Journal of Educational Data Mining

AN OPEN JOURNAL AVAILABLE AT
WWW.EDUCATIONALDATAMINING.ORG/JEDM/

VOLUME 7, NUMBER 1
2014

PUBLISHED BY THE

INTERNATIONAL EDUCATIONAL DATA MINING SOCIETY

Contents

Editorial Acknowledgement

The JEDM editor and associate editors express their sincere gratitude and thank the editorial board and colleagues who devoted their time and effort to reviewing for JEDM in 2014:

EDITORIAL BOARD

SPECIAL REVIEWERS

Laura Allan, Arizona State University, USA
Ma. Victoria Quintos Almeda, Teachers College Columbia University, USA
Yoav Bergner, Educational Testing Services, USA
Jean-Guy Blais, U. de Montréal, Canada
Matthew Bernacki, University of Nevada, USA
Sameer Bhatnagar, Polytechnique Montréal, Canada

Analysing Student Performance using Sparse Data of Core Bachelor Courses

Mirka Saarela
University of Jyväskylä
mirka.saarela@gmail.com

Tommi Kärkkäinen
University of Jyväskylä
tommi.karkkainen@jyu.fi

Curricula for Computer Science degrees are characterized by the strong occupational orientation of the discipline. In the BSc degree structure, with clearly separated CS core studies, learning skills between these and other required studies may vary a lot, showing in student's overall performance. To analyze such a situation, we apply nonstandard educational data mining techniques on a preprocessed log file of the passed courses. The joint variation of the course grades are studied through correlation analysis while the intrinsic groups of students are created and analyzed using a robust clustering technique. Since not all students have attended all the courses, there is a nonstructured sparsity pattern to cope with. Finally, multilayer perceptron neural network with cross-validation based generalization assurance is trained and analyzed using analytic mean sensitivity to explain the nonlinear regression model constructed. Local (within-methods) and global (between-methods) triangulation of different analysis methods is argued to improve the technical soundness of the presented approaches, giving more confidence on our final conclusion that general learning capabilities predict the success of students better than specific IT skills obtained as part of the core studies.

Keywords: Sparse Educational Data, Triangulation, Curricula Refinement, Correlation Analysis, Robust Clustering, Multilayer Perceptron

1. INTRODUCTION

The development of a curriculum for Computer Science (CS) can be challenging in an academic environment, given the strong occupational orientation of the discipline. Especially in multi-disciplinary universities (i.e., with many subject areas), the curriculum of CS differs from the curricula of many other disciplines, as the core courses reflect to a large extent the vocational side of the studies. In the case of the Department of Mathematical Information Technology (DMIT) at the University of Jyväskylä in Finland (reflecting both the Finnish and the European style of degree structure), the core bachelor courses compose only round 50 out of the minimum 180 ECTS (i.e., credits measured using the European Credit Transfer and Accumulation System) for the 3-year BSc degree (see Table 2). The degree contains other major studies along with separate introductory topics (e.g., science in general, language and communication skills, statistics) and minor subject studies (esp. mathematics). On one hand, students should acquire knowledge of very specific technical (e.g. programming) skills, on the other hand, computing interacts with many different domains, and in order to prepare students as the workforce of future, also domain knowledge as well as soft skills and personal attributes are important (Sahami et al., 2013a). Already over 40 years, roughly on a 10-year basis, ACM and IEEE promote the creation of international curricular guidelines for undergraduate programs in computing (Sahami et al., 2013b). So far, however, there has been little discussion about the relation between the specific CS courses and the other studies, by means of the overall study performance.

Some studies (see, e.g., Kinnunen et al. 2013 and references therein) indicate that primarily difficulties in mastering programming lead to high drop-out rates in CS and, therefore, one should pay special attention to them. Furthermore, a popular belief is that mathematical talent is the key skill for CS students to be successful (Jerkins et al., 2013). While these are important topics, they do not cover the whole degree. The CS core of the DMIT curriculum for the undergraduate students at the University of Jyväskylä, which is one of the largest and most popular multidisciplinary universities in Finland, has been more or less the same during the past years. Since the curriculum is typically updated on a three-year basis, the aim of this research is to focus on a set of mandatory courses related to the data collection period: $8/2009 - 7/2013$.

Besides, DMIT undergraduate students are known to need longer time to finish their studies compared to students of other disciplines at the University of Jyväskylä (Halonen, 2012). This happens even if the student's view on the quality of teaching and the study atmosphere in DMIT Jyväskylä is very positive, and, in fact, better than in the whole Faculty of Information Technology (where DMIT belongs) or in the rest of the departments at the university (Halonen, 2012). Actually, only very few students (on average 12.8%) of DMIT are completing the national target of at least 55 ECTS per academic year (Harden and Tervo, 2012). These study efficiency shortcomings apply both to the absolute and relative amount of studies and are especially worth mentioning in comparison with students of other departments at the University of Jyväskylä, who amass many more credits in an academic year (29% acquire at least 55 ECTS).

In order to assess the current curriculum, we apply educational data mining (EDM) approach. EDM consists of developing or utilizing data mining methods that are especially feasible for discovering novel knowledge originating from educational settings (Baker and Yacef, 2009), supporting decision-making in educational institutions (Calders and Pechenizkiy, 2012). Most of the current case studies in EDM (see Table 1) are devoted to analyzing the steadily growing amount of log data from different computer based learning environments, such as *Learning Management Systems* (e.g., Valsamidis et al. 2012), *Intelligent Tutoring Systems* (e.g., Hawkins et al. 2013; Bouchet et al. 2012; van de Sande 2013; Carlson et al. 2013; Springer et al. 2013), or even *Educational Games* (e.g., Kerr and Chung 2012; Harpstead et al. 2013). Mining those data supports the understanding of how students learn and interact in such systems.

In our study, on the other hand, we are interested in understanding the effects of core CS studies and providing novel information for the repetitive curricula refinements. More specifically, we want to understand the effect of the current profile of the core courses on the student's study success. These courses are taught in an ordinary fashion, meaning that in order to successfully complete a course, the student has to attend lectures, do related exercises, and pass a final exam or assignment at the end of the course. The data analyzed in this paper is the historical log file from the study database at DMIT about all the passed studies of the students for the period August 2009 [1] until the end of July 2013. Patterns in our data provide an improved profiling of the core courses and an indication of which study skills support timely and successful graduation.

[1]This is because since 8/2009 only ECTS-credits and separated Bachelor and Master degrees can be done.

The remainder of this paper is structured as follows. In Section 2 the overall methodology is explained. Section 3 is devoted to the correlation analysis, while Section 4 discusses our clustering analysis with robust prototypes. In Section 5, prediction analysis is realized with multilayer perceptron (MLP) neural network. Conclusions from the domain as well as from the methodological level are presented in Section 6.

2. The Overall Methodology: Advocating Multiphase Triangulation

Baker et al. (2010) classify EDM methods into five categories: prediction, clustering, relationship mining, discovery with models, and distillation of data for human judgment. In Table 1 we summarize a representative set of existing EDM studies according to a) their data and its environment, b) goal of the study, c) EDM category and the used methods, and d) the knowledge discovered. This work has been selected from forums, such as the Journal of Educational Data Mining, the related annual conferences, and Google Scholar during autumn 2013. According to the table, which is organized by the different tasks and publication dates, one usually applies methods belonging to one of the classes of Baker 's et al. taxonomy to address a particular EDM problem. Moreover, the predictive studies may apply many classifiers to assess the stability and reliability of the obtained results. We, on the other hand, aim at multiphase triangulation: different phases of the overall treatment both within-methods and between-methods are varied and assessed (using rankings) to increase both technical soundness of the procedures and the overall reliability of the concluded results.

Generally triangulation means that the same research objective is investigated by different data, theories, analysis methods, or researchers, and then combined to arrive at convergent findings (Denzin, 1970). Probably the most popular way to apply triangulation is to use qualitative and quantitative methods and unite their results (Jick, 1979). We employ *between-method* triangulation (e.g., Denzin 1970; Bryman 2003), using techniques belonging to distinct classes of the EDM taxonomy, to study success patterns of the students given the core courses of the computer science program in our department. First of all, we apply correlation analysis (Section 3), one key technique in *relationship mining*. Secondly, we utilize a special *clustering* approach (see Section 4) to find groups of students with similar study success. And thirdly, we apply *prediction* (see Section 5) with model sensitivity analysis. In all between-methods we discuss different *within-methods* that tighten the soundness of the respective between-method result. Moreover, we support our decision making a) in clustering by *distillation of data for human judgement* (see our explorative and visual analysis in Section 4.2.1), and b) in prediction by *discovery with models* (model sensitivity is used as a component to calculate the mean variable sensitivity of the prediction model, see Section 5.1). To combine and interpret our results from the individual EDM techniques, we introduce a ranking system to which all the between and within analysis methods contribute.

In practice, the whole knowledge discovery process in our study is conducted by following the classical five stages (select the target data from the application domain, preprocess, transform, mine the transformed data, and interpret the results) as introduced by Fayyad et al. (1996). Preprocessing and transformations of data were done in Java, while the data mining / machine learning techniques were either used as is (correlation analysis in Matlab's Mathematics package) or completely self-implemented (clustering and prediction as a whole) on the Mathworks

Matlab R2013b platform.

Table 1: Overview about related work.

Environment and Data	Goal	**Category**: Methods	Obtained Knowledge
(San Pedro et al., 2013), United States (New York):			
Interaction data of a web-based tutoring system for mathematics from 3747 middle schools students in New England plus college enrollment information for those students	Predict whether a student will (5 years later) attend college	**Prediction**: Logistic Regression Classifier	Students that are successful in middle school mathematics as measured by the tutoring system are more likely to enroll 5 years later in college, while students that show boredom, confusion, and slip/carelessness in the system have a lower probability of college enrollment.
(Vihavainen et al., 2013), Finland:			
Helsinki University, snapshot data from Computer Science student's programming course	Predict whether a student will fail introductory mathematics course	**Prediction**: Non-parametric Bayesian network tool (B-Course)	Students who start working close to deadline in their programming course are at high risk of failing their introductory mathematics course.
(Bayer et al., 2012), Czech Republic:			
Masaryk University, data of Applied Informatics bachelor students, their studies, and their activities in the information system of the university (e.g., communication with other students via email/discussion board)	Predict whether a bachelor student will drop-out from the university	**Prediction**: J48 decision tree learner, IB1 lazy learner, PART rule learner, SMO support vector machines, NB	Students who communicate with students having good grades can successfully graduate with a higher probability than students with similar performance but not communicating with successful students.
(Kotsiantis, 2012), Greece:			
Hellenic Open University, data from distance learning course on Informatics	Predict students' final marks	**Prediction** M5', BP, LR, LWR, SMOreg, M5rules	Two written assignments of the students predict their final grade the best.

Continued on next page

Table 1 – continued from previous page

Environment and Data	Goal	**Category**: Methods	Obtained Knowledge
(Bhardwaj and Pal, 2011), India:			
Purvanchal University, Department of Computer Applications, student data	Predict students' performance	**Prediction**: Bayesian Classifier	Living location has high influence on students' final grade.
(Mendez et al., 2008), United States (Arizona):			
Arizona State University, Science and Engineering student data	Prediction of student's persistence	**Prediction**: Decision Tree, Regression, Random Forest	High school and freshmen GPAs influence persistence the most.
(Erdogan and Tymor, 2005), Turkey:			
Maltepe University, data from student database	Find relations between entrance exam and later success	**Clustering**: K-means	The results of a student's university entrance exam determine in many cases the faculty of the student.
(Campagni et al., 2012), Italy:			
University of Florence, Department of Computer Science, data of how and when exams were taken	Determine whether students who take exams in recommended order are more successful	**Clustering**: K-means	Students who follow the *ideal path* perform better in terms of graduation time and final grade.
(Chandra and Nandhini, 2010), Nigeria:			
University in Nigeria, Department of Computer Science, course result data	Identify students' failure patterns	**Relationship Mining**: Apriori Association Rule Mining	Relationship between failed courses which can be used in order to restructure the curriculum (e.g., 2 introductory courses should be passed before the *Mathematical Modeling* course).

