# Student Performance Prediction by Discovering Inter-Activity Relations

Shaghayegh Sahebi
Department of Computer Science
University at Albany – SUNY
Albany, NY
ssahebi@albany.edu

Peter Brusilovsky
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA
peterb@pitt.edu

## ABSTRACT

Performance prediction has emerged as one of the most popular approaches to leverage large volume of online learning data. In the majority of current works, performance prediction is based on students' past activities in graded learning resources (such as problems and quizzes), while their activities in non-graded resources (such as reading material) are ignored. In this paper, we introduce an approach that can take advantage of students' work with non-graded learning resources, as *auxiliary* data, in order to predict students' performance in graded resources. This approach can discover the hidden inter-relationships between learning resources of different types, only using student activity data. Based on our experiments, the proposed approach can significantly reduce the error of student performance prediction, compared to baseline algorithms, while discovering meaningful and surprising relationships among learning resources.

## Keywords

student modeling, learning material correlation discovery

## 1. INTRODUCTION AND RELATED WORK

The learning data abundance, due to popularity of Massive Open Online Courses (MOOCs), introduces new opportunities and challenges for the educational data mining (EDM) field. On one hand, larger volumes of student data can increase performance of traditional EDM approaches. For example, a performance prediction approach that is popular in the area of intelligent tutoring systems, offers a good basis for learning personalization. If the data-driven performance model predicts that some problem will be solved by the current student with a high probability, this problem could be skipped in favor of a more challenging one. If the expected performance is low, students could be offered some help and supplementary material. MOOC-scale data can help improving performance prediction making this approach more usable. On the other hand, data coming from modern MOOCs is usually more heterogeneous and too complicated for traditional EDM approaches. Unlike conventional Intelligent Tutoring Systems (ITS), that are mostly based on problem-solving, MOOCs offer students to learn and assess their knowledge using a variety of learning resources, such as reading materials, lecture videos, assignments, exams, graded quizzes, and discussions. This leads to various types of learning activities for students. With that heterogeneity, come interesting challenges: how to use information about student work with diverse learning resources to assess student knowledge or predict student performance? what is the relationship between concepts that are offered in different learning resource types?

A number of research projects, focused on alternative learning resources, demonstrated that many kinds of resources could considerably contribute to student learning. For example, Najar et al. studied effect of adaptive worked examples versus unsupported problem solving and showed that adaptive worked examples can lead to faster and more effective learning [Najar et al. 2014]. Also, Agrawal et al. showed that enriching textbooks with additional forms of content, such as images and videos, increases the helpfulness of learning material [Agrawal et al. 2014]. This indicates that ignoring the interaction between various types of resources limits our understanding of students' learning behavior and the efficiency of mining and analytical tasks, such as student knowledge modeling or performance prediction. Additionally, understanding inter-relationships between different resource types and student activities can help instructors in having more well-informed decisions on their course design. Modeling such inter-relationships in students' data can provide a unified view to data heterogeneity and present a better understanding of student learning, by modeling these different resource types that present student activities.

While there are some studies in the literature on impact of various learning resources on learning, the relationship between learning resource types and their effect on predicting student performance is under-investigated. For example, Wen and Rosé studied student patterns across different activity types and concluded that these patterns can provide insights into different activity distributions between high-grade and low-grade students [Wen and Rosé 2014]. However, their goal was not to predict student grades from their activities. Velasquez et al. [Velasquez et al. 2014] identified learning aid use patterns using cluster analysis. They showed that high use of learning aids is significantly correlated with students' exam performance. But, they did not

predict student performance. Sao Pedro et al. [Sao Pedro et al. 2013] extended Bayesian Knowledge Tracing by conditioning the learning on whether the students received scaffolding in a topic or not. This model uses extra context information (topics) in addition to student performance, does not discover the relationship between learning resource types, and does not distinguish between different learning resources. Jiří and Pelánek studied learning resource similarities [Jiří and Pelánek 2017], but it was on graded resources, not considering resource types, and not predicting student performance.

One reason for unpopularity of using heterogeneous resources for predicting student performance is their potential conflicting effects. For example, Beck et al. investigated if providing assistance (help) to students benefits them using experimental trials, Bayesian Evaluation and Assessment framework, and learning decomposition [Beck et al. 2008]. In their studies, experimental trials and learning decomposition showed that assistance hurts students' learning. However, the Bayesian Evaluation and Assessment framework found that assistance promoted students' long-term learning. More recently, Huang et al. discovered that adaptation of their framework (FAST) for student modeling by including various activity types may lead researchers to contradictory conclusions [Huang et al. 2015]. More specifically, they studied the impact of example usage on student learning. In one of their formulations student example activity suggests a positive association with model parameters, such as probability of learning, while in another formulation this type of activity has a negative association with model parameters. Also, Hosseini et al. concluded that annotated examples show a negative relationship with students' learning, because of a selection effect: while annotated students may help students to learn, weaker students may study more annotated examples [Hosseini et al. 2016].

Another complication for considering heterogeneous resources is the difficulty in interpreting students' observed activities. In graded resource types, such as assignments and quizzes, a student's score explicitly represents her knowledge on the topic. Whereas in other resource types, such as reading material, there is no direct evaluation or explicit observation of student's knowledge. Hence, measuring the effect of such learning resources on students' knowledge, and thus predicting their future performance, would be a challenging task.

In this paper we propose an approach motivated by canonical correlation analysis (CCA) to discover the interaction between different learning resource types, using student activities, and to predict student performance on different learning resources. Our proposed approach can uncover latent relationships among subsets of learning recourses from different types and can quantify these relationships. Our analysis on two real-world datasets demonstrates that the discovered relationships are meaningful and can be used for course design and adaptive learning purposes. Additionally, the proposed approach can use student interactions with one *auxiliary* learning resource (such as examples) to predict students performance on another *target* learning resource type (such as problems). Our experiments on four real-world datasets show that our approach can efficiently use the extra information provided by auxiliary learning resources and significantly improve the student performance prediction error over the baseline models.

## 2. THE APPROACH

Our proposed approach is inspired by Canonical Correlation Analysis (CCA) [Hotelling 1936], which is a multivariate statistical model that studies the interrelationships among sets of multiple dependent and independent variables. CCA's goal is to find linear projections of these variable sets into a shared latent space such that the correlation between these projections are maximized. In this research, we use CCA as our main tool: we propose to find the relationship between students' ungraded activities (as independent variables) and students' graded activities (as dependent variables) using CCA. Our final goal is to propose a model for predicting student performance using pairs of resource types, motivated by the discovered relationships.

