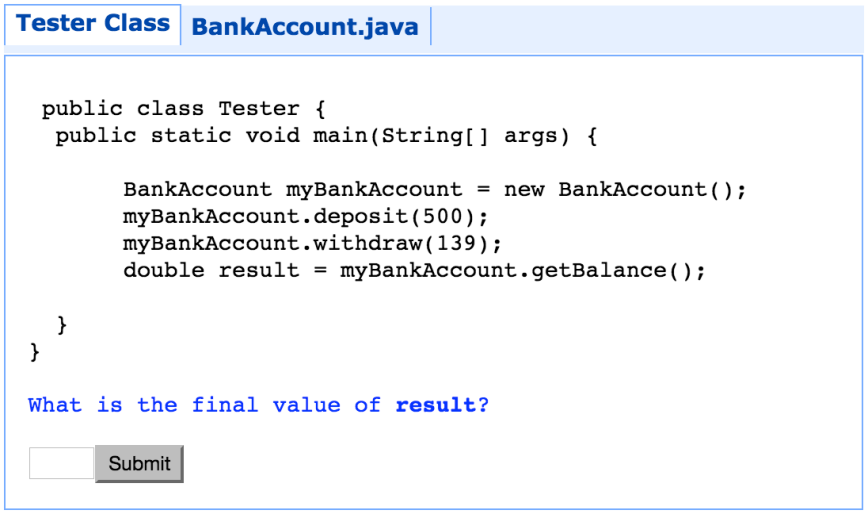


(b)

Figure 6. Heatmap plots presenting mean±SD of (a) Kendall's τ correlation and (b) overlap ratio, between the top – 5 examples recommended by each pair of approaches



(a)



(b)

Figure 7. An instance of a (a) low-difficulty and (b) high-difficulty problem in the study.

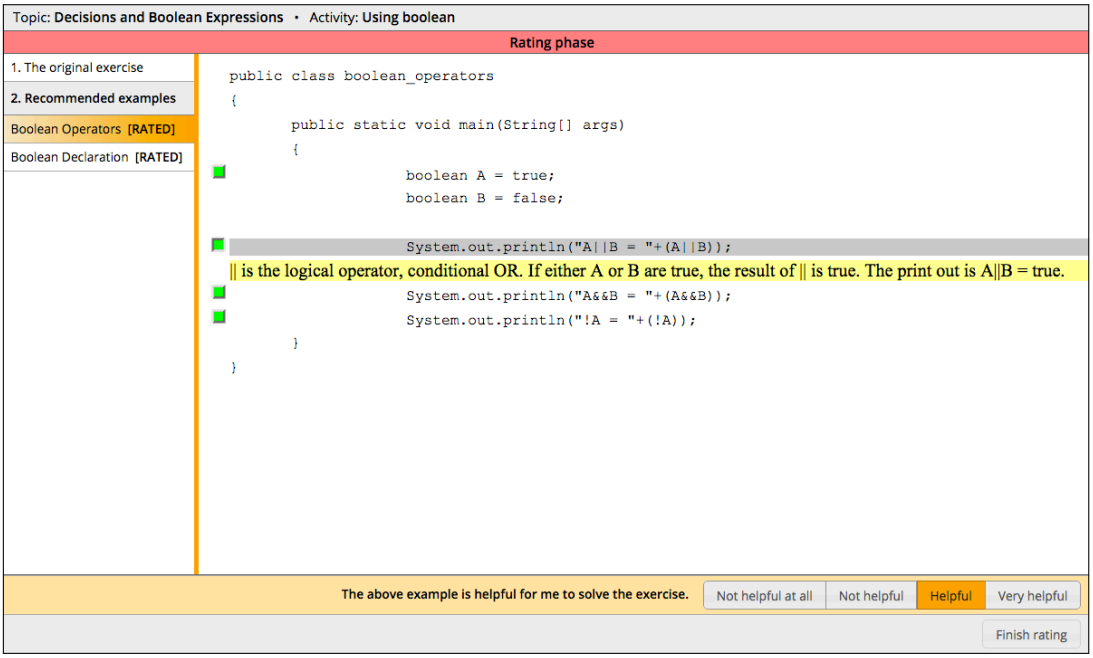
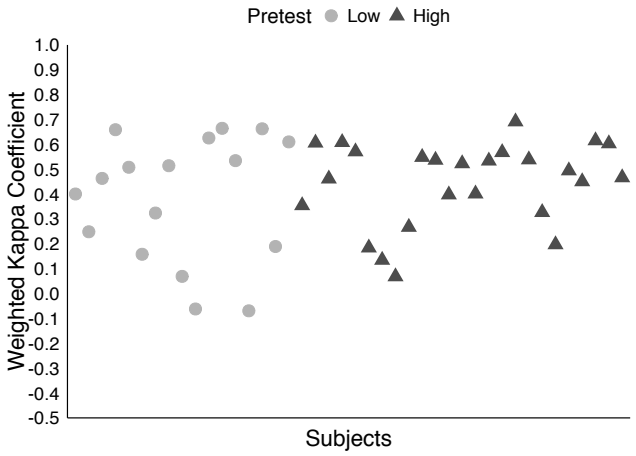
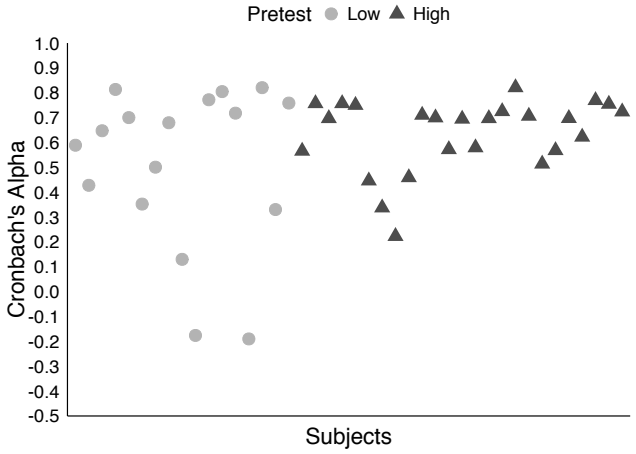


