Elivate: A Real-Time Assistant for Students and Lecturers as Part of an Online CS Education Platform

Suin Kim
School of Computing
KAIST
Daejeon, South Korea
suin.kim@kaist.ac.kr

Jungkook Park
School of Computing
KAIST
Daejeon, South Korea
pjknkda@kaist.ac.kr

Jae Won Kim
School of Computing
KAIST
Daejeon, South Korea
jaewonk@kaist.ac.kr

Alice Oh
School of Computing
KAIST
Daejeon, South Korea
alice.oh@kaist.edu

L@S 2016, April 25–26, 2016, Edinburgh, UK.
ACM 978-1-4503-3726-7/16/04.
http://dx.doi.org/10.1145/2876034.2893406

Abstract

We present Elice[1], an online CS (computer science) education platform, and **Elivate**, a system for (i) taking student learning data from Elice, (ii) inferring their progress through an educational taxonomy tailored for programming education, and (iii) generating the real-time assistance for students and lecturers. Online courses suffer from high average attrition rates, and early prediction can enable early personalized feedback to motivate and assist students who may be having difficulties. Elice captures detailed student learning activities including intermediate revisions of code as students make progress toward completing their programming exercises and timestamps of student logins and submissions. Elivate then takes those data to analyze each student's progress and estimate the time to completion. In doing so, Elivate uses a learning taxonomy and automatic clustering of source code revisions. Using more than 240,000 code revisions generated by 1,000 students, we demonstrate how Elivate processes large-scale student data and generates appropriate real-time feedback for students.

Demonstration

A major problem in online education is the high attrition rate which can be reduced by early feedback and one-on-one assistance for students experiencing problems.

[1]https://www.elice.io

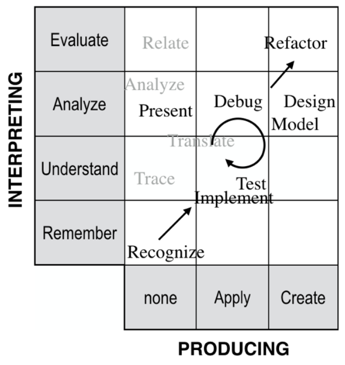


Figure 1: 6 stages of learning, from Fuller's CS-specific educational taxonomy.

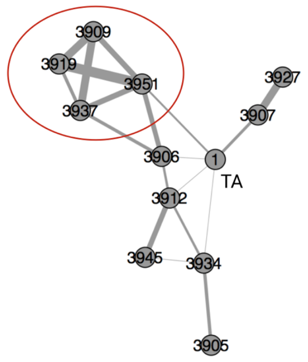
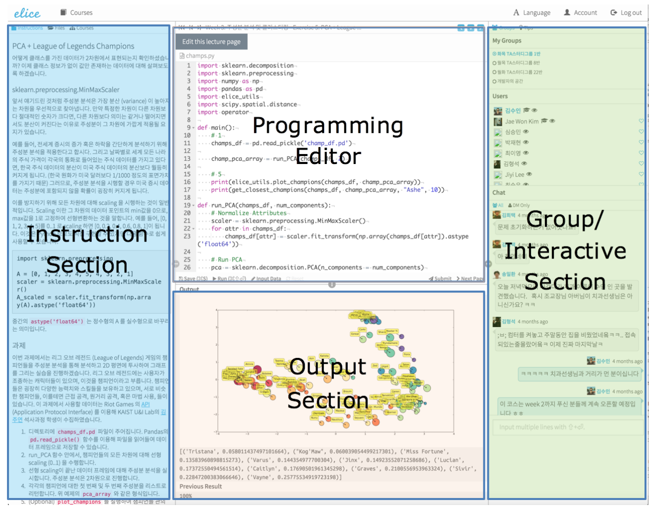


Figure 2: Part of the plagiarism network detected from KAIST Fall 2015 data structures course.

We demonstrate Elice, an online CS education platform with the goal to (i) identify the students who need assistance and (ii) reduce the teaching assistant (TA)'s time from redundant tasks. This way, we can scale CS education using our online education platform.

Student's Progress Using the Educational Taxonomy
From the educational taxonomy, we re-define the six steps of learning stages of solving a programming exercise in Elice (Figure 1; recognize, design & model, implement, test & debug, refactor, present). For each students intermediate code submission, Elivate can identify students current stage of learning and report to TA. With those reports, TAs can summarize each students individual characteristics. Using each student behavior in the initial two weeks, we can predict the student's final outcome at the end of 6 weeks (r=0.91).

Estimated Time to Completion
With more than 240,000 code submissions, we clustered student's intermediate revision snapshots to generate the state transition network for each programming exercise, so that every code revision is assigned to one of the states. The state starts with each student in the "skeleton" state (the initial code skeleton given to students), and they progress probabilistically between the states, eventually reaching the final state of "100%". As students submit more code revisions, the state network is automatically updated. Using this network, we can effectively predict each student's time to complete the exercise. Using Elivate, TAs can quickly identify students who need assistance, even before they proactively ask for help.



```
import sklearn.decomposition
import sklearn.preprocessing
import numpy as np
import pandas as pd
import elice_utils
import scipy.spatial.distance
import operator

def main():
    # 1
    champs_df = pd.read_pickle('champ_df.pd')

    champ_pca_array = run_PCA(champs_df, 2)

    # 5
    print(elice_utils.plot_champions(champs_df, champ_pca_array))
    print(get_closest_champions(champs_df, champ_pca_array, "Ashe", 10))

def run_PCA(champs_df, num_components):
    # Normalize Attributes
    scaler = sklearn.preprocessing.MinMaxScaler()
    for attr in champs_df:
        champs_df[attr] = scaler.fit_transform(np.array(champs_df[attr]).astype
('float64'))

    # Run PCA
    pca = sklearn.decomposition.PCA(n_components = num_components)
```

Figure 3: Elice Web-based programming platform. Current screen shows the exercise and result of the PCA algorithm.

Automated Grading and Plagiarism Detection
Surveys show that one of the most time-consuming tasks for TAs is grading. To save them time from such redundant and repeated task, we designed Elice to accept predefined test cases that run on the given skeleton code to allow automatic grading. In addition, we integrate a plagiarism detection module, thus TAs get notification of potential copying behavior of source code that may require further inspection.

Acknowledgements
This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. B0101-15-0307, Basic Software Research in Human-level Lifelong Machine Learning (Machine Learning Center))