2.1. Data and Nonstructured Sparsity Pattern

The original data, the historical log files of the four years, $8/2009 - 7/2013$, of all the completed studies from all the students of the DMIT, is challenging: Students are in different stage of their studies, their mandatory courses depend on their starting semester, they come with varying backgrounds, have diverse interests, choose their optional courses accordingly, and, as a

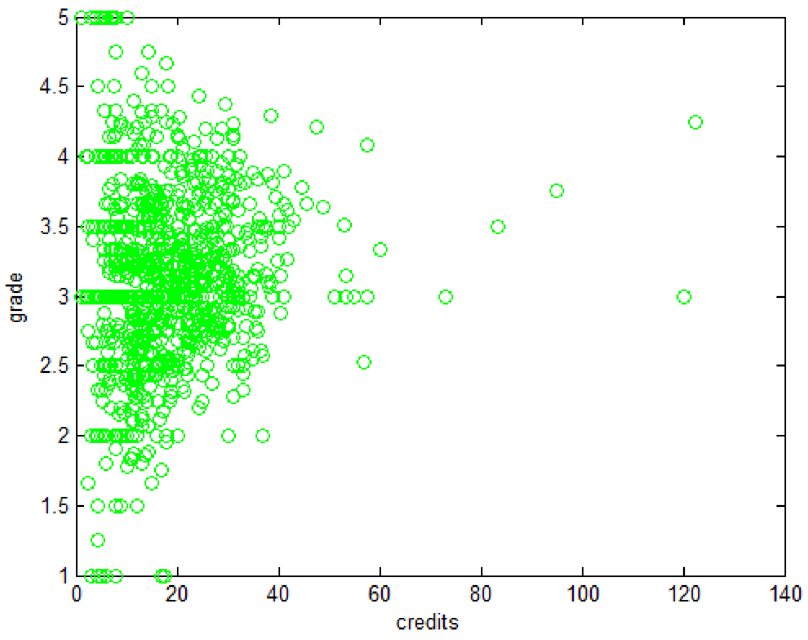


Figure 1: Relationship between the average credits per semester and grades.

consequence, realize very different study profiles. This can be a rather typical situation in multidisciplinary universities where students have the opportunity to choose from a large pool of courses. Altogether, our data set consists of 13640 study records with 21 attributes, related to the passed course and the student's affiliation, and of 1040 students who jointly have attended 1271 different courses, completing altogether 64905 credits. Only 64% of these credits the CS students obtained from courses of their own faculty.

When measuring the performance of the individual students, besides the quality, i.e. the grades, the quantity of studies, i.e. the number of all amassed credits, is important. However, since our data set consists of many students at different stages of their study, we cannot compare their individual sums of credits as is. Therefore, we assigned each passed course/record in our data set to a semester, so that *mean credits* over the active semesters could be computed for all students. Active semester, in turn, is computed as the sum of all semesters between the first semester and the last semester than a student successfully completed a course. For example, a student who passed his first course in April 2010 and his last course in June 2013 has 7 active semesters - this may also include semesters in which the student did not achieve any credits. The *mean grade* is simply the sum of all grades divided by the number of courses a particular student has passed.

In general, quality and quantity of the studies of DMIT students do not correlate. The correlation coefficient between the average number of credits per student and average grade is rather close to zero (0.0848). The studentwise plot of the relationship between the number of credits per semester and the average grade is shown in Figure 1. Also from the figure, that reminds of a turned bell curve which means that the grading of the studies resembles the normal distribution, it can be visually seen that the achieved credits per student do not correlate with the average grade.

Table 2: Core bachelor courses.

course name	course code	course type	completion mode[2]	credits
Computer and Datanetworks as Tools	PCtools	introductory	assignment	2-4
Datanetworks	Datanet	introductory	exercises & final exam	3-5
Object Oriented Analysis and Design[3]	OOA&D	professional	exercises&final exam	3-6
Algorithms 1	Alg1	professional	final exam	4
Introduction to Software Engineering	IntroSE	professional	final exam	3
Operating Systems	OpSys	professional	final exam	4
Basics of Databases and Data Management	DB&DMgm	professional	final exam	4
Programming 1	Prog1	programming	assigment&final exam	6
Programming 2	Prog2	programming	assigment&final exam	8
Computer Structure and Architecture	CompArc	introductory	exercises & final exam	3
Programming of Graphical User Interfaces	GUIprog	programming	exercises & final exam	5
Research Methods in Computing	CompRes	methodological	essay	2
All core courses				47-54

Our goal is to better understand the success patterns of the students, given the core courses, in relation to the rest of their studies. Therefore, we want to particularly analyze those students who already have completed a certain percentage of the courses of interest. The core courses, a specific set of 12 courses which, for that period of time we study, have been an obligatory part of the curriculum for all bachelor students of DMIT, are listed and characterized in Table 2.

If we transform our data in such a way that the 12 core courses become the variables and the attribute value of each observation, corresponding to one student, is the grade of the core course or *missing* in case the student did not attend or pass the course, the assembled matrix is very sparse. Only for thirteen students the rows are full, i.e., they have passed all the core courses. In Table 3, the high percentage of missing values and the sparsity of the matrix is summarized. The table shows how many students have completed exactly, and respectively at least, q of the 12 courses. Moreover, in each case the percentage of missing values of the cumulative data matrix is provided. The missing data values in the matrix are *missing at random* (Rubin, 1976; Rubin and Little, 2002). This means that the missing values are related to particular variables (some courses that are usually taken later in studies are completed by less students, see Figure 2) but not missing because of the values, i.e. grades, that could be observed if a particular course is passed.

To be able to analyze such data one cannot accept too many missing values. In this respect, the concept of *breakdown point* related to statistical estimates (see, e.g., Hettmansperger and McKean 1998) on how much contamination (errors, missing values) in data can be tolerated is informative. An upper bound is easy to establish: if more than 50% of data is missing, then

[2]The difference between *assignment* and *excercises* in our system are important: While *assignment* denotes a mandatory work that the student has to fulfill in order to pass the course and which also influences the final grade the student will receive, *exercises* are smaller (usually weekly) optional tasks that correspond to the current lecture material.

[3]In spring 2012, the *Object Oriented Analysis and Design* (*OOA&D*) course was splitted into two separate courses, namely *Object Oriented Analysis* and *Object Oriented Design*. Therefore, in further analysis the following strategy was applied: If a student completed the original *Object Oriented Analysis and Design* course, the grade from this course was taken for the analysis. However, in case the student did not attend the original course, we used the mean grade of the *Object Oriented Analysis* and the *Object Oriented Design* course as grade for *OOA&D* if the student had done both of the newly created courses, or just the grade of the one course if the student had completed only one of these two courses.

Table 3: Number of students who have completed exactly q (n_q) or at least q ($\sum_{q=12}^{Q} n_q,\ Q = 12, \ldots, 0$) of the core courses

q	n_q	$\sum n_q$	missing values
12	13	13	0.0%
11	16	29	4.56%
10	22	51	9.81%
9	26	77	14.93%
8	49	100	19.17%
7	28	128	24.10%
6	35	163	29.65%
5	44	207	35.75%
4	46	253	41.37%
3	40	293	45.96%
2	82	375	54.13%
1	126	501	63.57%
0	539	1040	82.45%

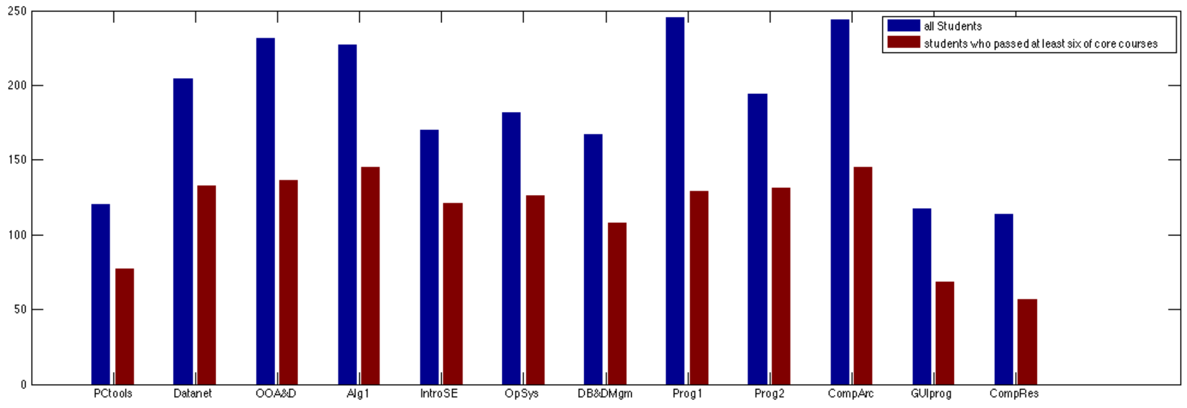


Figure 2: Number of students who passed coursewise.

"missing" is the most typical value (mode) of data. Furthermore, tests that have been conducted with synthetic data show that, for example in clustering with robust methods, reliable results, i.e. almost zero error, can be obtained even if around 30% of data is missing (Äyrämö 2006, see in particular Figure 22 at page 131). Therefore, our *data selection strategy* is to use that part of the whole, sparse data matrix, which contains those students who have completed at least half of the core courses. This data set has about thirty percent of missing values (see Table 3) for the multivariate techniques. In the correlation analysis (see Section 3), where the courses are analyzed individually, we similarly use those subsets of the students who have passed the particular course and at least five other courses additionally. In addition, different subsets of the sparse study matrix are utilized to realize some parts of both cluster analysis and predictive analysis procedures.

A further challenge, particularly for the prediction of the study success (see Section 5), is that, for our primary target group, the amount of studies related to the core courses is typically less than half of the total number of the obtained credits. In Table 4 the percentages of credits originating from the core courses in relation to the total number of studies for the 163 students

Table 4: Binning of students (nr = number) by means of the number of core courses in relation to whole studies.

% core courses	nr	cumulative (%)
0-10%	16	16 (10%)
10-20%	24	40 (25%)
20-30%	24	64 (39%)
30-40%	29	93 (57%)
40-50%	25	118 (72%)
50-60%	17	135 (83%)
60-70%	15	150 (92%)
70-80%	7	157 (96%)
80-90%	4	161 (99%)
90-100%	2	163 (100%)

of interest (see Table 3) are shown. As can be seen from the table, for more than 70% of the students, the core courses cover less than half of their studies.

Summing up, for our analysis we have, on one hand, the entire base of completed studies (1040x21) which is processed and transformed to further subsets, and, on the other hand, the sparse 163x12 data matrix of those students who have completed at least half of the core courses and the grades they achieved in these courses.

3. Correlation Analysis with Bonferroni Correction

As our first EDM technique, we apply relationship mining using correlation analysis. In general, we know from Figure 1 that in terms of grades well-scoring students are not necessarily more likely to study actively. But how about the correlation for our target group, those students who have already completed at least half of the core courses? In the correlation analysis we do not need special methods for the sparse data. However, it should be noticed that the number of students who have passed an individual course differs considerably (see Figure 2) so that the correlation coefficients are computed for different student subsets. The mean number of credits and mean grade are computed in the same way as explained in Section 2.1.

In Table 5, the correlation of each of the core course to (i) the mean grade of a student (denoted as *corr.grades*) and (ii) the mean number of credits per semester (denoted as *corr.credits*) is summarized. In each case, *r* identifies the calculated correlation and *p* corresponds to the *p-value* for testing the hypothesis of no correlation, respectively. The number of stars indicates how strong the evidence of no correlation is. As usual, $\star$ symbolizes the *borderline to be significant* ($p <= 0.05$), $\star\star$ symbolizes *statistically significant* ($p <= 0.01$), and $\star\star\star$ symbolizes *highly statistically significant* ($p <= 0.005$). *rank* denotes the ordering of courses by means of the computed correlations.

From Table 5 we can conclude that, except the *Research Methods in Computing*, all the courses have a moderate positive linear relationship to the general study success of the student. The coursewise correlations to mean credits per semester are all positive as it should be (passing a course increases credits). Also all *corr.grades* illustrate that students who score high in those courses tend to score high in their other courses as well. In particular, this applies to the four courses: *Algorithms 1*, *Computer Structure and Architecture*, *Datanetworks*, and *Programming 2*. The found correlation between the grade of these four courses and the average grade of the

Table 5: Correlation of each of the core courses to the general performance of the student.

	corr.grades				corr.credits			
Course Code	*r*	*p*	*bonferroni*	*rank*	*r*	*p*	*bonferroni*	*rank*
PCtools	0.4164	⋆⋆⋆	⋆⋆⋆	11	0.1058	–	–	12
Datanet	0.6244	⋆⋆⋆	⋆⋆⋆	**3**	0.3887	⋆⋆⋆	⋆⋆⋆	**1**
OOA&D	0.5346	⋆⋆⋆	⋆⋆⋆	6	0.1327	–	–	11
Alg1	0.6593	⋆⋆⋆	⋆⋆⋆	**1**	0.3082	⋆⋆⋆	⋆⋆⋆	**4**
IntroSE	0.4197	⋆⋆⋆	⋆⋆⋆	10	0.1717	–	–	9
OpSys	0.5113	⋆⋆⋆	⋆⋆⋆	8	0.1905	⋆	–	8
DB&DMgm	0.5572	⋆⋆⋆	⋆⋆⋆	5	0.3312	⋆⋆⋆	⋆⋆	5
Prog1	0.4314	⋆⋆⋆	⋆⋆⋆	9	0.2438	⋆⋆	–	7
Prog2	0.5731	⋆⋆⋆	⋆⋆⋆	**4**	0.3549	⋆⋆⋆	⋆⋆⋆	**2**
CompArc	0.6511	⋆⋆⋆	⋆⋆⋆	**2**	0.3216	⋆⋆⋆	⋆⋆⋆	**3**
GUIprog	0.5343	⋆⋆⋆	⋆⋆⋆	7	0.3054	⋆	–	6
CompRes	0.2543	–	–	12	0.1609	–	–	10

student is in all cases highly statistically significant as the p-values for testing the hypothesis of no correlation are all smaller than 0.005. Similarly as with the classical p-test, we obtained also with the conservative *Bonferroni Correction* (Rice, 1989) that the correlation of all the core courses to the overall grade of the student (except the *Research Methods in Computing*) are highly statistical relevant.