Our reason for choosing CCA as inspiration is twofold. First, CCA provides different views to the same data samples. Since we have the same students interacting with multiple resource types (e.g., examples and problems), we need to have a tool to model these interactions at the same time, while distinguishing between distinct resource types (as different views). Other factor analysis models, such as Principal Component Analysis (PCA), operate on one single view of the data and are not appropriate for our problem. Second, because of having multiple learning resources within each resource type (e.g., multiple problems and multiple examples) and several students (as datapoints) we need a multi-variate statistical model to capture the two-dimensional variability in the data. Bivariate or simpler multivariate models such as correlation or regression analysis can only capture the data variance for one dependent variable at a time and thus miss the variability of either students or learning material. We first give a brief background on CCA and then explain how to model and solve our problems using it.

***CCA.*** If matrix $X_{m\times n}$ represents $n$ data samples and $m$ variables and matrix $Y_{p\times n}$ contains the values for $p$ variables of same $n$ data samples, CCA aims to find linear transformations, $w_x$ and $w_y$, such that the correlation between projections of $X$ and $Y$ through $w_x$ and $w_y$ (reflected as $\rho$ in Equation 1) is maximized.

$$\rho = \frac{w_x^T X Y^T w_y}{\sqrt{(w_x^T X X^T w_x)(w_y^T Y Y^T w_y)}} \tag{1}$$

Since multiplication of $w_x$ and $w_y$ by a constant does not change the value of $\rho$ in Equation 1, the problem of finding $w_x$ and $w_y$ can be formulated as in Equation 2.

$$\begin{aligned} &\max_{w_x, w_y} w_x^T X Y^T w_y \\ &\text{subject to } w_x^T X X^T w_x = 1, w_y^T Y Y^T w_y = 1 \end{aligned} \tag{2}$$

Adding the regularization parameters to Equation 2, for controlling over-fitting of $\rho$, Sun et al. show that this regularized-CCA problem can be represented as in Equation 3, and solved using a least squares approach [Sun et al. 2008]. The formulation for $w_y$ is a symmetrical version of Equation 3.

$$X Y^T (Y Y^T)^{-1} Y X^T w_x = \eta (X X^T + \lambda I) w_x \tag{3}$$

In addition to $w_x$ and $w_y$ that produce the maximum correlation $\rho$, there can be other projection vector pairs that can

map $X$ and $Y$ matrices with correlations less than or equal to $\rho$. The optimization problem in Equation 4 finds these multiple projection vectors for $X$ (in matrix $W_x$).

$$\max_{W_x} \text{Trace}(W_x^T XY^T (YY^T)^{-1} YX^T W_x) \quad \text{subject to } W_x^T XX^T W_x = I \tag{4}$$

## 2.1 Relation Discovery between Learning Resource Types

As students work with various learning resources that are provided in an online course or a tutoring system, they gain more knowledge about the concepts presented in the course and can tackle more complicated problems. Knowing the relationship between different learning resource types and the way they interact in affecting students' knowledge can help better course design.Having the learning material from one resource type (e.g., problems) as one set of variables and learning material from another type (e.g., examples) as the other set of variables, we can interpret canonical correlation as a measure of relatedness between resource types.

More specifically, to map our problem to the CCA setting, we suppose that there are $n$ students that have at least one activity in each of the resource types. For example, these students may have tried some problems and studied some examples in the course. We represent the students' performance on problems as a matrix $Y_{p\times n}$, with $n$ students, $p$ problems, and $Y_{i,j}$ representing the student $j$'s score in quiz $i$. This score can be a grade or pass/fail indicator. Similarly, students' example activities can be represented as another matrix $X_{m\times n}$, with $n$ students, $m$ examples, and $X_{i,j}$ as an indication that user $j$ has read example $i$. Given these two activity matrices, we use CCA to find linear transformations $W_x$ and $W_y$ and canonical correlations $P$ as in Equation 4.

Formulating our problem as an instance of CCA, $W_x$ and $W_y$ can represent linear transformation matrices that map the original activity matrices $X$ and $Y$ into a shared latent space. These projections are scaled based on the canonical correlation values in a diagonal matrix $P_{c\times c}$, in which each of the diagonal elements are equivalent to the canonical correlation value $\rho_i$ for each projection vector pair $W_{x_{:,i}}$ and $W_{y_{:,i}}$. Meanwhile, the projection matrices $W_{x_{m\times c}}$ and $W_{y_{p\times c}}$ are representations of learning resources, projected into the shared space. Having this shared component space, we can compare and relate activities that are present in the two resource types.

In other words, each learning material $i$ from the auxiliary learning resource in matrix $X$, will be represented as a $1\times c$ vector $W_{x_{i,:}}$ and each learning material $j$ from the target learning resource in matrix $Y$, will be represented as a $1\times c$ vector $W_{y_{j,:}}$. So, we can find the most similar resources from different types by looking at the cosine similarity between those vectors in the shared component space.

Note that this is different from simply comparing matrices $X$ and $Y$ in the shared *student* space by calculating their cosine similarity. Here, we have the canonical correlation effect on finding similar learning resources. To be more clear, if we suppose that $w_x^T XX^T w_x = 1$ and $w_y^T YY^T w_y = 1$ (by which we transformed Equation 1 to Equation 2), then we have:

$$\hat{\rho} = w_x^T XY^T w_y \tag{5}$$

$\hat{\rho}$ in Equation 5 is equivalent to $\rho$ in Equation 1, scaled by its denominator. Now, if we left-multiply both sides of Equation 5 by ${w_x^T}^{-1}$, and right-multiply both sides of it by ${w_y}^{-1}$, we achieve $XY^T = {w_x^T}^{-1}\rho {w_y}^{-1}$. Equivalently, when having multiple canonical correlations, we can see that:

$$XY^T = {W_x^T}^{-1} P {W_y}^{-1} \tag{6}$$

Equation 6 shows the relationship between the projection matrices with the cosine similarity of $X$ and $Y$ ($XY^T$). Clearly, $YX^T$ and $W_y W_x^T$ are not equal.

