Knowledge-based Recommender System for Students of an Introductory Programming Course*

Extended Abstract

Oda Inanna Klemetsdal Stene
University of Bergen
Bergen, Norway

Ingrid Næss Johansen
University of Bergen
Bergen, Norway

Colin Nordgård
University of Bergen
Bergen, Norway

ABSTRACT

In this extended abstract we describe the development of a knowledge-based, personalized, recommender system for programming exercises in an introductory programming course using Python.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Social and professional topics → CS1**;

KEYWORDS

recommender system, knowledge-based recommender system, constraint-based recommender system, concept-based recommender system

ACM Reference Format:
Oda Inanna Klemetsdal Stene, Ingrid Næss Johansen, and Colin Nordgård. 2018. Knowledge-based Recommender System for Students of an Introductory Programming Course: Extended Abstract. In *Proceedings of SPLASH 2018 Student Research Competition (SPLASH SRC'18).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PROBLEM AND MOTIVATION

Computer science education can be viewed as a constructive process, where some concepts depend on others [1]. These relationships between concepts, and the fact that there are many ways to select and sequence them in any given language explain why it can be difficult for students to find programming exercises that depend exclusively on concepts they have encountered so far. A knowledge-based recommender system that knows which concepts a student has used in previous work and which concepts have been covered in lectures should be able to recommend relevant exercises to individual students based on their programming experience. This would enable students to find relevant programming exercises more effectively.

2 BACKGROUND AND RELATED WORK

Recommender systems help users make better decisions in the face of information overload by recommending relevant items to users [5]. *Knowledge-based recommender systems* rely on domain knowledge to make recommendations [6]. *Constraint-based recommendation* is a subtype of knowledge-based recommendation focused on the satisfaction of constraints [2]. An example of a constraint would be recommending programming exercises that depend exclusively on concepts that are also present in mandatory assignments.

Course, user and item models can be fine-grained or coarse-grained, with more fine-grained models enabling more specific recommendations. For example, a user model that contains information about use of the modulo operator will be able to make more specific recommendations than a model that groups together all binary operators. According to Hosseini and Brusilovsky, fine-grained indexing is critical to finding and filling gaps in student knowledge [4].

3 APPROACH AND UNIQUENESS

3.1 Context

Our recommender system is an integrated part of a larger system that will be used to deliver and grade programming assignments in an introductory course in Python programming at the University of Bergen in the fall of 2018. The system will give students access to a set of programming exercises in the form of Jupyter notebooks[2] in an online portal. Notebooks are created using the nbgrader[3] extension of Jupyter, which supports delivery and grading of student code based on unit tests. All notebooks will be automatically tested, and mandatory assignments will be manually graded as well.

*Undergraduate entry

SPLASH SRC'18, 2018, Boston, MA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[2]Jupyter is an open-source editor solution that allows for combining text and live code. See jupyter.org for more information.
[3]github.com/jupyter/nbgrader

3.2 Ontology and parser

We created an ad hoc ontology for the course by looking at the course plan and assigned reading material [3]. We then built a concept-parser that extracts concepts from a code sample by traversing the abstract syntax tree. The parser is fine-grained enough to extract the names of method and function calls as well as mathematical operations and reserved keywords in Python like *if* and *try*.

3.3 Modelling the course, users and items

The course, users and items are represented by a *course model*, *user models* and *item models*. The core of each model is a table of concepts and the number of times they have been used. The main table in the course model comes from parsing code from lectures and mandatory assignments while the main table in item models come from parsing the solutions to programming exercises. In a user model, the main table comes from parsing an individual student's solutions to programming exercises.

3.4 The Recommender System

The course, item and user models can be used to hide items that are either completed or contain concepts that are unused or unseen by a student.

User settings for the recommender system:

(1) Show or hide exercises that contain unseen concepts. *Reasoning: practice all the concepts covered in the course so far.*
(2) Show or hide exercises that contain unused concepts. *Reasoning: practice concepts the student has used. Useful if the student is ahead or behind.*
(3) Only show exercises than contain a certain concept.

4 RESULTS AND CONTRIBUTIONS

The students will start using the recommender system this August. We will collect student logs of interactions with the recommender system, as well as attempted and completed programming exercises and whether these were recommended or not. When the course has begun and the database has collected enough user data, the recommender system could be extended with recommendation techniques like content-based or collaborative filtering. We predict that the recommender system will help students navigate the programming exercises more effectively, spending less time searching for relevant exercises and more time doing them.

REFERENCES

[1] Mordechai Ben-Ari. 2001. Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching* 20, 1 (2001), 45–73.

[2] A. Felfernig and R. Burke. 2008. Constraint-based Recommender Systems: Technologies and Research Issues. In *Proceedings of the 10th International Conference on Electronic Commerce (ICEC '08)*. ACM, New York, NY, USA, Article 3, 10 pages. https://doi.org/10.1145/1409540.1409544

[3] Paul Gries, Jennifer Campbell, and Jason Montojo. 2017. *Practical Programming: An Introduction to Computer Science Using Python 3.6* (3rd ed.). Pragmatic Bookshelf.

[4] Roya Hosseini and Peter Brusilovsky. 2013. JavaParser: A fine-grain concept indexing tool for Java problems. In *CEUR Workshop Proceedings*, Vol. 1009. University of Pittsburgh, 60–63.

[5] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. Recommender systems: introduction and challenges. In *Recommender systems handbook*. Springer, 1–34.

[6] John K Tarus, Zhendong Niu, and Ghulam Mustafa. 2018. Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. *Artificial Intelligence Review* 50, 1 (2018), 21–48.