Figure 8. Ratings phase. Here, the subject rated the first recommended example as Helpful for the problem



(a)



(b)

Figure 9. The agreement level and reliability between each subject and the wisdom of the crowd, as measured by a weighted Kappa coefficient (a) and Cronbach's alpha (b), respectively. The x-axis in each plot is in an increasing order of the subject's pretest score.

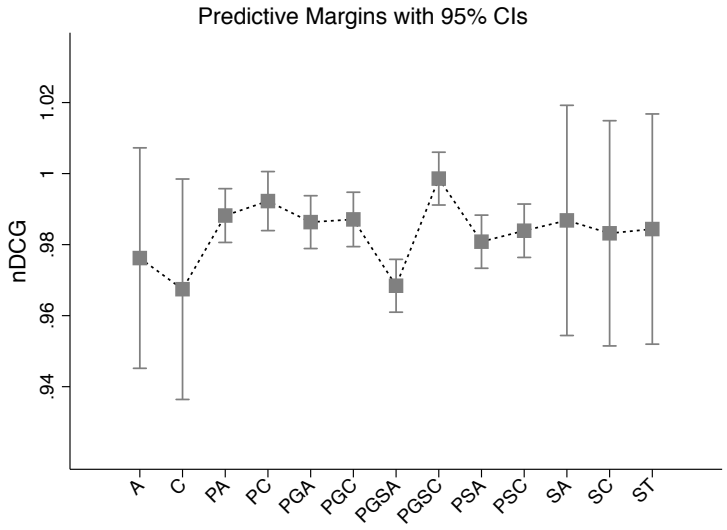


Figure 10. Influence of similarity approaches on nDCG from the prospects of experts' majority vote

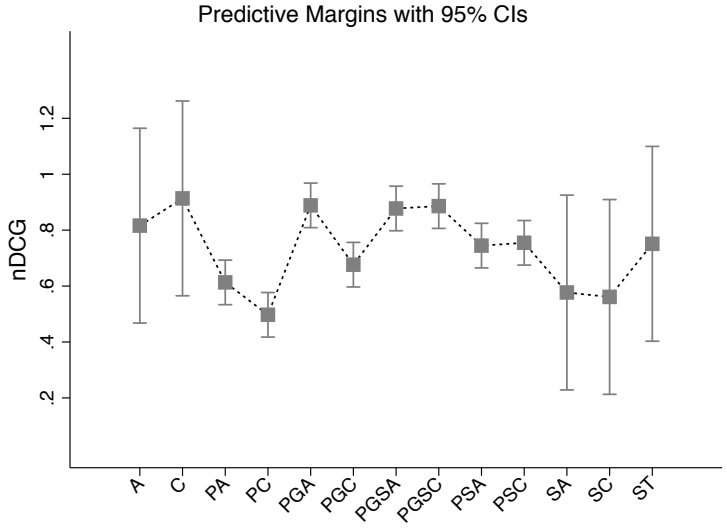


Figure 11. Influence of similarity approaches on nDCG from the prospects of subjects' majority vote