## 2.2 Inter-Activity Performance Prediction

Predicting how a student performs on a problem can help teachers to adjust the course material based on students' predicted performance and can lead to personalized learning. Also, it can guide students towards a structured and effective learning. As in many prediction problems, educational data is usually incomplete: not all students try all resources. We focus on predicting students' scores for the first time that they try a problem. Thus, the problem of predicting students' performance can be interpreted as estimating the missing values in the student activity matrix ($Y$) that is described in the beginning of Section 2.

As proposed in Section 2.1, we can find the relationship between sets of learning resources of two types using CCA. Thus, if we know students' performance on auxiliary learning resources in matrix $X$ and their performance in the target learning resource in matrix $Y$, we can understand how students' activities on auxiliary learning resources affect the same students' performance on the target learning resources. When the student activity matrix ($Y$) is incomplete, we can estimate $w_x$ and $w_y$ by calculating the canonical correlations between the auxiliary activity matrix $X$ and the incomplete target activity matrix $Y$ to achieve the estimated projection vectors $\hat{w_x}$ and $\hat{w_y}$. Using these projection vectors, we can estimate a complete activity matrix $\hat{Y}$ as in Equation 7.

$$\hat{Y} = \hat{w_y} \rho {\hat{w_x}}^T X \tag{7}$$

Here, student activities in the auxiliary learning resource are mapped to the shared latent space, scaled by the canonical correlation factor $\rho$, and then mapped back to the target learning resource space. In case of calculating multiple ($c$) projection vector pairs ($\hat{W}_{x_{m\times c}}$ and $\hat{W}_{y_{p\times c}}$), with canonical correlations represented in $P_{c\times c}$, we estimate students' performance ($\hat{Y}$) as in Equation 8.

$$\hat{Y} = \hat{W}_y P \hat{W}_x^T X \tag{8}$$

# 3. DATASETS

We use four datasets from two online platforms for our experiments. The anonymized data represent log files of student interaction with course resources (activities), and their performance in them. Each of these platforms allow their students to learn from multiple learning resource types that calls for modeling inter-activity relations. The first two datasets are richer since they have learning resource names, topics, and contents although we do not use them for the discovery and prediction purposes. The third and fourth datasets are larger, from a MOOC platform, with more variation in learning resource types. However, we do not have

access to these learning resources beyond their assigned IDs. In the following sections, we describe each of these datasets.

**Table 1: Statistics of Mastery Grids datasets**

| | | students | prob. | Parsons prob. | annot. exam. | anim. exam. |
|---|---|---|---|---|---|---|
| Python | number | 319 | 37 | 43 | 58 | 53 |
| | average activity records | 65.5 | 147.5 | 112.3 | 97.2 | 93.8 |
| | density | 0.34 | 0.46 | 0.35 | 0.30 | 0.29 |
| Java | number | 206 | 113 | - | 101 | 50 |
| | average activity records | 127.2 | 108.3 | - | 93.9 | 89.7 |
| | density | 0.78 | 0.53 | - | 0.47 | 0.44 |

**Table 2: Statistics of Canvas Network datasets**

| | | students | quiz-assign. | assign. | discus. topics |
|---|---|---|---|---|---|
| Business and Management | number | 232 | 32 | 38 | 34 |
| | average activity records | 62.7 | 208.1 | 190.8 | 18.9 |
| | density | 0.60 | 0.89 | 0.82 | 0.08 |
| Professions and Applied Sciences | number | 1160 | 18 | 26 | 70 |
| | average activity records | 16.25 | 427.3 | 372.5 | 21.1 |
| | density | 0.14 | 0.37 | 0.32 | 0.02 |

## 3.1 Mastery Grids Datasets

Our first two datasets are collected from an online intelligent tutoring system, Mastery Grids [Loboda et al. 2014]. This system provides personalized access to three types of interactive content for Java programming and four types of content for Python programming. Parameterized semantic problems, annotated examples (code snippets with explanations), and animated examples (interactive simulations that visually demonstrate the runtime behavior of a code snippet) are the three types of resources that are available for both Java and Python courses. In addition to those, Python course includes the so-called *Parsons problems* originally introduced in [Parsons and Haden 2006].

The parameterized semantic problems (problems, for short) are generated by QuizJet and QuizPet system [Hsiao et al. 2009] from a pool of parameterized questions on Java and Python programming. As a result, the same problem can be attempted multiple times by students with various parameters. We only consider students' first attempt on each problem for our experiments. Annotated examples presented by WebEx allow students to interactively explore line-by-line explanation of code snippets [Brusilovsky and Yudelson 2008]. Working with animated examples, which are generated using Jsvee library [Sirkiä 2016], students can execute a Java or Python program visually, observing internal operation, such as variable assignments and printing on a console. In Parsons problems, students are asked to solve a programming task by selecting and sorting provided code lines.

Mastery Grids groups different learning resources into multiple learning topics. Although this system offers a recommended topic sequence in its interface, the students are free to select and work on any of the topics and learning resources at any given time. The Java dataset from this system is collected from Fall and Spring semesters of 2016. Among all of the students, we selected the ones who have at least one activity in each of the problems, annotated examples, and animated examples. A summary of statistics for these datasets are shown in Table 1. The Python dataset about two times sparser than the Java dataset in terms of number of all activities per student. Among different resource types, the density of student activities on problems are the closest between the two datasets. In both of the datasets, student activities on problems are the densest and activities on animated examples are the most sparse.

## 3.2 Canvas Network Datasets

Our third and fourth datasets are publicly available from Canvas Network (`http://canvas.net`) [Network 2016]. Canvas Network hosts many freely available open online courses in which it offers multiple leaning resource types. More specifically, in addition to learning modules, each course can have different types of assignments, discussions, and pop-quizzes. Participants are not limited to a specific sequence of learning material or assignments. Categories of the learning resources include "assignments", "quiz-assignments", "pop-quizzes", "discussions", and "wikis". The dataset is anonymized such that student IDs, course names, discussion contents, submission contents, and course contents are not available.

Course assignments can be quiz-style ("quiz-assignment") or in longer format, for which students submit a text or video file ("assignments"). We choose two of the offered courses in Canvas Network as the third and fourth datasets for our experiments. These two courses are selected because they provide multiple learning resource types and have more active students in all of these resource types. The first course is in the "Professions and Applied Sciences" field and the second course is in the "Business and Management" field.