Another conclusion that can be made from Table 5 is that the same four courses which have the highest correlations to the general success of the student, also have the highest correlation to the average number of credits. This means that if a student gets a high grade in these courses s/he will probably make, on average, a rather high number of credits in a semester as well. Again, all of these findings are, according to the classical p-test as well as according to bonferroni correction, highly statistically significant. Although the ranking is different (e.g., while *Algorithm 1* correlates the most with mean grade of the student, *Datanetworks* correlates the most with the mean amount of credits per semester), we can conclude that those four courses correlate with the general performance of the students the best.

To sum up, it can be inferred that a student who achieves a high grade in *Algorithms 1*, *Computer Structure and Architecture*, *Datanetworks*, or *Programming 2* is likely to be successful in the remaining part of his/her studies - not only by means of the level of grades but also in terms of speed of studies. Albeit overall semesterwise credits and average grade do not correlate at all (see Figure 1), a linear dependency between the grades a student got in the core courses and the general performance exists.

4. Cluster Analysis using Robust Prototypes

Our second EDM method of this study is clustering. Generally, clustering can be divided into *partitional* and *hierarchical* clustering (Jain, 2010; Steinbach et al., 2004). However, hierarchical clustering is only appropriate in very small data sets since most of the hierarchical algorithms have quadratic or higher computational complexity (Emre Celebi et al., 2012). Partitional clustering, on the other hand, is known to be very efficient and scalable. It attempts to seek a partition of the data, such that similar observations are assigned to the same subset of data (referred as cluster), each observation is attributed to exactly one subset, and each subset contains at least

Algorithm 1: Iterative relocation clustering algorithm

```
Input: Dataset and the number of clusters K.
Output: K partitions of the given dataset.
Select K points as the initial prototypes;
repeat
    1. Assign individual observation to the closest prototype;
    2. Recompute the prototypes with the assigned observations;
until The partition does not change;
```

one observation. As we want to obtain a directly interpretable result, prototype-based partitional clustering is an appropriate approach here. If we can find a partition of data, where each cluster is represented by exactly one prototype, we can use this prototype to analyze the corresponding cluster. Prototype-based partitional clustering can be realized using the iterative relocation algorithm skeleton presented in Algorithm 1 with different score functions (Han et al., 2001) according to which the two steps inside the loop of Algorithm 1 are optimized.

However, in order to realize a prototype-based partitive clustering algorithm, two main issues should be addressed. First of all, a well-known problem of all iterative relocation algorithms is their initialization. They minimize the given score function locally by iteratively relocating data points between clusters until an optimal partition is attained. Therefore, basic iterative algorithms, such as K-means, always converge to a local, and not necessarily to the global, optimum. Although a lot of work has been attributed to this problem, still no efficient and universal method for identifying the initial partitions and the number of clusters exists. This problem is discussed more thoroughly in Section 4.2. The second problem is the sparse student data with around 30% missing values (see Section 2.1). In Section 4.1, a solution is presented how one can adjust the score function of the basic algorithm skeleton in order to deal with a random sparsity pattern. A similar approach was also applied in a recent work to other educational data (Saarela and Kärkkäinen, 2014).

4.1. Score function for K-spatialmedians

Our (available) data consists of course grades of fixed values $\{1, 2, 3, 4, 5\}$. Therefore, there is evidently a significant quantization error from uniform distribution in the probability distribution for a grade g_i:

$$g_i(x) = \begin{cases} 1, \text{if } g_i - \frac{1}{2} \leq x < g_i + \frac{1}{2}, \\ 0, \text{elsewhere.} \end{cases} \tag{1}$$

Hence, second order statistics which relies on the normally distributed error is not suitable here, and we need to use the so-called nonparametric, i.e., robust statistical techniques (Huber, 1981; Rousseeuw and Leroy, 1987; Hettmansperger and McKean, 1998). The most simplest of robust location estimates are median and spatial median. Median, i.e., the middle value of the ordered univariate sample, is inherently one-dimensional, and hence with missing data uses only the available values of an individual variable. Spatial median, on the other hand, is truly a multi-dimensional location estimate and can take advantage of the available data pattern as a whole. This is illustrated and more thoroughly explained by Kärkkäinen and Heikkola (2004) (especially, formulae (2.8) and (2.9) and Figures 1 and 2). Like stated, e.g., by Croux et al. (2010), the spatial median is not affine but only orthogonally equivariant. However, because we have the fixed scale of grades, such property of a statistical estimate is not necessary here. Moreover, for elliptical distributions this behavior effects more scatter than location estimation (Croux et al., 2010). As a whole, the spatial median has many attractive statistical properties; especially its breakdown point is 0.5, i.e., it can handle up to 50% of contaminated data which makes it very appealing for high-dimensional data with severe degradations and outliers. Basically a missing value can be thought of as an infinite outlier because it can have any value (from the value range).

Äyrämö (2006) introduced a robust approach utilizing the spatial median to cluster very sparse and apparently noisy data: The *K-spatialmedians* clustering algorithm is based on the same algorithm skeleton as presented in Algorithm 1 but uses the projected spatial median as a score function:

$$\mathcal{J} = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \| \operatorname{diag}\{\boldsymbol{p}_i\}(\boldsymbol{x}_i - \boldsymbol{c}_j)\|_2, \tag{2}$$

Here, diag transforms a vector into a diagonal matrix. The latter sum in (2) is computed over the subset of data attached to cluster j and the projection vectors $\boldsymbol{p}_i, i = 1, \ldots, N$, capture the existing variable values:

$$(\boldsymbol{p}_i)_j = \begin{cases} 1, \text{if } (\boldsymbol{x}_i)_j \text{ exists,} \\ 0, \text{otherwise.} \end{cases}$$

In Algorithm 1, the projected distance as defined in (2) is used in the first step, and recomputation of the prototypes, as spatial median with the available data, is realized using the SOR (Sequential Overrelaxation) algorithm (Äyrämö, 2006) with the overrelaxation parameter $\omega = 1.5$. In what follows, we refer to Algorithm 1 with the score function (2) as *K-spatialmedians* clustering.

4.2. Initialization

It is a well-known problem that all iterative clustering algorithms are highly sensitive to the initial placement of the cluster prototypes and, thus, such algorithms do not guarantee unique clustering (Meilă and Heckerman, 1998; Emre Celebi et al., 2012; Bai et al., 2012; Jain, 2010). One might even argue that the obtained results are not reliable if initial prototypes are randomly chosen since the algorithms do not converge to a global optimum. Numerous methods have been introduced to address this problem. Random initialization is still often chosen as the general strategy (Xu and Wunsch, 2005). However, several researchers (e.g., Aldahdooh and Ashour 2013; Bai et al. 2011) report that having some other than random strategy for the initialization often improves final clustering results significantly.

An important issue when clustering data and finding an appropriate initialization method for it is the definition of (dis-)similarity of objects. Bai et al. (2011) and Bai et al. (2012) proposed initialization methods for categorical data. The attribute values of our data set (grades from 1-5, or missing) are also categorical. However, ordering of our attribute values has a meaning (ordinal data). For example, a student who got grade 5 in all his/her courses is more dissimilar to a student who got mostly grade 2 than to a student who received on average grade 4. Therefore, an initialization method for data where only enough information is given to distinguish one object from another (nominal data) might not be suitable for our case.

Also Chen et al. (2009) proposed a novel approach to find good initial prototypes. Chen et al. argue that in the high-dimensional space data are inherently sparse. Therefore, the distance

Algorithm 2: Constructive initialization approach for robust clustering

```
Input: Datasets $\mathbf{D_0}$ to $\mathbf{D_6}$.
Output: The set of prototypes for every value of $K$
for $K = size(D_0)$ to $2$ do
    $KBestPrototypes$ = globalBestSolution($D_0$,K);
    for $p = 1$ to $6$ do
        $KBestPrototypes$ = K-spatialmedians($D_p$,$K$,$KBestPrototypes$);
    end
end
```

between each pair of observations becomes almost the same for a wide variety of data distributions. However, this approach seems more suitable for very high-dimensional data than for our 12-dimensional case. Emre Celebi et al. (2012) compared different initialization methods. They conclude that for small data sets (less than 10000 observations) the method by Bradley and Fayyad leads to best results. In Bradley and Fayyad's method (Bradley and Fayyad, 1998) the original data set is first splitted into smaller subsets which themselves are clustered. Then those temporary prototypes obtained from clustering the subsets are combined and clustered as many times as there are different subsets. Hereby, each time one different set of temporary prototypes is tried as initialization and the best, i.e., that set of temporary prototypes which resulted in the smallest clustering error, is finally used as initialization for clustering the original data set.

To sum up, it can be said that the ideal approach to compute initial prototypes depends very much on data, and is therefore context dependent. However, some criteria apply in general: First of all, initial prototypes should be as far from each other as possible (Khan and Ahmad, 2013; Jain, 2010). Second, outliers or noisy observations are no good candidates as initial prototypes. Moreover, for relatively small data sets it seems to be a good idea to further divide the set into subsets and utilize the respectively best prototypes of the smaller sets for further computations. Furthermore, as pointed out by Bai et al. (2012), it is advantageous if at least one initial protoype is close to a real solution. Having all of the issues above in mind, we have developed a new deterministic and context-sensitive approach to find good initial prototypes.

4.2.1. Initialization for sparse student data

As pointed out above, our intention is to interpret and characterize each cluster by its prototype. Therefore, we should prefer full prototypes, i.e., prototypes which have no missing values. For such an approach, we first note that the rows of Table 3 represent cascadic (see Kärkkäinen and Toivanen 2001) sets of data. Let us denote those data sets as D_p with $p = 0 \dots 12$, where $p = q - 12$. Hence, D_0 represents the very small but full data set with those 13 students who have completed all the 12 core courses and D_1 those 29 students who have completed at least 11 of them (containing D_0). Hence, in general D_p consists of those students who have passed exactly $12 - p$ of the core courses, and we always have $D_{p-1} \subset D_p$. This creates the basis of the proposed initialization approach, which is depicted as a whole in Algorithm 2.

Our initial, full data set D_0 is so small that we can easily determine the globally best solution by means of minimizing the error of the spatial median by testing all possible initializations for the values of K[4]. In Algorithm 2, *globalBestSolution* refers a function that tests all possible

[4]Even if K is unknown we can assume that K is at least 2 and smaller than the total number of observations

Table 6: Comparison of context-sensitive and random initialization for robust clustering.

	context-sensitive		random	
K	error	missing values	error	missing values
13	373.54	15.38%	425.46	51.54%
12	374.04	8.33%	429.26	46.67%
11	372.31	0.00%	432.89	38.18%
10	376.57	0.00%	434.51	44.00%
9	391.42	0.00%	436.98	38.89%
8	396.06	0.00%	434.77	33.75%
7	409.70	0.00%	444.50	31.43%
6	425.93	0.00%	443.27	18.33%
5	437.32	0.00%	452.74	16.00%
4	454.27	0.00%	461.71	10.00%
3	471.79	0.00%	480.99	6.66%
2	506.84	0.00%	515.06	5.00%

K combinations of the observations in the small full data set and returns the prototypes of that combination which resulted in the smallest clustering error. In that way we obtain for every K for our small data set the K global best prototypes. We then use those K best prototypes (denoted as $KBestPrototypes$ in the algorithm) on D_p as initial prototypes for the next larger data set D_{p+1}. Hence, throughout the constructive approach full prototypes and small clustering error are favored. The data set D_6, the students who have completed at least half of the core courses, is our actual target data for clustering.

In Table 6, it is shown how the score function changes and what is the amount of missing values with the proposed initialization strategy for different values of K for D_6. For comparison, the table also shows the average results of 10 test runs of the *K-spatialmedians* algorithm with the random initialization. Indeed, we obtain better results with our approach both by means of the clustering error and, especially, with respect to the missing values. For example, already for $K = 3$, 6.66% of the prototypes' values are missing with random initialization and, hence, uninterpretable. Moreover, we also studied the stability of the results by checking whether the students in $D_{p-1},\ p \geq 1$, still belong to the same cluster when new students are added and the reclustering of D_p is performed in Algorithm 2. For this purpose, confusion matrices between the two consecutive clustering levels were computed. It turned out that the confusion matrices are almost perfect, so that the formation of clusters is very stable and the clusters themselves are reliably structured. We conclude that the proposed context-sensitive initialization provides a clustering result with low error and high interpretability.

