Institute for Intelligent Systems
THE UNIVERSITY OF MEMPHIS

SelfCode: An Annotated Corpus and a Model for Automated Assessment of Self-Explanation During Source Code Comprehension

Jeevan Chapagain, Zak Risha, Rabin Banjade, Priti Oli,Lasang Tamang,Peter Brusilovsky, Vasile Rus

The Florida Artificial Intelligence Research Society

FLAIRS

May,2023

Outline

- Introduction
- Related Works
- Data Sets
- Baseline Models
- Results
- Conclusion

Introduction

Introduction

- Assessment is the process of understanding and improving students learning
- Assessment is a central task in education in general and adaptive education technologies
- Assessing students’ knowledge states is key to personalized instruction
- Automated assessment provides an estimate of the mastery level of the learner

Introduction

- Assessment in various domain but not in source code comprehension focusing self-explanation
- Source code comprehension means identifying the functional pieces of a computer program
- Code comprehension is essential for both learners and professionals
- Understanding code is the most time consuming process in software maintenance, responsible for 70% of a software product’s overall life-cycle cost (Rugaber 2000)

Introduction

- Instructional strategy to improve code comprehension is self-explanation

- Students engaging in self-explanations are better learners (Chi 2000)

- Self-explanations can have different degrees of impact on various learners

- Self-explanations are helpful for learning because they involve various cognitive processes

Introduction

- Self-explanations positive impact in different domains like physics(Conati and VanLehn 2000), math (Aleven and Koedinger 2002), and programming (Tamang et al. 2020; Rus et al. 2021)

- Scaffolds learners' code comprehension processes by eliciting self-explanations and providing feedback

- Automatically assess students' self-explanations measuring how semantically similar the student self-explanations are to benchmark explanations provided by expert

Related Works

Related Works

- Overview of paraphrase identification corpora, including datasets for assessing student answers (Rus, Banjade, and Lintean 2014).

- SimLex-999 (Hill, Reichart, and Korhonen 2015) measure similarity rather than relatedness.

- Banjade and colleagues (Banjade et al. 2016) developed the DT-Grade corpus

- Mohler and Mihalcea (Mohler and Mihalcea 2009) collection of short student responses for a computer science course to assess student responses based on textual similarity

Data Collection & Annotation

Data Collection

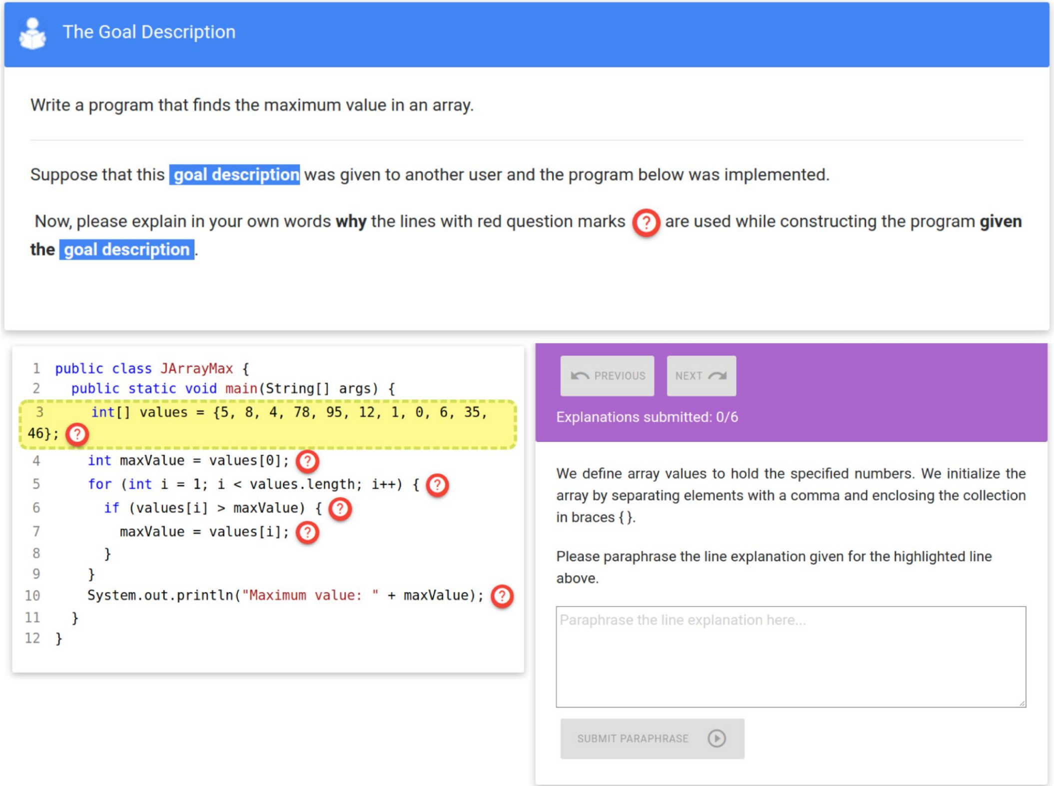

Fig1 : PCEX web application for collecting line-by-line student self explanation

Data Annotation

- 1770 sentence pairs
- Human experts annotated the sentence pairs with semantic similari ratings ranging from 1-5
- Annotated by 6 Ph.D. students having expertise in computer programming
- Annotated in two stages, with Fleiss' Kappa score of 0.33 in first stage and 0.99 in second stage.