Since assignments, quiz-assignments, and discussions have the most activities, we focus on these resource types in our experiments. Among these three, assignments and quiz-assignments are graded. For consistency, we normalize students' grades between zero and one based on their maximum possible grade. For discussions, we consider a binary variable representing if a student has posted a message or not. We select the students who have at least one activity in each of these learning resources. A summary of statistics for these datasets is shown in Table 2. Discussion topics have the least dense activity matrices in the two datasets. They are very sparse compared to student activities on assignments and quiz-assignments. Comparing the two datasets from Canvas Network, overall student activities in professional and applied sciences domain course is very sparse. But, the density of student activities on all resources in business and management domain course is comparable with the datasets from Mastery Grids system. However, the distribution of student activities among various resource types are more skewed in the Canvas Network datasets.

# 4. EXPERIMENTS

## 4.1 Experiment Setup

Per the proposed model in Section 2, element $X_{i,j}$ in activity matrix $X$ represents the result of student $j$'s first attempt

on learning resource $i$. This activity result can be different for different learning resource types. For graded learning resources, such as assignments and quiz-assignments, we use the normalized score of students; for problems and Parsons problems with success or failure feedback, we use binary scores; and for non-graded activities, such as reading an annotated example or posting in a discussion forum, we use a binary indicator that shows the students' attempt. We use average imputation for missing values.

For prediction experiments, we follow a 5-fold user stratified separation of the student performance data to perform cross-validation on it. Particularly, in each round of experiments, we select 20% of students as test students, 15% of them for validation purposes, and 65% of them as train. Our task is to predict test students' performance on activities in a target learning resource type, observing 20% of these students' activities, and the training data. In the CCA-based proposed approach, the training data includes all students' activities in the auxiliary learning resource type, in addition to observed activities of students in the target resources. We repeat each round of the experiments for 5 times.

Since only quiz-assignments and assignments are graded in the Canvas Network datasets, and only problems and Parsons problems are graded in the Mastery Grids datasets, we define the prediction tasks on these resource types. Discussions from the Canvas Network datasets and examples (annotated and animated) from the Mastery Grids datasets are only used as auxiliary resources. Note that each of graded resource types (quiz-assignments, assignments, problems, and Parsons problems) can also be used as an auxiliary resource for another type of graded resource in the same dataset.

**Baselines.** In previous works, collaborative filtering methods have been proved successful in predicting students performance [Thai-Nghe et al. 2011, Sahebi et al. 2014]. Since our proposed approach is similar to these approaches in discovering latent relationships among learning resources, through factorizing activity matrices, we choose two settings of SVD++ algorithm [Koren et al. 2009] as our baselines. To study if adding student activities in auxiliary resource type would help better estimation of students performance in the target resource type, we compare our approach with single-resource SVD++ algorithm. In this setting we run SVD++ algorithm only on the target learning resource matrix, assuming that we do not have the information on student activities in the auxiliary resource types, and compare the results with our proposed method. To understand our CCA-based method's efficiency on capturing important relationships between different learning resource types, we compare it with a paired-resource setting of SVD++ algorithm. Particularly, we merge the two auxiliary and target learning resource types into one set of learning materials (represented by one matrix) and run the SVD++ algorithm on this augmented matrix. Note that our proposed method factorized two separate matrices at the same time but SVD++ can only factorize one matrix.

Since the student activity datasets are biased towards student success (e.g., average grade for problems in the Python dataset is 0.67 out of 1), we compare the methods with an average baseline. To do this, we use the training dataset average as the predicted performance for all of the students in each of the 5 data splits.

## 4.2 Discovering Relationships between Learning Resource Types

One of our goals in this paper is to understand relationships and interactions between sets of learning resources with various types. CCA has the ability to represent each pair of learning resource types in the same latent space. This enables us to relate learning material of different types only based on student activities, without relying on their content or presented concepts. Since the Mastery Grid datasets provide learning resource names and topics we can confirm the discovered relationships by comparing them with learning resource topic similarities. These topics have been manually assigned to learning resources by experts, during course design in Mastery Grids. In order to take a deeper look at the discovered similarities, we study the top similar learning resources of different types in the same course (as shown in Table 3). To calculate these similarities, we look at projections of each learning resource in the shared latent space, $W_x$ and $W_y$ and calculate the cosine similarity between them, as mentioned in Section 2.1. We look at the most similar learning resources of each course in the following.

**The Java Dataset.** For the Java dataset, we can calculate the cosine similarity of problems with animated examples and problems with annotated examples. We can see the most similar problems and animated examples in rows 1-4 of Table 3. As we can see, three of these four learning resource pairs are from the same expert-labeled topic. For example, both problem "jWhile1" and animated example "ae_while_demo" are about "while loops" in Java. This shows that our approach can accurately figure out the most similar problems and animated examples, only based on student activities and their performance, not knowing about their topic or content. However, the resources in row 3 are from different expert-labeled topics "boolean expressions" and "switch". While these two are not exactly the same, the switch expressions in Java use boolean expressions in their conditional statements. So these two topics are closely related to each other: if a student cannot understand the "boolean expressions" topic, understanding the "switch" topic would be difficult for this student.

The most similar Java annotated examples and problems, found by CCA projection matrices, are listed in rows 5-8. Here, we do not see the obvious similarities that was apparent between animated examples and problems. In row 5, there is topic similarity between the problem with "loops do-while" topic and the annotated example with "loops for" topic: both of them are about loops in Java. For row 8, we know that Java for loops use "arithmetic operations" in their conditional statement. However, topics for similar resources discovered in rows 6 and 7 look irrelevant. Row 6's problem is labeled by experts with the "interfaces" topic, while the similar annotated example is labeled with the "variables" topic. Likewise, the problem topic in row 7 is "interfaces", while the topic of similar annotated example is "objects".