The best value for K is next determined using visual inspection. To avoid overfitting, our goal is to have a fairly small number of clusters. However, the observations should not be too far away from the prototype they belong to. From Figure 3, the plot of the second column in Table 6 (change in the score function when D_6 is clustered using the proposed strategy), we conclude that $K = 3, K = 5$, and $K = 8$ are potential values for the number of clusters. Namely, after precisely these points the speed of decrease (improvement) of the clustering error, i.e., the discrete derivative, slows down slightly (see, e.g., Zhong et al. 2008 for a similar approach). Out of the potential values, we choose the first one providing the smallest number of clusters to be

which in our case is 13.

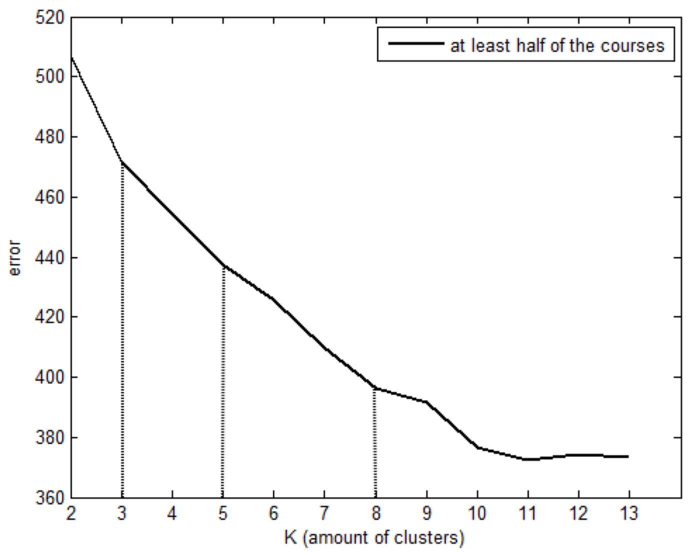


Figure 3: Decrease of error for target data when more clusters are introduced successively.

further analyzed and, in such a way, generalizing data the most. The prototypes for $K = 3$ are visualized in Figure 4.

4.3. Analyzing the clustering result

In the first two columns of Table 7, the ranking of the core courses based on their prototype separation is provided. Since the general profile of the three clusters is ”medium” (*cluster 1*), ”high” (*cluster 2*), and ”low” (*cluster 3*), we compute, for each variable, two distances: $d_1 = |C_2 - C_1|$ and $d_2 = |C_3 - C_1|$. The two measures are computed (i) as the mean of $\{(d_1)_i, (d_2)_i\}$ (denoted as *measure I*) and (ii) the minimum of $\{(d_1)_i, (d_2)_i\}$ (denoted as *measure II*). As can be seen from the table, *measures I* and *II* provide practically the same ranking. However, we think that out of these two indicators, the second measure provides clearer variable separation. For example, with *measure I* we could have a high distance value for a course even if only one prototype value C_i would be very dissimilar from the other two. Moreover, in order to assess even further the explanative power of variables related to the clustering result with $K = 3$, we also applied the nonparametric Kruskal-Wallis test (Hollander et al., 2013) to compare the subsets of data in the three clusters. Since the actual clusterwise datasets contain missing values, we used one iteration of the so-called *hot deck imputation* (Äyrämö, 2006; Batista and Monard, 2003) to complete them, i.e. we imputed the missing values using cluster prototype values of the *K-spatialmedians* algorithm (see Section 4.1) with 8 clusters (see Figure 3). As concluded in Section 4.2.1, 8 was another good value for the number of clusters *K*. Note that both because of this imputation and because of the form of the quantization error as explained in connection with formula (1), a nonparametric test should be used. According to the Kruskal-Wallis test, the difference between the different clusters is for all the courses highly statistically significant. Again, the same four courses provide the highest differentiation between the clusters (see third column of Table 7) with only the *Operating Systems* being very different compared to the distance-based separators. In the fourth column of Table 7, the sum of ranks from the

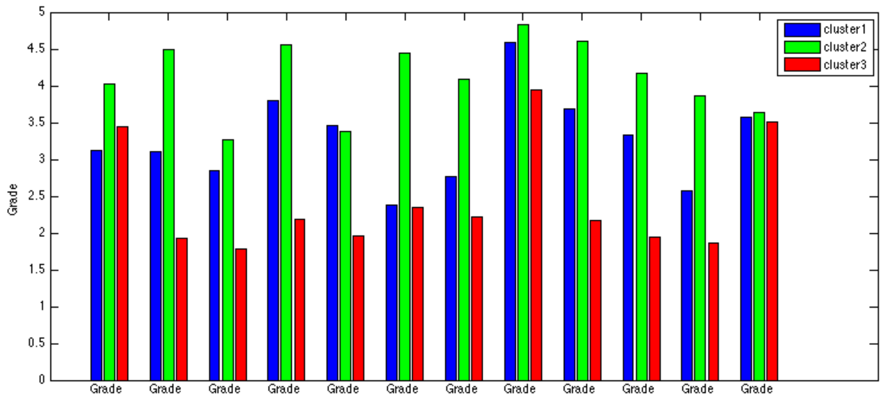


Figure 4: Prototypes of the three student clusters.

Table 7: Distances between the clusters.

	measure I		measure II		Kruskal-Wallis			
course code	distance	*rank*	distance	*rank*	χ^2	p	*rank*	sum (*rank*)
PCtools	0.1472	8	0.3220	8	33.10	⋆⋆⋆	9	17 (9)
Datanet	0.8183	1	1.1754	1	84.69	⋆⋆⋆	1	2 (**1**)
OOA&D	0.2249	7	0.4247	7	47.97	⋆⋆⋆	7	14 (6)
Alg1	0.6090	3	0.7501	4	82.39	⋆⋆⋆	2	6 (**2**)
IntroSE	0.0588	10	0.0781	10	34 .94	⋆⋆⋆	8	18 (10)
OpSys	0.0413	11	0.0402	12	67.06	⋆⋆⋆	4	16 (7)
DB&DMgm	0.3666	6	0.5490	6	53.96	⋆⋆⋆	6	12 (5)
Prog1	0.0796	9	0.2417	9	32.21	⋆⋆⋆	10	19 (11)
Prog2	0.7064	2	0.9309	2	54.35	⋆⋆⋆	5	7 (**4**)
CompArc	0.5872	4	0.8439	3	78.49	⋆⋆⋆	3	6 (**3**)
GUIprog	0.4602	5	0.7118	5	31.04	⋆⋆⋆	11	16 (8)
CompRes	0.0021	12	0.0633	11	17.23	⋆⋆⋆	12	23 (12)

second distance measure and Kruskal-Wallis are given and the overall ranking of the courses based on the sum is provided. This rank-of-rankings approach is an example of within-method triangulation, where the final order of importance combines assessments of the prototypes and the clusterwise data subsets.

The first observation that can be made by comparing the overall cluster ranking with the correlation analysis (see Table 5) is that the correlations are reflected in the different clusters. The four courses with the highest correlations clearly separate the three clusters. This can be seen as well from the visualization of the cluster prototypes (Figure 4). Those students in the lowest performing *cluster 3* also show the lowest performance in the *Datanetworks* and *Algorithms 1* course. And the prototype of the best *cluster 2* is represented by a remarkable higher grade for those courses. The same applies for the other two courses with a high correlation to the average grade and average number of credits of the students, *Computer Structure and Architecture* and *Programming 2*. A second interesting observation is that one of the smallest deviations in the grade is obtained for the *Research Methods in Computing*. This was also the only course that

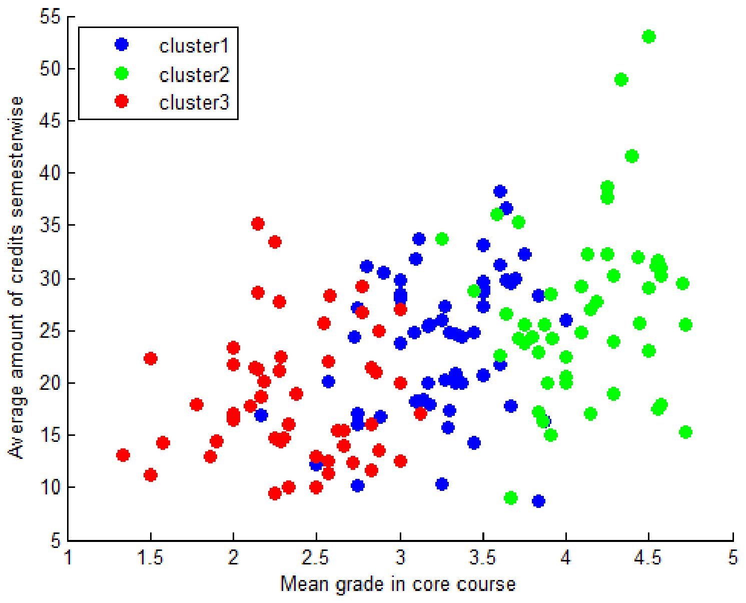


Figure 5: Semesterwise credits versus mean grade in core courses for the students in each cluster.

did not show a significant correlation to the students' overall grade (see Section 3). However, also the content of this course differs from the other core courses by being not directly related to IT knowledge. Moreover, in contrast to all other courses, this course is solely evaluated by an essay that the student has to write (see Table 2). A somewhat exceptional behavior for this course was expected.

Qualitywise the students in D_6 are clearly separated by the three clusters. In order to check whether the clusters also differentiate the students according to their quantity of study, we looked also (see Section 2.1 and 3) here at their average number of credits. In Figure 5, the semesterwise relation of grades in the core courses and the overall credits of the individual students in the different clusters is visualized. From this figure we deduce that the students who belong to *cluster 2* not only are the best when it comes to the average grades in the core courses but they also are the most efficient as they make on average the most number of credits per semester. Likewise, the students in the grade-wise low performing *cluster 3* also make the fewest credits per semester (on average 8 credits less than the students in *cluster 2*). The correlation coefficient of the mean grade in the core courses and the average number of credits semesterwise per student is 0.4415 with a p-value that is highly statistical significant. We know that such relation does not exist in the whole student level and when the average of all studies is used (see Figure 1). Hence, we can conclude that inside their own core CS studies those students who are performing well by means of grades also perform well by means of the amount of studies.

5. Predictive Analysis using Multilayer Perceptron

The goal, when addressing the third EDM category in this study, is an attempt to predict the mean grades and credits of the students given only the grades of the core courses they have passed. Similarly as in Section 4, we are interested in interpretable results, which here corresponds to detecting those inputs, i.e., courses, that contribute to the prediction model the most. Concerning the model, Multilayer perceptron (MLP) neural networks are universal nonlinear regression approximators (see, e.g., Pinkus 1999 and articles therein), which can be used in supervised learning. The feedforward MLP transformation starts directly from the input variables, different from other popular techniques like Radial Basis Function Networks or Support Vector Machines, which construct their basis in the space of observations. This is an appropriate starting point here, because the purpose, in what follows, is to assess the importance of the model inputs, which correspond to the core courses being analyzed. In this way, we close our between-method triangulation by contrasting the previous results and conclusions based on unsupervised analysis with the corresponding results from a supervised, predictive technique.

There are many inherent difficulties when a flexible model is used in prediction and trained using a given set of input-output samples. Firstly, because of the universality, such a model could actually represent the discrete data set precisely (e.g., Tamura and Tateishi 1997; Huang 2003), which especially would mean that also all the noise in the samples would be reproduced. Hence, one needs to restrict the flexibility of such models. Basically, this can be done in two ways: by restricting the size of the network's configuration (number and size of layers; structural simplicity) or restricting nonlinearity of the encoded function (size of weights, see Bartlett 1998; functional simplicity). Here we will assess the network's simplicity along both of these dimensions, in order to favor and restore the simplest model (cf. Occam's razor). Secondly, we look for a prediction model that provides the best generalization of the sample data, and, for this purpose, apply the well-known stratified cross-validation (see Kohavi 1995) to compute an estimate of the generalization error. Stratification means that, given a certain labelling to encode classes in a discrete dataset, the number of samples in the created folds (subsets) coincide with the sizes of the different classes as closely as possible. Clearly, number of classes and number of folds do not need be the same. Thirdly, as in clustering, use of a local optimizer to solve the nonlinear optimization problem to determine the network weights provides only local search (exploitation) and for exploration, we use multiple restarts with random initialization (see Kärkkäinen 2002). The whole training approach as just summarized has been more thoroughly introduced and tested by Kärkkäinen (2014), and succesfully applied in time-series analysis by Kärkkäinen et al. (2014).