Data Sets

Example Code	Crowd Source Explanation	Standard Explanation	Annotation Label
Int [] arr = { 14, 33, 1, 35 }	Declares the array we want to use for our assignment	We initialize the array of type int to hold the specified numbers	4
Int num = 15;	Variable declaration: declares the number we are trying to divide	We could initialize it to any positive integer greater than 1	1
Divisor += 1;	If the loop condition is true, then we increment the divisor by 1	When the divisor is not a factor of the number, we increment the variable divisor by 1	3
Int seconds = scan.nextInt();	Get the number entered by the user	We read the seconds by calling the nextInt() method because the input is an integer	2
System.out.println(“Enter an integer:”);	Ask the user to enter an integer	We prompt the user to enter an integer	5

Table 1 : Snapshot of the data set

Data Sets

Annotation Label	No. of Instances
1	529
2	507
3	419
4	253
5	62

Table 2 : SelfCode Dataset Statistics

Baseline Models

Baseline Models

- Extracted textual features from sentence pairs
- Combined textual features with different classification models
- Classification Models used:
    - Logistic Regression (LR)
    - Support Vector Machine (SVM)
    - Decision Tree (DT)
    - Naive Bayes (NB)

Baseline Models

- Features Used:
  - Word Count Difference
  - No. of overlapping words
  - No. of bi-gram overlapping words
  - Semantic Similarity score using SentenceBERT

Experiment & Results

Experiment

- 10-fold stratified cross-validation technique

- Confusion Matrix to understand the performance of various models for each annotation category.

Results

Models	Precision	Recall	F1-Score	Accuracy
LR	0.30	0.27	0.25	36.91%
DT	0.30	0.30	0.29	33.24%
SVM	0.18	0.21	0.15	30.69%
NB	**0.36**	**0.35**	**0.32**	**37.93%**

Table 2: Performance of the models with textual features (M1)

Results

Models	Precision	Recall	F1-Score	Accuracy
LR	**0.37**	**0.37**	**0.36**	**47.31%**
DT	0.32	0.33	0.32	37.25%
SVM	0.18	0.21	0.15	30.92%
NB	0.43	0.41	0.40	46.40%

Table 3: Performance of the models with textual features and sim score bert (M2)

Results

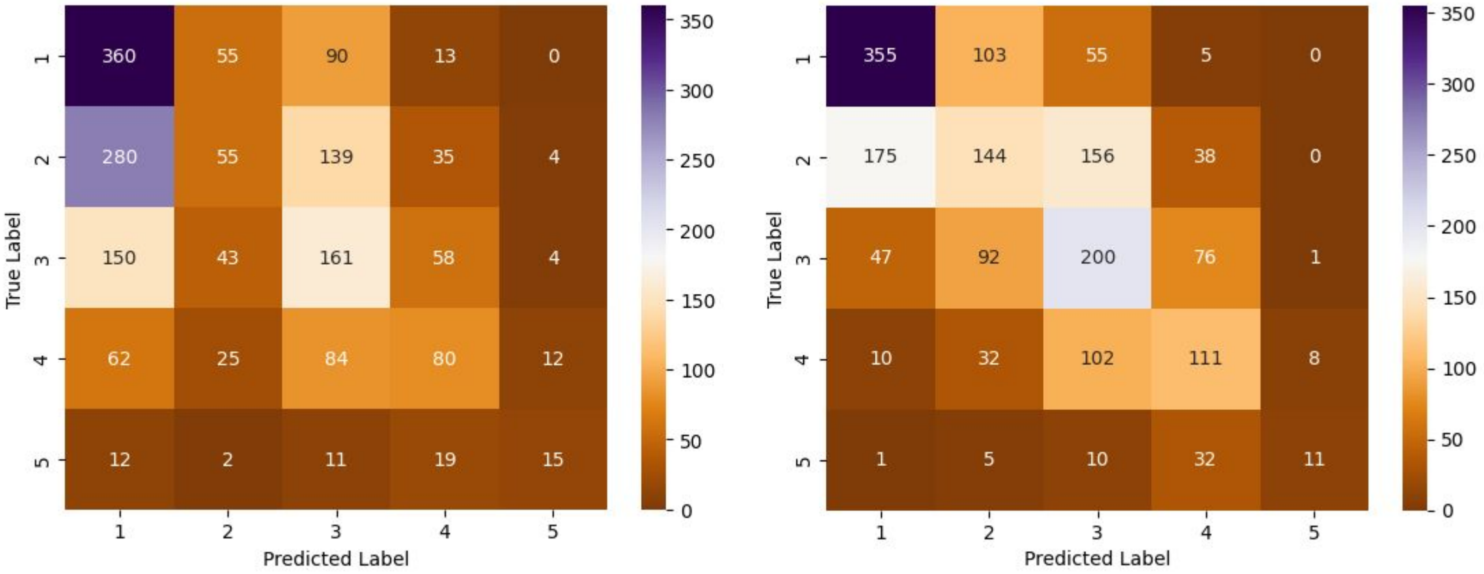

Figure 2: Confusion Matrix for M1 and M2 respectively

Conclusion

Conclusion

- SelfCode corpus which consists of crowdsourced and experts line-by-line JAVA code self-explanations annotated based on semantic similarity (https://github.com/jeevanchaps/SelfCode)

- Assist the development of supervised machine learning methods for automated assessment

- Future Work includes: a) extending the single expert explanations to multiple sentences,b) check the quality of the explanations provided by crowd workers, which can also be added as alternate explanations that help make the dataset richer

Thank You!
Jeevan Chapagain
jchpgain@memphis.edu