To gain more insight about these learning resources, we looked at their contents. We discovered that although the general topics for these problems and their discovered anno-

**Table 3: Most similar learning materials of different types, from Java and Python courses, according to their similarity using CCA projection vectors.**

| *course* | *material type* | *row* | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *Java* | *prob. & anim. exam.* | | *prob. ID* | *prob. name* | *prob. topic* | *anim. exam. topic* | *anim. exam. name* | *anim. exam. ID* |
| | | 1 | 14 | jArrayList5 | ArrayList | ArrayList | ae_arraylist2_v2 | 3 |
| | | 2 | 18 | jBoolean_Operators | Boolean expressions | Switch | ae_switch_demo2 | 44 |
| | | 3 | 65 | jMathFuc2 | Arithmetic operations | Arithmetic operations | ae_arithmetic_v2 | 1 |
| | | 4 | 100 | jWhile1 | Loops while | Loops while | ae_while_demo | 49 |
| | *prob. & annot. exam.* | | *prob. ID* | *prob. name* | *prob. topic* | *annot. exam. topic* | *annot. exam. name* | *annot. exam. ID* |
| | | 5 | 37 | jDowhile1 | Loops do_while | Loops for | for1_v2 | 28 |
| | | 6 | 57 | jInterfaces1 | Interfaces | Variables | PrintTester | 78 |
| | | 7 | 61 | jInterfaces5 | Interfaces | Objects | AccessorMutatorDemo | 1 |
| | | 8 | 63 | jMathCeil | Arithmetic operations | Loops for | JavaTutorial_4_6_8 | 57 |
| *Python* | *prob. & annot. exam.* | | *prob. ID* | *prob. name* | *prob. topic* | *annot. exam. topic* | *annot. exam. name* | *annot. exam. ID* |
| | | 9 | 3 | q_py_arithmetic1 | Variables | Variables | pyt1.3 | 5 |
| | | 10 | 21 | q_py_nested_if_elif1 | if_statements | values_references | pytt10.25 | 58 |
| | | 11 | 23 | q_py_obj_account1 | classes_objects | Lists | pyt7.2 | 53 |
| | *prob. & anim. exam.* | | *prob. ID* | *prob. name* | *prob. topic* | *anim. exam. topic* | *anim. exam. name* | *anim. exam. ID* |
| | | 12 | 7 | q_py_dict_access1 | dictionary | loops | ae_adl_while | 39 |
| | | 13 | 29 | q_py_output1 | output_formatting | variables | ae_adl_arithmetics2 | 1 |
| | | 14 | 10 | q_py_fun_car1 | functions | exceptions | ae_adl_tryexcept2 | 34 |
| | *prob. & pars. prob.* | | *prob. ID* | *prob. name* | *prob. topic* | *pars. prob. topic* | *pars. prob. name* | *pars. prob. ID* |
| | | 15 | 10 | q_py_fun_car1 | functions | exceptions | ps_python_try_adding | 38 |
| | | 16 | 12 | q_py_if_elif1 | if_statements | loops | combo_python_while | 9 |
| | | 17 | 35 | q_py_swap1 | variables | variables | combo_swap | 11 |
| | *pars. prob. & annot. exam.* | | *pars. prob. ID* | *pars. prob. name* | *pars. prob. topic* | *annot. exam. topic* | *annot. exam. name* | *annot. exam. ID* |
| | | 18 | 1 | combo_avg | variables | variables | pyt2.1 | 32 |
| | | 19 | 14 | ps_python_addition | variables | variables | pyt1.2 | 4 |
| | | 20 | 41 | ps_return_bigger_or_none | functions | functions | pyt10.7 | 30 |
| | *pars. prob. & anim. exam.* | | *pars. prob. ID* | *pars. prob. name* | *pars. prob. topic* | *anim. exam. topic* | *anim. exam. name* | *anim. exam. ID* |
| | | 21 | 1 | combo_avg | variables | variables | ae_python_assignment | 40 |
| | | 22 | 12 | ps_hello | variables | variables | ae_adl_arithmetics2 | 1 |
| | | 23 | 43 | ps_simple_params | functions | functions | ae_adl_returnvalue | 29 |

```
public class Tester {
  public static void main(String[] args) {
                Mechanism mech1 = new Computer(2.0, 2.0, true);
                Mechanism mech2 = new Car("Honda", 2);

                Computer comp = (Computer) mech1;

                System.out.println(comp.getProcessorSpeed());
                System.out.println(comp.reportProblems());

                System.out.println(((Car) mech2).getBrand());
                System.out.println(mech2.reportProblems());
  }
}

What is the output?
Be careful of the whitespace(space, newline) in your answer.
```

**Figure 1: Content of problem with "Interfaces" topic (row 6 of Table 3)**

tated examples are not the same, they include very similar concepts. For example, Figure 1 shows the content for problem "jInterfaces1" (topic: "interfaces"), and Figure 2 shows the content for annotated example "PrintTester" (topic: "variables"). As we can see, the concept of printing an output in the console is very important in both of these learning resources. Interestingly, it appears that although the designers of Java course were interested in the mentioned topics while designing these learning resources, we are discovering other possible "latent topics" for them. Another factor in these newly-found relations can be the mixed relationship of annotated examples with students performance. Hosseini et al. have studied the use and impact of annotated and animated examples in the same online tutoring system and concluded that students are likely to learn more from animated examples [Hosseini et al. 2016]. Particularly, they showed that although more views of animated examples is associated with a higher course grade, the number of views on annotated examples has a negative effect on it. A possible reason is the negative process of associating examples with poor knowledge: students with poor knowledge are more likely to study annotated examples. This association can potentially overcome the positive impact of learning from annotated examples and lead to a negative impact. Also, they show that animated examples provided better impact on problem solving success and post-test scores.

**The Python Dataset.** We study 5 pairs of resource types and the cosine similarities between $W_y$s and $W_x^T$s in the Python dataset: problems vs. animated examples, problems vs. annotated examples, Parsons problems vs. animated

```
public class PrintTester
{
   public static void main(String[] args)
   {
      System.out.println(3 + 4);

      System.out.println("Hello");
      System.out.println("World!");

      System.out.print("00");
      System.out.println(3 + 4);

      System.out.println("Goodbye");
   }
}
```

**Figure 2: Content of annotated example with "Variables" topic (row 6 of Table 3)**

```
class Account:
    def __init__(self, deposit=0):
        self.balance = deposit

    def deposit(self, sum):
        self.balance += sum

    def withdraw(self, sum):
        self.balance -= sum
    def get_balance(self):
        return self.balance

def main():
    accounts = {}
    accounts[0] = Account()
    accounts[1] = Account(379)

    accounts[0].deposit(379)
    accounts[1].deposit(379)
    accounts[0].withdraw(379-50)
    accounts[1].withdraw(379-100)
    print(accounts[0].get_balance() + accounts[1].get_balance())
main()

What is the output?
Be careful of the whitespace(space,newline) in your answer.
```

**Figure 3: Content of problem with "classes_objects" topic (row 11 of Table 3)**

examples, Parsons problems vs. annotated examples, and problems vs. Parsons problems. Samples of discovered similar learning resources are shown in Table 3.