Next we will derive and detail the whole predictive approach. First, MLP neural network and its determination are formalized, and then the overall training algorithm and the input-sensitivity analysis are developed and described.

5.1. Prediction with input sensitivity analysis

5.1.1. MLP training approach

Action of the multilayer perceptron in a layerwise, compact form can be given by (e.g., Hagan and Menhaj 1994)

$$\mathbf{o}^0 = \mathbf{x}, \quad \mathbf{o}^l = \mathcal{F}^l(\mathbf{W}^l \tilde{\mathbf{o}}^{(l-1)}) \text{ for } l = 1, \dots, L. \tag{3}$$

Here the layer number (starting from zero for the input) has been placed as an upper index. By $\tilde{}$ we indicate the addition of bias-terms to the transformation, which is realized by enlarging a vector $\mathbf{v}$ with constant: $\tilde{\mathbf{v}}^T = \begin{bmatrix} 1 & \mathbf{v}^T \end{bmatrix}$. In practice, this places the bias weights as first columns of the layer matrices which then have the factorization $\mathbf{W}^l = \begin{bmatrix} \mathbf{W}_0^l & \mathbf{W}_1^l \end{bmatrix}$. $\mathcal{F}^l(\cdot)$ denotes the application of activation functions on the lth level. Formally this corresponds to matrix-vector multiplication where the matrix components are functions and componentwise multiplication is replaced with application of the corresponding component function (Kärkkäinen, 2002). The dimensions of the weight-matrices are given by $\dim(\mathbf{W}^l) = n_l \times (n_{l-1}+1)$, $l = 1, \ldots, L$, where n_0 is the length of an input-vector $\mathbf{x}$, n_L the length of the output-vector $\mathbf{o}^L$, and $n_l, 0 < l < L$, determine the sizes (number of neurons) of the hidden layers.

Using the given training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, with $\mathbf{x}_i \in \mathbb{R}^{n_0}$ denoting the input-vectors and $\mathbf{y}_i \in \mathbb{R}^{n_L}$ the output-vectors, respectively, the unknown weight matrices $\{\mathbf{W}^l\}_{l=1}^L$ in (3) are determined as a solution of an optimization problem

$$\min_{\{\mathbf{W}^l\}_{l=1}^L} \mathcal{J}(\{\mathbf{W}^l\}). \tag{4}$$

Here we restrict ourselves to MLP with one hidden layer and the actual cost function reads as follows:

$$\mathcal{J}(\mathbf{W}^1, \mathbf{W}^2) = \frac{1}{2N} \sum_{i=1}^N \left\| \mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)(\mathbf{x}_i) - \mathbf{y}_i \right\|^2 + \frac{\beta}{2n_1} \sum_{(i,j)} \left(|\mathbf{W}^1_{i,j}|^2 + |(\mathbf{W}^2_1)_{i,j}|^2 \right) \tag{5}$$

for $\beta \geq 0$ and $\mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)(\mathbf{x}_i) = \mathbf{W}^2 \tilde{\mathcal{F}}^1(\mathbf{W}^1 \tilde{\mathbf{x}}_i)$. The special form of regularization omitting the bias-column $\mathbf{W}^2_0$ is due to Corollary 1 by Kärkkäinen (2002): *Every locally optimal solution to* (4) *with the cost functional* (5) *provides an unbiased regression estimate having zero mean error over the training data.*

The universal approximation property guarantees potential accuracy of an MLP network for a given data and the unbiasedness as just described provides statistical support for its use, but, as explained above, we also address the network's *simplicity* and *generalization*. Hence, in our actual training method we grid-search both the size of the hidden layer n_1 and the size of the regularization coefficient β: the smaller n_1, the simpler the structure of the network is; and the larger β, the smaller the weight values are and the closer the MLP to a (simpler) linear, single-layered network is. Moreover, cross-validation is used as the technique to assure that generalization ability of the network is taken as the main criterion of accuracy. Finally, the usual gradient based optimization methods for minimizing (5) act locally, so that we repeat the optimization with the random initialization twice when searching for the values of metaparameters n_1 and β. When these have been fixed, the final network is optimized using five local restarts to further improve the exploration of the search landscape.

The whole training approach for the MLP network is given in Algorithm 3. We use the following set of possible regularization parameter values, which were determined according to prior computational tests:

$$\vec{\beta} = \begin{bmatrix} 10^{-2} & 7.5 \cdot 10^{-3} & 5 \cdot 10^{-3} & 2.5 \cdot 10^{-3} & 10^{-3} & 7.5 \cdot 10^{-4} & 5 \cdot 10^{-4} & 2.5 \cdot 10^{-4} & 10^{-4} \end{bmatrix}.$$

The prediction error with a training or test set is computed as the mean Euclidian error

$$\frac{1}{N} \sum_{i=1}^N \left\| \mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)(\mathbf{x}_i) - \mathbf{y}_i \right\|. \tag{6}$$

We use the most common sigmoidal activation functions $s(x) = \frac{1}{1+\exp(-x)}$ for $\mathcal{F}^1$. All input variables are preprocessed into the range $[0, 1]$ of $s(x)$ to balance their scaling with each other

Algorithm 3: Reliable determination of MLP neural network.

```
Input: Training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$.
Output: MLP neural network $\mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)$.
Define a vector $\vec{\beta}$ of regularization coefficients, maximum size of the hidden layer
$n1max$, and $nfolds$, the number of folds for cross-validation, created using stratified
random sampling;
for $n_1 \leftarrow 1$ to $n1max$ do
    for $regs \leftarrow 1$ to $|\vec{\beta}|$ ($|\cdot|$ denotes the size of a vector) do
        for $k \leftarrow 1$ to $nfolds$ do
            for $i \leftarrow 1$ to 2 do
                Initialize $(\mathbf{W}^1, \mathbf{W}^2)$ from the uniform distribution $\mathcal{U}([-1, 1])$;
                Minimize (5) with current $n_1$ and $\vec{\beta}(regs)$, and the CV Training set;
                Store Network for smallest Training_Set_Prediction_Error;
            end
            Compute Test_Set_Prediction_Error for the stored Network;
        end
        Store $n_1^* = n_1$ and $\beta^* = \beta$ for the smallest mean Test_Set_Prediction_Error;
    end
end
for $i \leftarrow 1$ to 5 do
    Initialize $(\mathbf{W}^1, \mathbf{W}^2)$ from $\mathcal{U}([-1, 1])$;
    Minimize (5) using $n_1^*, \beta^*$ and the whole training data;
end
```

and with the range of the overall MLP transformation (see Kärkkäinen 2002 for more thorough argumentation).

5.1.2. Derivation of input sensitivity of MLP

To assess the relevancy of input (see John et al. 1994; Kohavi and John 1997) of an MLP model, one basic technique is to estimate the sensitivity of the network's output with respect to its input. Seven possible definitions of sensitivity were compared by Gevrey et al. (2003) in an ecological context and four of these, further, in relation to chemical engineering by Shojaeefard et al. (2013). Both comparisons concluded that in order to assess the relevancy and also rank the features, the 'PaD' (Partial Derivatives) method proposed by Dimopoulos et al. (1995) provides appropriate information and computational coherency in the form of stability. Hence, we also use the analytic partial derivative as core of the sensitivity measure, but in a more general and more robust fashion than Dimopoulos et al. (1995).

An analytical formula for the MLP input sensitivity can be directly calculated from the layer-wise formula (3). The precise result is stated in the next proposition.

Proposition 1

$$\nabla_{\mathbf{x}} \mathcal{N}(\{\mathbf{W}^l\})(\mathbf{x}) = \frac{\partial \mathbf{o}^L}{\partial \mathbf{x}} = \prod_{l=L}^{1} \operatorname{diag}\left\{(\mathcal{F}^l)'\right\} \mathbf{W}_1^l. \tag{7}$$

Here $\mathbf{W}_1^l$ denotes, as before, the lth weight matrix without the first bias-column. In particular, for an MLP with one hidden layer and linear output ($\mathbf{o}^2 = \mathbf{W}^2\,\tilde{\mathcal{F}}^1(\mathbf{W}^1\,\tilde{\mathbf{x}})$), (7) states that

$$\frac{\partial \mathbf{o}^2}{\partial \mathbf{x}} = \mathbf{W}_1^2 \operatorname{diag}\{(\mathcal{F}^1)'\}\,\mathbf{W}_1^1. \tag{8}$$

With a discrete data $\{\mathbf{x}_i\}_{i=1}^N$, input sensitivity must be assessed and computed over the data set. Hence, we apply (7) to compute the *mean absolute sensitivity, MAS* (see Ruck et al. 1990):

$$\frac{1}{N}\sum_{i=1}^{N}\left|\frac{\partial \mathbf{o}^L}{\partial \mathbf{x}_i}\right| \tag{9}$$

of the trained network for all input variables. After applying this formula, the approach for input ranking is based on the following concept: the higher the MAS the more salient the feature is for the network. This is due to the well-known Taylor's theorem in calculus related to local approximation of smooth functions (see Apostol 1969). Namely, if a function is locally constant, its gradient vector (i.e., the vector of partial derivatives) is zero and such a function could be (locally) represented and absorbed to the MLP bias. Hence, the larger the mean sum of the absolute values of the local partial derivatives with respect to an input variable, the more important that input variable is for representing the variability of an unknown function approximated by the MLP. Hence, the descending order of MAS-values defines the ranking of input variables over one run of Algorithm 3. Actually the method described by Dimopoulos et al. (1995) starts with the similar analytic formula (formula (3)) as in (7), but (7) is a generalization because our MLP model contains the bias nodes in order to always guarantee unbiased regression estimate for the training data in Algorithm 3. Moreover, as with clustering, we compute the overall input-output sensitivity formulae using the robust *mean absolute error* instead of sum-of-squares that was proposed by Dimopoulos et al. (1995), which nonuniformly concentrates on large deviations from zero (see Kärkkäinen and Heikkola 2004).

The whole algorithm for deriving the MLP input sensitivity is given in Algorithm 4. To this end, there are many points in this algorithm which may produce variability to the final result, i.e. the ranking. With different runs, different foldings appear in cross-validation and different local initializations are tested when seeking for the values of the metaparameters n_1 and β. Hence, it typically happens that different final network is encountered from repetitions of Algorithm 3 whose ranking (1–12, where 1 represents the most significant) is then determined using Algorithm 4. To assess the stability and soundness of this result, we repeat Algorithm 4 five times, store the rankings obtained by different runs, and, then, compute the classical Fleiss' kappa κ (Fleiss, 1971), which precisely quantifies the reliability of agreement between a fixed number of MLP network raters. The actual variable rating is then based on the ascending order of the sum of rankings from these five repetitions (between 5–60, where 5 means that such a variable was declared as the most significant for all the repetitions).

5.2. Predictive results and their analysis

As input data for MLP, we use the same set as in the cluster analysis, i.e. the grades of the students who have completed at least half of the core courses, see Table 3. Moreover, the missing values (29.65% altogether) are again completed by using the hot-deck imputation with 8 prototypes (see Section 4.3 for more thorough description). As output data, for each student considered, we use (i) mean grade and (ii) mean number of credits per semester, individually.

Algorithm 4: Input sensitivity ranking.

```
Input: Data $(\mathbf{X}, \mathbf{Y}) = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ of inputs and desired outputs.
Output: Ranked list of MLP input variables.
1: Fix $\vec{\beta}$ and $n1max$, and apply Algorithm 3 to obtain $\mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)$;
2: Compute MAS of $\mathcal{N}(\mathbf{W}^1, \mathbf{W}^2)$ according to formula (9);
3: Order input variables in descending order with respect to MAS to establish ranking;
```

Results of the predictive analysis process, as described above, are provided in Table 8. There, for each course, the "RSum" provides the sum of rankings (1–12) of five individual runs of Algorithm 4. Moreover, in order to asses the stability of the final ranking, we have tested 3-fold, 7-fold, and 10-fold stratified cross-validation. As labels for the 3-fold stratification, we used the three cluster indices that were obtained in the previous section for $K = 3$ (the analyzed result). For the 7-fold CV, the labels corresponded to the amount of completed courses in Table 3, i.e., to the separate groups of students for $q = 6, \ldots, 12$, whose sizes are given by n_q. In the third stratified cross-validation strategy with 10 folds, we used the labels that were obtained when clustering the students into 8 clusters (same as in imputation).

Hence, the strategy to create the different number of stratified folds was completely different, but the final rankings of the 7-fold and 10-fold CV were exactly the same and there was only one very small difference compared to 3-CV: for the mean grade, rankings of the *PCtools* and *Datanet* courses were swapped. We conclude that there is high reliability concerning the final rankings, because Fleiss κ show *moderate agreement* for grades with 7 and 10 folds and the rest of the cases witness *substantial agreement* between the ratings of the individual runs of Algorithm 4. From "MeanError" (see Table 8), which represent the mean of the prediction error (6) over the five runs, we conclude that mean grades can be predicted (in the generalization sense as explained above) with about twice as accurately as the mean number of credits semesterwise. Again, this illustrates the higher and more random individual variability of the number of credits obtained per semester compared to the level of grades (see also Figures 6 and 7).