As shown in rows 9-11, the first problem and its matched annotated example have the same topic of "variables". But, the next two pairs do not have a common topic. We study the content of these learning resources to understand the nature of their similarity. For example, if we look at row 11, we see that annotated example "pyt7.2" has topic of "lists". Now if we look at problem "q_py_obj_account1" with topic of "classes_objects" in Figure 3, we can see that this problem uses lists (accounts variable) in it. We avoid showing the content for the pair in row 10 due to space limits.

Rows 12-14 show similar animated examples and problems in the Python dataset. To show the similarities between concepts used in these animated examples and problems, we look at one pair: problem "q_py_fun_car1" with topic "functions" (Figure 4) and animated example "ae_adl_tryexcept2" with topic "exceptions" (Figure 5). We can see that there is a function call and a function definition in this animated example (Figure 5). Consequently, although this animated example is not designed to teach the "function" topic and despite of it being labeled with the "exceptions" topic only, the discovered similarities show the associations between students' learning of functions and this animated example.

The most similar problems and Parsons problems are shown in rows 15-17 of Table 3. Two of the top similar pairs are from the same ("variables") or related ("if statements" and "loops") topics. The resources in row 15 are from different topics: a "functions" problem and an "exceptions" Parsons problem. But, as can be seen in Figures 4 and 6 the Parsons problem includes a function definition. So, students can learn about functions while executing this animated example that is about exceptions.

```
def fuel(gallons, gas, tank_size):
    gas = min(gallons + gas, tank_size)
    return gas
gas = 50-42
gallons = fuel(25, gas, 50)
print(gallons)

What is the output?
Be careful of the whitespace(space,newline) in your answer.
```

**Figure 4: Content of problem with "functions" topic (rows 14 and 15 of Table 3)**

```
1  def average(a, b):
2    sum = int(a) + int(b)
3    return sum / 2
4
5
6  def main():
7    try:
8      avg = average("1", "two")
9      print("Avg is:", avg)
10   except ValueError:
11     print("Error occurred!")
12
13
14 main()
```

**Figure 5: Content of animated example with "exceptions" topic (row 14 of Table 3)**

Drag from here

```
print("Can only add numbers together.")
except TypeError:
return a + b
def add_two_numbers(a,b):
try:
```

New instance Get feedback

Construct a function that adds two numbers together and handles non-numeric input.

**Figure 6: Content of Parsons problem with "Exceptions" topic (row 15 of Table 3)**

Finally, as we can see in rows 18-23, analogous samples of Parsons problems vs annotated examples, and Parsons problems vs animated examples are all from the same topics.

One may think that the discovered similarities are a result of topic arrangements in the course design and conclude that we can find these similar learning resources by only looking at the co-occurrence of student activities in two learning resource types, e.g., by calculating the cosine similarities between learning resources in the original student-space, or matrices $X$ and $Y$. However, looking at some of the discovered similarities, such as the second row of Table 3, reassures us that our approach can find the relationships beyond their trivial co-occurrence. As we have mentioned, the "switch" and "boolean Expressions" topics are not the same, but are very related. In the Mastery Grids interface, these two topics are not placed right next to each other. But, another topic ("if-else" topic) is placed between them. This means that the discovered similarity is not solely based on activity co-occurrence due to topic placement in Mastery Grids.

To discover what we can gain from trivial co-occurrences, without using our proposed method, we looked at samples of the most similar learning resources, based on the cosine similarity between student activities in the original student space (similarity between matrices $X$ and $Y$). In this case, the most similar discovered learning resource pairs are either placed closely in the same topic (and thus, may happen due to the students following the sequence imposed by learning resource arrangements in the interface), or do not have any meaningful content-based relationship. For example, the most similar animated example that is discovered in student space for the "jBoolean_Operators" problem (problem in row 2 of Table 3) is labeled with the "primitive data types" topic, demonstrating "Double" and "Short" data types.

To summarize, the discovered CCA-based similarities in both datasets are meaningful. Some of the related learning resource pairs are from the same topics, others are related in the concepts or sub-topics that they present. In general, this is a very promising result, especially for applications in which the learning resource contents are difficult to analyze and compare. Discovering these similarities, instructors can rearrange their learning material in ways that most benefits students' learning. Also, it can be used for multi-source knowledge modeling of students. Namely, we can model student knowledge in shared concepts between problems and animated examples and understand how a student's ability in a learning recourse type (e.g., to solve a problem) increases by trying another learning resource of a different type (e.g., a related animated example).

## 4.3 Predicting Student Performance Using Auxiliary Resource Types

Using the formulation proposed in Section 2.2, our goal here is to predict students' performance using auxiliary learning resource types and compare it with similar baseline approaches. We measure performance of the proposed and baseline approaches using Root Mean Squared Error (RMSE). This measure quantifies the average difference between actual students' score and their predicted performance.

**Mastery Grids Datasets** For the Java programming dataset, we run two sets of experiments. The first set of experiments is on predicting students performance on problems, using their activities on annotated examples as auxiliary data ("annotated examples → problems"). In the second set of experiments, we use animated example activities as the auxiliary resource for predicting students performance on problems ("animated examples → problems"). As mentioned before, we compare the results of our proposed approach with single-resource SVD++ –only using student logs on problems– and paired-resource SVD++ –with the same input as our proposed approach–.

For the Python programming dataset, we run six sets of experiments. Having problems and Parsons problems as target learning resource types, we use annotated examples and animated examples as the auxiliary learning resources. Additionally, problems may help us in predicting students' performance in Parsons problems, and vice versa.