Based on the results presented in Table 8, we draw the following main conclusions: Comparably to the correlation and clustering analysis results, also based on the predictive MLP input sensitivity analysis, the courses *Datanetworks* and *Computer Structure and Architecture* seem to be most influential to the overall performance in the studies. For the performance in grades, also the course *Object Oriented Analysis and Design* pops up, and, for the overall credits, the largest course *Programming 2* shows - like in the previous analyses - high significance.

For some course, like *Computer and Datanetworks as Tools* and *Programming of Graphical User Interfaces*, there is a large difference of the ranks between mean grades and mean credits, which was not addressed so strongly by the other two EDM techniques. One reason for this might be the varying number of students passing a course, which is reflected in the predictive analysis as higher need of imputation. As can be seen from Figure 2, much less students have passed these two courses compared to the other courses[5].

The obtained predictions and the prediction errors for grades and credits, studentwise, are illustrated in Figures 6 and 7. In the figures, the x-axis corresponds to a student index, where

[5]Actually, also the *Research Methods in Computing* has been passed by less students but this course has already been found to be less influential and the most different to the courses (see especially the discussion in Section 4.3).

Table 8: Input rankings for the three foldings.

	3-fold CV				7-fold CV				10-fold CV			
	grades		credits		grades		credits		grades		credits	
MeanError	6.44e-3		1.22e-2		6.37e-3		1.22e-2		6.36e-3		1.22e-2	
Fleiss κ	0.76		0.72		0.49		0.62		0.52		0.78	
Course	RSum	*rank*	RSum	*rank*	RSum	*rank*	RSum	*rank*	RSum	*rank*	RSum	*rank*
PCtools	19	4	49	10	16	3	47	10	15	3	50	10
Datanet	17	3	10	2	19	4	10	2	18	4	10	2
OOA&D	10	2	39	7	12	2	40	7	13	2	39	7
Alg1	30	6	20	4	31	6	20	4	31	6	20	4
IntroSE	36	7	42	9	36	7	42	9	36	7	41	9
OpSys	39	8	57	11	41	8	57	11	41	8	57	11
DB&DMgm	45	9	15	3	42	9	15	3	42	9	15	3
Prog1	60	12	30	6	59	12	32	6	59	12	30	6
Prog2	50	10	5	1	50	10	5	1	50	10	5	1
CompArc	5	1	25	5	6	1	25	5	6	1	25	5
GUIprog	24	5	58	12	22	5	58	12	23	5	58	12
CompRes	55	11	40	8	56	11	41	8	56	11	40	8

the students are taken in the ascending order with respect to missing courses, i.e., the larger the index the more of core course grades are missing, and have been imputed in the MLP training data. With this respect, the accuracy on the mean number of credits semesterwise shows large increase at the end. As can be seen from Figure 6, the grades of the core courses predict, with reasonable accuracy, the overall mean grade level of a student. We think that this is a promising result, especially when the number of studies related to the analyzed core courses is typically less than half of the total number of credits, see Table 4.

On the contrary, the generalization accuracy of the average number of credits semesterwise is very bad (see Figure 7), and the last students, i.e., those with the most missing values, are the most erroneous. Hence, we do not recommend the final network for actual prediction but, as explained, it is considered suitable for the sensitivity analysis. The difference between accurate prediction and stable detection of input relevance is also clearly captured in Table 8 as explained above: The rankings in the repeated attempts in Table 8 are very stable, as shown by the Fleiss κ's, even if the prediction accuracy can be very poor as shown in Figure 6 and 7.

Hornik et al. (1989) summarize the essence of MLP training: ”We have thus established that such 'mapping' networks are universal approximators. This implies that any lack of success in applications must arise from inadequate learning, insufficient numbers of hidden units or the lack of a deterministic relationship between input and target.” The proposed training approach here tries to manage all these issues in order to end up with *the most reliably generalizing MLP network*. Hence, we try to capture the deterministic behavior within the data and use this to compute the input relevance. Stability of the results as witnessed in Table 8, with substantial within-method triangulation, supports the conclusion that this was, indeed, obtained here.

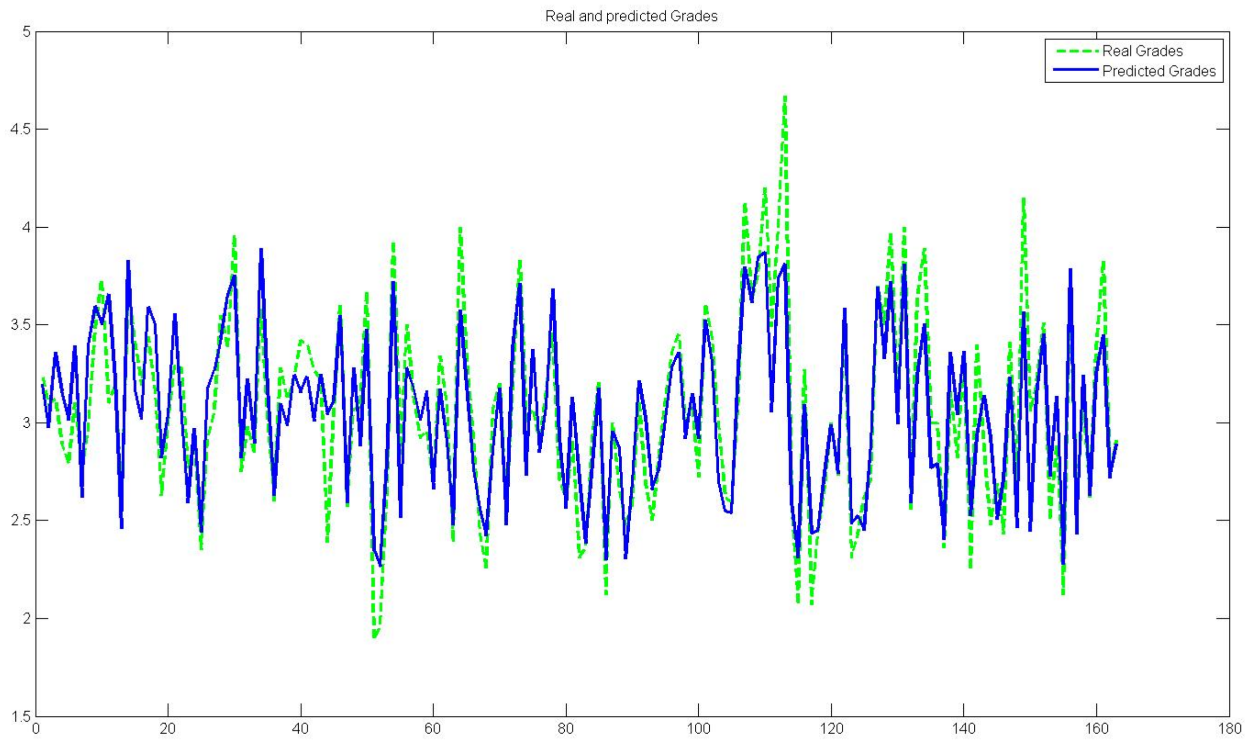


Figure 6: Prediction of mean grades: real (green color) and predicted (blue color) values.

6. Conclusions

This paper presents methods for detecting main courses which determine the general success in CS oriented studies. We employed techniques from the three main categories of educational data mining, partly working in relations to the remaining two categories as assistance in individual analyses. Moreover, we showed how to cope with the nonstructured sparsity pattern in data, using both available data strategy and prototype-based imputation. In Table 9, all the analysis results are summarized. We can conclude the study from the educational domain level point of view and from the methodological point of view.

From the domain level point of view and based on Table 9, we conclude that the quality of studies is already very much determined by the first introductory courses, *Datanetworks* and *Computer Structure and Architecture*, offered in the first year of study. Both of these courses test more the general capability of a student to study than the actual knowledge of professional CS skills. Though they have technical topics, they are taught on a conceptual level and, especially in comparison with the third introductory course (see Table 2), they are completed by a final examination at the end of the course. Therefore, these courses test how well the student is able to learn, understand and explain concepts instead of testing specific (IT) skills. When it comes to credits/timely graduation, the success of a student is also determined by sedulousness and perseverance: The *Programming 2* which is also creditwise the largest course (see Table 2), is strongly related to the number of credits that a student can achieve in general by hard work-

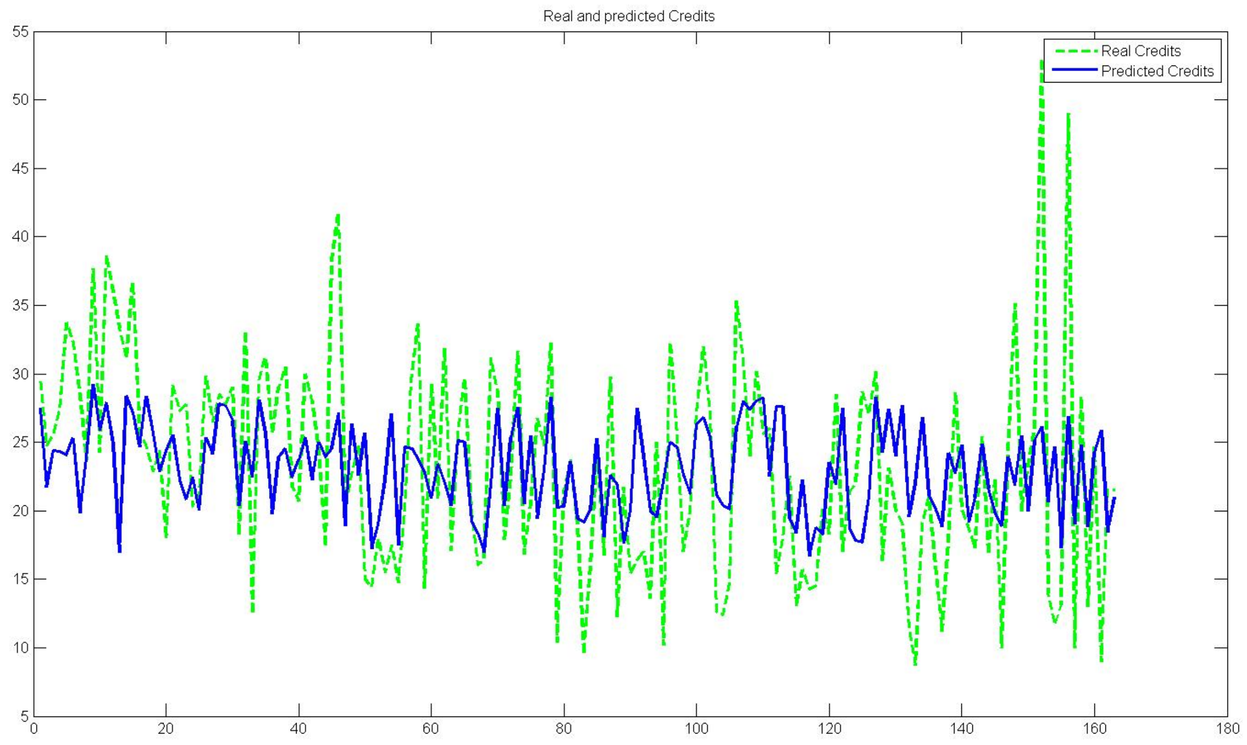


Figure 7: Prediction of mean credits: real (green color) and predicted (blue color) values.

ing. Hence, for the overall performance, general study capabilities are more important than the occupational skills and students can succeed in CS studies with diligent and goal-oriented study behaviour without being the most skilled programmers with mathematical talent. This is important knowledge which should be communicated to the students in the beginning of their studies.

Naturally, our conclusions from the the organizational level as such are not generalizable to other institutions as educational data and the subsequent knowledge on particular courses are different. From the methodological perspective, on the other hand, both the overall approach and the individual methods with their varying but argumented details are general and can be applied to analyze sparse data of student performance. Note that if a snapshot of a study registry of an arbitrary educational institution would be taken, there would be missing values similarly to our case for the nonfinished courses.

In general, on the methodological level, the combination of within-method and between-method triangulation provided very solid results concerning the overall effects and impact of the analyzed courses. In order to deal with our student data it was necessary to augment the existing methods and approaches to work with the sparse data. What about soundness of the algorithms and the overall analysis presented here? There is lot of novelty in the procedures applied. The prototype-based clustering approach with available data spatial median as statistical estimate is not a standard data mining technique. It has been developed in the earlier work of the research

Table 9: Summary of the results

	grades				credits			
course	M1	M2	M3	sum (rank)	M1	M2	M3	sum (rank)
Computer and Datanetworks as Tools	11	9	3	23 (8)	12	9	10	31 (12)
Datanetworks	3	1	4	8 (2)	1	1	2	4 (1)
Object Oriented Analysis and Design	6	6	2	14 (4)	11	6	7	24 (7)
Algorithms 1	1	2	6	9 (3)	4	2	4	10 (3)
Introduction to Software Engineering	10	10	7	27 (10)	9	10	9	28 (10)
Operating Systems	8	7	8	23 (9)	8	7	11	26 (9)
Basics of Databases and Data Management	5	5	9	19 (6)	5	5	3	13 (5)
Programming 1	9	11	12	32 (11)	7	11	6	24 (6)
Programming 2	4	4	10	18 (5)	2	4	1	7 (2)
Computer Structure and Architecture	2	3	1	6 (1)	3	3	5	11 (4)
Programming of Graphical User Interfaces	7	8	5	20 (7)	6	8	12	26 (8)
Research Methods in Computing	12	12	11	35 (12)	10	12	8	30 (11)

group (Äyrämö, 2006; Kärkkäinen and Äyrämö, 2005; Kärkkäinen and Äyrämö, 2004) and its application is based on our own implementation throughout. Similarly, the way how the clustering algorithm is constructively initialized and how the variable ranking of prototypes is derived are not standard choices in cluster analysis. Moreover, the whole computational process for the predictive analysis - use of MLP with a) hot-deck imputation, b) complexity-aware training for best generalization, c) analytic formula based robust input sensitivity derivation, d) sensitivity ranking, e) Fleiss κ as stability measure for rankings - is completely novel. It is also based on an own implementation throughout. The training phase b) has been recently proposed and tested by Kärkkäinen (2014) and Kärkkäinen et al. (2014).

The underlying principle to study soundness in all the treatments here has been based on local and global triangulation: In the correlation analysis, significancy was computed with and without Bonferroni Correction. In cluster analysis, variable ranking was computed in two ways and assessed using nonparametric Kruskal-Wallis test. Similarly, in the predictive analysis three different foldings (number of folds and how they are created) were used and Fleiss κ was then applied to the results of five iterations of the overall algorithm to study its stability. Hence, locally (for each method separately) we have made serious and versatile attempts to vary the metaparametrization of the approaches and also reported all the results. Globally, on the whole analysis level, we have again based our overall conclusions on the results and conclusions of the three methods of different orientations in EDM. We reason that such a two-level treatment, where both locally and globally the same results and their interpretation are supported by different approaches, improves the technical soundness of the study. Furthermore, the way to obtain the final ranking, both in clustering and in the MLP analysis, is novel and establishes a practical framework which can be used in similar applications.

Acknowledgement

The support of this research by the Department of Mathematical Information Technology, University of Jyväskylä, Finland, is much appreciated. The unknown reviewers are gratefully acknowledged; their suggestions and comments led to significant improvements of the original manuscript. Finally, we thank Sami Äyrämö for enlightening discussions on the clustering.

References

Aldahdooh, R. T. and Ashour, W. 2013. Dimk-means distance-based initialization method for k-means clustering algorithm. *International Journal of Intelligent Systems and Applications (IJISA) 5,* 2, 41.

Apostol, T. M. 1969. *Calculus, Volume 2: Multi-variable Calculus and Linear Algebra with Applications to Differential Equations and Probability*. Wiley.

Äyrämö, S. 2006. *Knowledge Mining Using Robust Clustering*. Jyväskylä Studies in Computing, vol. 63. University of Jyväskylä.

Bai, L., Liang, J., and Dang, C. 2011. An initialization method to simultaneously find initial cluster centers and the number of clusters for clustering categorical data. *Knowledge-Based Systems 24,* 6, 785–795.

Bai, L., Liang, J., Dang, C., and Cao, F. 2012. A cluster centers initialization method for clustering categorical data. *Expert Systems with Applications 39,* 9, 8022–8029.

Baker, R. et al. 2010. Data mining for education. *International Encyclopedia of Education 7*, 112–118.

Baker, R. S. and Yacef, K. 2009. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining 1,* 1, 3–17.

Bartlett, P. L. 1998. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *Information Theory, IEEE Transactions on 44,* 2, 525–536.

Batista, G. and Monard, M. C. 2003. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence 17*, 519–533.

Bayer, J., Bydzovská, H., Géryk, J., Obšivac, T., and Popelinskỳ, L. 2012. Predicting dropout from social behaviour of students. In *Proceedings of the 5th International Conference on Educational Data Mining-EDM 2012*. 103–109.

Bhardwaj, B. and Pal, S. 2011. Mining educational data to analyze students' performance. *(IJCSIS) International Journal of Computer Science and Information Security, 9,* 4.

Bouchet, F., Kinnebrew, J. S., Biswas, G., and Azevedo, R. 2012. Identifying students' characteristic learning behaviors in an intelligent tutoring system fostering self-regulated learning. In *EDM*. 65–72.

Bradley, P. and Fayyad, U. 1998. Refining initial points for k-means clustering. In *ICML*. Vol. 98. 91–99.

Bryman, A. 2003. Triangulation. *The Sage encyclopedia of social science research methods. Thousand Oaks, CA: Sage*.

Calders, T. and Pechenizkiy, M. 2012. Introduction to the special section on educational data mining. *ACM SIGKDD Explorations Newsletter 13,* 2, 3–6.

Campagni, R., Merlini, D., and Sprugnoli, R. 2012. Analyzing paths in a student database. In *The 5th International Conference on Educational Data Mining*. 208–209.

Carlson, R., Genin, K., Rau, M., and Scheines, R. 2013. Student profiling from tutoring system log data: When do multiple graphical representations matter? In *Proceedings of the 6th International Conference on Educational Data Mining*.

Chandra, E. and Nandhini, K. 2010. Knowledge mining from student data. *European Journal of Scientific Research 47,* 1, 156–163.

CHEN, L., CHEN, L., JIANG, Q., WANG, B., AND SHI, L. 2009. An initialization method for clustering high-dimensional data. In *Database Technology and Applications, 2009 First International Workshop on*. IEEE, 444–447.

CROUX, C., DEHON, C., AND YADINE, A. 2010. The k-step spatial sign covariance matrix. *Adv Data Anal Classif 4*, 137–150.

DENZIN, N. 1970. Strategies of multiple triangulation. *The research act in sociology: A theoretical introduction to sociological method*, 297–313.

DIMOPOULOS, Y., BOURRET, P., AND LEK, S. 1995. Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Processing Letters 2,* 6, 1–4.

EMRE CELEBI, M., KINGRAVI, H. A., AND VELA, P. A. 2012. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*.

ERDOGAN, S. AND TYMOR, M. 2005. A data mining application in a student database. *Journal Of Aeronautics and Space Technologies 2*.

FAYYAD, U., PIATESKY-SHAPIRO, G., AND P., S. 1996. Extracting useful knowledge from volumes of data. *Communications of the ACM 39,* 11, pp. 27–34.

FLEISS, J. L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin 76,* 5, 378–382.

GEVREY, M., DIMOPAULOS, I., AND LEK, S. 2003. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling 160*, 249–264.

HAGAN, M. T. AND MENHAJ, M. B. 1994. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. Neural Networks 5*, 989–993.

HALONEN, P. 2012. Tietotekniikan laitos. 2. TIETOTEKNIIKKA 12a- valintasyyt- opetuksen laatumielipiteet.pdf.

HAN, J., KAMBER, M., AND TUNG, A. 2001. Spatial clustering methods in data mining: A survey, h. miller and j. han (eds.), geographic data mining and knowledge discovery.

HARDEN, T. AND TERVO, M. 2012. Informaatioteknologian tiedekunta. 1. ITK 4- opinnoista suoriutuminen.pdf.

HARPSTEAD, E., MACLELLAN, C. J., KOEDINGER, K. R., ALEVEN, V., DOW, S. P., AND MYERS, B. A. 2013. Investigating the solution space of an open-ended educational game using conceptual feature extraction. In *EDM2013*.

HAWKINS, W., HEFFERNAN, N., WANG, Y., AND BAKER, R. S. 2013. Extending the assistance model: Analyzing the use of assistance over time. In *EDM2013*.

HETTMANSPERGER, T. P. AND MCKEAN, J. W. 1998. *Robust nonparametric statistical methods*. Edward Arnold, London.

HOLLANDER, M., WOLFE, D. A., AND CHICKEN, E. 2013. *Nonparametric statistical methods*. Vol. 751. John Wiley & Sons.

HORNIK, K., STINCHCOMBE, M., AND WHITE, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks 2*, 359–366.

HUANG, G. B. 2003. Learning capability and storage capacity of two-hidden-layer feedforward networks. *Neural Networks, IEEE Transactions on 14,* 2, 274–281.

HUBER, P. J. 1981. *Robust Statistics*. John Wiley & Sons Inc., New York.

JAIN, A. K. 2010. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters 31,* 8, 651–666.

JERKINS, J. A., STENGER, C. L., STOVALL, J., AND JENKINS, J. T. 2013. Establishing the impact of a computer science/mathematics anti-symbiotic stereotype in cs students. *J. Comput. Sci. Coll. 28,* 5 (May), 47–53.

JICK, T. D. 1979. Mixing qualitative and quantitative methods: Triangulation in action. *Administrative science quarterly 24,* 4, 602–611.

JOHN, G. H., KOHAVI, R., AND PFLEGER, K. 1994. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning*. 121–129.

KÄRKKÄINEN, T. 2002. MLP in layer-wise form with applications in weight decay. *Neural Computation 14*, 1451–1480.

KÄRKKÄINEN, T. 2014. Feedforward network - with or without an adaptive hidden layer. *IEEE Transactions on Neural Networks and Learning Systems*. in revision.

KÄRKKÄINEN, T. AND ÄYRÄMÖ, S. 2004. Robust clustering methods for incomplete and erroneous data. In *Proceedings of the Fifth Conference on Data Mining*. WIT Press, 101–112.

KÄRKKÄINEN, T. AND ÄYRÄMÖ, S. 2005. On computation of spatial median for robust data mining. *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, EUROGEN, Munich.*

KÄRKKÄINEN, T. AND HEIKKOLA, E. 2004. Robust formulations for training multilayer perceptrons. *Neural Computation 16*, 837–862.

KÄRKKÄINEN, T., MASLOV, A., AND WARTIAINEN, P. 2014. Region of interest detection using MLP. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2014*. 213–218.

KÄRKKÄINEN, T. AND TOIVANEN, J. 2001. Building blocks for odd–even multigrid with applications to reduced systems. *Journal of computational and applied mathematics 131,* 1, 15–33.

KERR, D. AND CHUNG, G. 2012. Identifying key features of student performance in educational video games and simulations through cluster analysis. *Journal of Educational Data Mining 4,* 1, 144–182.

KHAN, S. S. AND AHMAD, A. 2013. Cluster center initialization algorithm for k-modes clustering. *Expert Systems with Applications*.

KINNUNEN, P., MARTTILA-KONTIO, M., AND PESONEN, E. 2013. Getting to know computer science freshmen. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*. Koli Calling '13. ACM, New York, NY, USA, 59–66.

KOHAVI, R. 1995. Study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*. 1137–1143.

KOHAVI, R. AND JOHN, G. H. 1997. Wrappers for feature subset selection. *Artificial Intelligence 97*, 273–324.

KOTSIANTIS, S. 2012. Use of machine learning techniques for educational proposes: a decision support system for forecasting students grades. *Artificial Intelligence Review 37,* 4, 331–344.

MEILĂ, M. AND HECKERMAN, D. 1998. An experimental comparison of several clustering and initialization methods. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 386–395.

MENDEZ, G., BUSKIRK, T., LOHR, S., AND HAAG, S. 2008. Factors associated with persistence in science and engineering majors: An exploratory study using classification trees and random forests. *Journal of Engineering Education 97,* 1.

PINKUS, A. 1999. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 143–195.

RICE, W. R. 1989. Analyzing tables of statistical tests. *Evolution 43,* 1, 223–225.

ROUSSEEUW, P. J. AND LEROY, A. M. 1987. *Robust regression and outlier detection*. John Wiley & Sons Inc., New York.

RUBIN, D. B. 1976. Inference and missing data. *Biometrika 63,* 3, 581–592.

RUBIN, D. B. AND LITTLE, R. J. 2002. Statistical analysis with missing data. *Hoboken, NJ: J Wiley & Sons*.

RUCK, D. W., ROGERS, S. K., AND KABRISKY, M. 1990. Feature selection using a multilayer perceptron. *Neural Network Computing 2,* 2, 40–48.

SAARELA, M. AND KÄRKKÄINEN, T. 2014. Discovering gender-specific knowledge from finnish basic education using pisa scale indices. In *Proceedings of the 7th International Conference on Educational Data Mining*. 60–68.