Table 4 shows the RMSE of CCA-based and baseline approaches for these sets of experiments on both of Mastery Grids datasets. The numbers in parentheses report the 95-percentile confidence interval for the reported errors. As we can see here, our CCA-based approach performs significantly better than the baselines in all of the experiment setups in both datasets. As our proposed approach performs better than single-resource SVD++, we can conclude than adding the auxiliary data significantly improves student performance prediction. On the other hand, we can see that the proposed CCA-based approach works better than SVD++ in the multi-recourse setting using the same set of auxiliary and target data. Therefore, we can conclude that our approach is a better fit for effectively using auxiliary data.

Comparing the two settings for SVD++, in the Python dataset single-resource SVD++ performs as good as, or significantly better than paired-resource SVD++. Specifically, for combinations "animated examples → problems" and "annotated examples → problems", paired-resource SVD++ has a significantly higher error than single-resource SVD++. This confirms our findings in Section 4.2 about smaller similarities between problems and examples in the Python dataset. As expected in biased datasets, we can see that average baseline is working very well. Comparing with paired-resource SVD++, its error is significantly lower in four of the experiments on the Python dataset. Single-resource SVD++ is significantly better than (in "animated examples → problems", "annotated examples → problems", and "problems → Parsons problems") or similar to the average baseline.

In contrast, in the Java dataset, the average baseline has slightly, but significantly, higher error than the proposed approach and the other two baselines for "annotated examples → problems". For "animated examples → problems", the average baseline has better predictions compared to the other two baselines. Also, paired-resource SVD++ works significantly better than single-resource SVD++ for "annotated examples → problems". This shows that paired-resource SVD++ is not consistent on different datasets, even if similar learning resource types are used, and to be able to take advantage of auxiliary information, a more advanced approach, such as the proposed one, is needed.

**Canvas Network datasets.** Canvas Network datasets give us the opportunity to test our approach on more varied data of MOOCs and in different domains. Notably, "Professions and Applied Sciences" data has more users and is very

**Table 4: RMSE for student performance prediction task on Mastery Grids datasets.**

| | | anim. example → problem | annot. example → problem | pars. prob. → problem | anim. example → pars. prob. | annot. example → pars. prob. | prob. → pars. prob. |
|---|---|---|---|---|---|---|---|
| Java | paired-resource CCA | **0.4148 (0.0097)** | **0.4159 (0.0057)** | - | - | - | - |
| | paired-resource SVD++ | 0.5304 (0.0127) | 0.4696 (0.0047) | - | - | - | - |
| | single-resource SVD++ | 0.5178 (0.0214) | 0.4537 (0.0119) | - | - | - | - |
| | average baseline | 0.4859 (0.0071) | 0.4854 (0.0039) | - | - | - | - |
| Python | paired-resource CCA | **0.4584 (0.0035)** | **0.4566 (0.0024)** | **0.4579 (0.007)** | **0.4122 (0.0081)** | **0.4098 (0.0043)** | **0.4105 (0.0075)** |
| | paired-resource SVD++ | 0.516 (0.0124) | 0.5122 (0.0156) | 0.5524 (0.0083) | 0.5213 (0.022) | 0.456 (0.0084) | 0.4954 (0.0123) |
| | single-resource SVD++ | 0.4921 (0.0147) | 0.4921 (0.0147) | 0.4921 (0.0147) | 0.4409 (0.0059) | 0.4409 (0.0059) | 0.4409 (0.0059) |
| | average baseline | 0.4961 (0.0024) | 0.4972 (0.0036) | 0.4957 (0.0014) | 0.4724 (0.0056) | 0.4716 (0.0047) | 0.4723 (0.0072) |

**Table 5: RMSE for student performance prediction task on Canvas Network datasets, using discussions, quiz-assignments, and assignments as auxiliary resources.**

| | | quiz-assignments → assignments | discussions → assignments | assignments → quiz-assignments | discussions → quiz-assignments |
|---|---|---|---|---|---|
| Business and Management | paired-resource CCA-based | **0.1073 (0.0209)** | **0.1093 (0.0163)** | **0.0911 (0.0124)** | **0.1207 (0.0109)** |
| | paired-resource SVD++ | 0.1871 (0.0143) | 0.1569 (0.0115) | 0.1696 (0.0111) | 0.1903 (0.0085) |
| | single-resource SVD++ | 0.1890 (0.0208) | 0.1890 (0.0208) | 0.1532 (0.0125) | 0.1532 (0.0125) |
| | average baseline | 0.1741 (0.0182) | 0.1741 (0.0182) | 0.1752 (0.0118) | 0.1752 (0.0118) |
| Professions and Applied Sciences | paired-resource CCA-based | **0.1264 (0.0085)** | **0.1252 (0.0049)** | **0.1252 (0.0035)** | **0.1287 (0.0105)** |
| | paired-resource SVD++ | 0.2070 (0.0112) | 0.1897 (0.0140) | 0.2039 (0.0211) | 0.3254 (0.0171) |
| | single-resource SVD++ | 0.5235 (0.0196) | 0.5235 (0.01960) | 0.2057 (0.0176) | 0.2057 (0.0176) |
| | average baseline | 0.4596 (0.0019) | 0.4596 (0.0019) | 0.3838 (0.0037) | 0.3838 (0.0037) |

sparse compared to all other datasets. For Canvas Network datasets we run four sets of experiments. In the first two sets, we use quiz-assignments and discussions as auxiliary resources to predict students' performance in assignments. In the third and fourth sets of experiments we predict students' grade in quiz-assignments using general assignments and discussions as auxiliary resources.

Table 5 shows RMSE of all approaches on both "Professions and Applied Sciences" and "Business and Management" datasets. Similar to our results on the Mastery Grids dataset, we can see that the proposed approach can effectively use auxiliary resources to provide better estimation of student performance in all resource pairs. Comparing paired-resource SVD++ to single-resource SVD++, we can see that in most of the experiments their error is not significantly different. Only for "quiz-assignments → assignments" and "discussions → assignments", in "Professions and Applied Sciences" dataset, paired-resource SVD++ is significantly better than single-resource SVD++. Comparing the average baseline results, it's error is significantly higher than (in "Professions and Applied Sciences" dataset) or similar to paired-resource SVD++. Whereas compared to single-resource SVD++, it works better in predicting assignments, and worse in predicting quiz-assignments. This is because there is more variation in students' scores in quiz-assignments.

In addition to the way different courses are designed and learning resources are prepared, one of the reasons behind the different results between the two datasets can be due to the variations between two course datasets. For example, having more students and being sparser may lead to added value of auxiliary information in the "Professions and Applied Sciences" dataset (Table 2). In other words, effectiveness of adding auxiliary data for the task of performance prediction depends on the dataset and its characteristics.

# 5. CONCLUSIONS

We proposed an approach inspired by canonical correlation analysis for discovering interrelationships between learning resources of different types, only using student performance in them. This approach can also be used to predict students' performance. That is to say, we can predict students' performance in one type of learning resources, with the help of student activities in another resource type. We evaluated the proposed approach with four datasets and two tasks.

For the task of finding learning resource interrelationships, we evaluated our approach on the Java programming dataset with three resource types, and the Python programming dataset with four resource types. Finding the most similar resources of different types, only based on student activities, we showed that our approach is very promising in detecting these similarities, especially for learning resources that have been proved to have a positive effect on students' learning. Also, we found that our approach goes beyond the designated topics for learning resources and discovers latent similarities that provide clues of their content similarity.

Having four datasets from two online learning systems, we ended up with 16 total experiment sets for predicting student performance in paired resource types. We compared our proposed approach with an average baseline and two algorithmic baselines: one using student activities in both auxiliary and target resource types (paired resource SVD++), and one with using student activities in only target resource type (single resource SVD++). The experiments showed that our proposed approach can significantly improve estimation of student grades in all setups and datasets. This success is in part due to the extra information from the auxiliary resource types on students' performance: in three out of 16 setups, the baseline algorithm with auxiliary data performed better than the baseline algorithm without auxiliary data . However, in two of the setups the baseline with auxiliary data performed significantly worse than the baseline without it. Meanwhile, the proposed approach performed better than both baselines in all of the 16 experiments. It showed that better performance of the proposed approach is not only because of having extra information, but also because of its ability to use latent interrelationships between auxiliary and target resource types, in a more efficient way.

# 6. REFERENCES


[Agrawal et al. 2014] R. Agrawal, M. Christoforaki, S. Gollapudi, A. Kannan, K. Kenthapadi, and A. Swaminathan. 2014. Mining videos from the web for electronic textbooks. In *International Conference on Formal Concept Analysis*. 219–234.

[Beck et al. 2008] J. Beck, K. Chang, J. Mostow, and A. Corbett. 2008. Does help help? Introducing the Bayesian Evaluation and Assessment methodology. In *Intelligent Tutoring Systems*. 383–394.

[Brusilovsky and Yudelson 2008] P. Brusilovsky and M. Yudelson. 2008. From WebEx to NavEx: Interactive Access to Annotated Program Examples. *IEEE* 96 (2008), 990–999.

[Hosseini et al. 2016] R. Hosseini, T. Sirkiä, J. Guerra, P. Brusilovsky, and L. Malmi. 2016. Animated Examples As Practice Content in a Java Programming Course. In *Technical Symposium on Computing Science Education*. 540–545.

[Hotelling 1936] H. Hotelling. 1936. Relations Between Two Sets of Variates. *Biometrika* 28, 3/4 (1936).

[Hsiao et al. 2009] I. Hsiao, S. Sosnovsky, and P. Brusilovsky. 2009. Adaptive navigation support for parameterized questions in object-oriented programming. In *European Conference on Technology Enhanced Learning*. 88–98.

[Huang et al. 2015] Y. Huang, J. P. González-Brenes, and P. Brusilovsky. 2015. Challenges of Using Observational Data to Determine the Importance of Example Usage. In *International Conference on Artificial Intelligence in Education*. 633–637.

[Jiří and Pelánek 2017] R. Jiří and R. Pelánek. 2017. Measuring Similarity of Educational Items Using Data on Learners' Performance. In *Educational Data Mining*. 16–23.

[Koren et al. 2009] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[Loboda et al. 2014] T. D. Loboda, J. Guerra, R. Hosseini, and P. Brusilovsky. 2014. Mastery grids: An open source social educational progress visualization. In *European Conference on Technology Enhanced Learning*. 235–248.

[Najar et al. 2014] A. Najar, A. Mitrovic, and B. M. McLaren. 2014. Adaptive Support versus Alternating Worked Examples and Tutored Problems: Which Leads to Better Learning?. In *International Conference on User Modeling, Adaptation, and Personalization*. 171–182.

[Network 2016] Canvas Network. 2016. Canvas Network Courses, Activities, and Users (4/2014 - 9/2015) Restricted Dataset. (2016). `https://doi.org/10.7910/DVN/XB2TLU`

[Parsons and Haden 2006] D. Parsons and P. Haden. 2006. Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses. In *Australasian Conference on Computing Education*. 157–163.

[Sahebi et al. 2014] S. Sahebi, Y. Huang, and P. Brusilovsky. 2014. Parameterized Exercises in Java Programming: using Knowledge Structure for Performance Prediction. In *Workshop on AI-supported Education for Computer Science*. 61–70.

[Sao Pedro et al. 2013] M. Sao Pedro, R. Baker, and J. Gobert. 2013. Incorporating scaffolding and tutor context into bayesian knowledge tracing to predict inquiry skill acquisition. In *Educational Data Mining*.

[Sirkiä 2016] T. Sirkiä. 2016. Jsvee Kelmu: Creating and Tailoring Program Animations for Computing Education. In *Working Conference on Software Visualization*. 36–45.

[Sun et al. 2008] L. Sun, S. Ji, and J. Ye. 2008. A least squares formulation for canonical correlation analysis. In *International Conference on Machine Learning*. 1024–1031.

[Thai-Nghe et al. 2011] N. Thai-Nghe, L. Drumond, T. Horváth, A. Nanopoulos, and L. Schmidt-Thieme. 2011. Matrix and Tensor Factorization for Predicting Student Performance.. In *International Conference on Computer Supported Education*. 69–78.

[Velasquez et al. 2014] N. Velasquez, I. Goldin, T. Martin, and J. Maughan. 2014. Learning Aid Use Patterns and Their Impact on Exam Performance in Online Developmental Mathematics. In *Educational Data Mining 2014*. 379–380.

[Wen and Rosé 2014] M. Wen and C. P. Rosé. 2014. Identifying latent study habits by mining learner behavior patterns in massive open online courses. In *International Conference on Information and Knowledge Management*. 1983–1986.