SAHAMI, M., DANYLUK, A., FINCHER, S., FISHER, K., GROSSMAN, D., HAWTHRONE, E., KATZ, R., LEBLANC, R., REED, D., ROACH, S., CUADROS-VARGAS, E., DODGE, R., KUMAR, A., ROBINSON, B., SEKER, R., AND THOMPSON, A. 2013a. Computer science curricula 2013.

SAHAMI, M., ROACH, S., CUADROS-VARGAS, E., AND LEBLANC, R. 2013b. Acm/ieee-cs computer science curriculum 2013: Reviewing the ironman report. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. SIGCSE '13. ACM, New York, NY, USA, 13–14.

SAN PEDRO, M. O. Z., BAKER, R. S., BOWERS, A. J., AND HEFFERNAN, N. T. 2013. Predicting college enrollment from student interaction with an intelligent tutoring system in middle school. In *Proceedings of the 6th International Conference on Educational Data Mining*. 177–184.

SHOJAEEFARD, M. H., AKBARI, M., TAHANI, M., AND FARHANI, F. 2013. Sensitivity analysis of the artificial neural network outputs in friction stir lap joining of aluminum to brass. *Advances in Material Science and Engineering 2013*, 1–7.

SPRINGER, A., JOHNSON, M., EAGLE, M., AND BARNES, T. 2013. Using sequential pattern mining to increase graph comprehension in intelligent tutoring system student data. In *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, 732–732.

STEINBACH, M., ERTÖZ, L., AND KUMAR, V. 2004. The challenges of clustering high dimensional data. In *New Directions in Statistical Physics*. Springer, 273–309.

TAMURA, S. AND TATEISHI, M. 1997. Capabilities of a four-layered feedforward neural network: Four layers versus three. *IEEE Transactions on Neural Networks 8,* 2, 251–255.

VALSAMIDIS, S., KONTOGIANNIS, S., KAZANIDIS, I., THEODOSIOU, T., AND KARAKOS, A. 2012. A clustering methodology of web log data for learning management systems. *Educational Technology & Society 15,* 2, 154–167.

VAN DE SANDE, B. 2013. Applying three models of learning to individual student log data. *Under review*.

VIHAVAINEN, A., LUUKKAINEN, M., AND KURHILA, J. 2013. Using students programming behavior to predict success in an introductory mathematics course. *Under review*.

XU, R. AND WUNSCH, D. C. 2005. Survey of clustering algorithms. *IEEE Transactions on Neural Networks 16,* 3, 645–678.

ZHONG, C., MIAO, D., WANG, R., AND ZHOU, X. 2008. Divfrp: An automatic divisive hierarchical clustering method based on the furthest reference points. *Pattern Recognition Letters 29,* 16, 2067–2077.

Analyzing Process Data from Game/Scenario-Based Tasks: An Edit Distance Approach

Jiangang Hao
Center for Advanced Psychometrics
Educational Testing Service
Princeton, NJ 08541, U.S.A.

Zhan Shu
Educational Testing Service
Princeton, NJ 08541, U.S.A.

Alina von Davier
Center for Advanced Psychometrics
Educational Testing Service
Princeton, NJ 08541, U.S.A.

January 6, 2015

Students' activities in game/scenario-based tasks (G/SBTs) can be characterized by a sequence of time-stamped actions of different types with different attributes. For a subset of G/SBTs in which only the order of the actions is of great interest, the process data can be well characterized as a string of characters (i.e., action string) if we encode each action name as a single character. In this article, we report our work on evaluating students' performances by comparing how far their action strings are from the action string that corresponds to the best performance, where the proximity is quantified by the edit distance between the strings. Specifically, we choose the Levenshtein distance, which is defined as the minimum number of insertions, deletions, and replacements needed to convert one character string into another. Our results show a strong correlation between the edit distances and the scores obtained from the scoring rubrics of the pump repair task from the National Assessment of Education Progress Technology and Engineering Literacy assessments, implying that the edit distance to the best performance sequence can be considered as a new feature variable that encodes information about students' proficiency, which sheds light on the value of data-driven scoring rules for test and task development and for refining the scoring rubrics.

1. Introduction

Innovations in educational assessments have been accelerated by the advance of technology over the past decade. The objective of educational assessments is to collect and make sense of information about what students know and can do and to evaluate their progress or shape their future learning experience (Mislevy et al., 2014). The more students are engaged and put effort into the assessments, the better will be the information collected about them (Schmit and Ryan, 1992; Sundre and Wise, 2003). Gee (2007) has pointed out that video games or simulations have the capability to increase students' engagement and to create better conditions that boost learning. From the assessment perspective, game/scenario-based tasks (G/SBTs) promise to offer a sweet spot in the assessment design space for some purposes and some circumstances (Mislevy et al., 2014). Moreover, G/SBTs can provide students with new opportunities to demonstrate proficiencies in complex interactive environments that traditional assessment formats cannot afford (Klopfer et al., 2009).

Given these nice features, G/SBTs are considered one of the major future directions of educational assessment and have received considerable attention over the past decade. In practice, G/SBTs have been widely used in real assessments ranging from low- to medium-stakes tests, such as the Technology and Engineering Literacy assessments (TEL) from the National Assessment of Educational Progress (NAEP) (TEL, 2013), to high-stakes tests, such as the innovative assessments in the U.S. Medical Licensure Examinations (USMLE, 2014).

For traditional item formats, such as multiple choice (MC) or constructed response (CR), the scoring rubrics are generally straightforward to develop and implement. But for G/SBTs, developing the scoring rubrics itself becomes much more complicated, and it is no longer a trivial task to implement the scoring rubrics too. The complications come from at least two sources. First, owing to the increased complexity provided by the interactive environment of G/SBTs, developing operational scoring rubrics requires more iterations with the help of proper data-mining techniques to uncover meaningful features from students' activities. G/SBTs are generally developed by following an evidence-centered design (ECD) (Almond et al., 2002; Mislevy and Riconscente, 2006), where several rounds of iterations are implemented to explicate the actions or state information relevant to the targeted construct. However, in practice, it is almost impossible to predict all possible behaviors in the game or simulation ahead of time, which makes data mining the log file a necessary step to uncovering empirical evidence relevant to the construct to be measured. Second, the logistics of handling the log files from the G/SBTs are very challenging. Students' activities are kept in the log files, and parsing through the log file to extract useful information is generally not a trivial task, requiring additional data analysis techniques.

Discovering new features from the logs of specific G/SBTs requires creativity and insight, which are generally difficult to standardize. Therefore, finding new approaches that can be applied to rather generic G/SBTs is highly desirable. There are a lot efforts have been devoted to analyzing the sequential data from educational games/simulations. For example, temporal Bayesian Networks has been used to model students' performance in Orthopedic Surgery Training (Chieu et al., 2010). Various sequence detection techniques are discussed for generating adaptive feedback in mathematical generalisation (Gutierrez-Santos et al., 2010). Cluster analysis has been used to analyze action sequences in personalized e-learning (Köck and Paramythis, 2011), scenario based assessment (Bergner et al., 2014) and systematic inquiry behaviors (Sao Pedro et al., 2013). In addition to directly cluster the event sequence, the clustering of activity state sequence with interval sampling technique has been studied (Desmarais and Lemieux, 2013). Moreover, a set of tools used for sequence mining have been assembled together into a R package, TraMineR (Gabadinho et al., 2009).

In this article, we propose an edit distance–based approach to extract useful information from the logs of certain types of G/SBTs, where we know what the best practices are.[1] In this approach, students' performances are measured by how far or close their action sequences are from the action sequences corresponding to the "best" practices. To measure how far or close they are, we propose to use the "action distance," which is the (weighted) number of certain operations needed to bring one action sequence to another (usually, the one corresponding to the best practice). If we dummy code each action name as a single character and choose the operations as some text-editing operations, such as insertion, deletion, and substitution, the action distance is exactly the well-known edit distance that originally emerged in natural language processing (NLP).

To demonstrate the usefulness of this approach, we apply it to a specific task, the pump repair task from the NAEP TEL (PumpRepair, 2013). In our analysis, we dummy code the action names with lowercase letters and turn each student's response into a character string. Then, to compare the differences between the strings, we choose one of the most widely used edit distances, the Levenshtein distance (Levenshtein, 1966), which is defined as the minimum number of deletions, insertions, and substitutions that will transform one string into another. Our analysis shows that though such a measure is obtained from a very different perspective, it correlates significantly with the scores obtained from the scoring rubrics. This provides a new perspective from which we can quantify students' performances in G/SBTs and that can be readily applied to a number of similar G/SBTs where only the order of the actions is of primary interest.

It is worth noting that the approach presented in this article, like other data-mining approaches, should not be considered as a replacement for rubric-based scoring. Rather, data-mining approaches should be considered independent checks for scoring rubrics, useful in flagging potential problems. Direct score reporting based on data-mining approaches can only be done after results are properly interpreted.

The article is organized as follows. In section 2, we introduce edit distance, the basic algorithms, and their applications to G/SBAs. In section 3, we apply the edit distance approach to a specific simulation-based task, the pump repair task from NAEP, and show the results. In section 4, we discuss the applicabilities and limitations of the edit distance approach.

2. EDIT DISTANCE

2.1. FROM ACTION DISTANCE TO EDIT DISTANCE

A student's response to the G/SBTs forms a sequence of time-stamped actions, which we refer to as an action sequence. For certain types of G/SBTs, we know what action sequences correspond to the best performance based on the design rubrics, which we refer to as ideal action sequences. Therefore, we can introduce an action distance that is a measure of how far the action sequence from a response is to the ideal action sequence. The action distance reduces to the edit distance if we dummy code each action name as a single character, ignoring the temporal component corresponding to the time stamp of each action and restricting the operations only to editing operations such as insertion, deletion, and substitution. In this article, we focus on a subset

[1]Note that, in many cases, finding the best practice is one of the important goals of data mining. Here we assume we know this, which is true for a subset of G/SBTs.

of the G/SBTs where the temporal information is not of major concern.[2] Therefore, from now on, we simply use edit distance as a surrogate for action distance. Edit distance is defined as the minimum weights (costs) of the editing operations used to transform one string into another (Jurafsky and Martin, 2000). Under this definition, there can be many possible variants in terms of the types of editing operations (such as insertion, deletion, substitution, transposition) and the associated weight of each operation.

It is worth noting that we are not fully free to use any type of operation and weight. To properly define the edit distance in metric space, some rules (known as metric axioms) must be met (Bryant, 1985). Assume that we have two strings, denoted as $X[1, ...i, ..N]$ and $Y[1, ...j, ...M]$; the additional requirements are depicted by

- $d(X, Y) \geq 0$ if $X \neq Y$: Nonnegativity
- $d(X, Y) = 0$ if $X = Y$: Identity of the indiscernible
- $d(X, Y) = d(Y, X)$: Symmetry
- $d(X, Y) \leq d(X, Z) + d(Z, Y)$: Triangle inequality

where $d(X, Y)$ denotes the edit distance between string X and string Y. These conditions lead to the following requirements for the operations and weights: (1) there is always an inversion operation with equal weight for each editing operation and (2) all weights associated with the edit operations are positive. These requirements significantly reduce the possible space of the variants of the edit distance and are important guidelines for "inventing" new edit distances.

Obviously, the choice of specific edit distance may be "optimized" for a specific purpose by adjusting the types of edit operations and their associated weights. However, for exploratory data analysis, where we generally do not have information about what is the best way to adjust those parameters a priori, it will be sensible to start with the simplest situation, that is, the Levenshtein distance. Throughout the rest of the article, unless noted otherwise, all edit distances we discuss are Levenshtein distances.

2.2. Weighted Levenshtein distance

Among all possible variants of the edit distance, the most widely used is the Levenshtein distance. If we keep the edit operations but allow the weights associated with each operation to be variable, we arrive at the weighted Levenshtein distance (Jurafsky and Martin, 2000). This is a more general situation than the Levenshtein distance, and manipulating the weights gives one an extra knob to "optimize" the edit distance for a specific task. So, in the following, we introduce the algorithm for the weighted Levenshtein distance and hold the Levenshtein distance as a special case where all weights are set to 1. Algorithmically, the computational time for calculating the weighted Levenshtein distance is $O(M * N)$ when realized by dynamic programming (Jurafsky and Martin, 2000). The algorithm is generally specified as two steps. The first step is initialization:

[2]For a subset of G/SBTs, the order of the actions encodes most of the information, and we can ignore the specific temporal information of each action.

$$\begin{aligned} d_{00} &= 0, && \\ d_{i0} &= d_{i-1,0} + w_{del}(X_i), && 1 \le i \le N, \\ d_{0j} &= d_{0,j-1} + w_{ins}(Y_j), && 1 \le j \le M. \end{aligned} \tag{1}$$

Following initialization, a recurrence relation updates the edit distance:

$$d_{ij} = \begin{cases} d_{i-1,j-1}, & X_i = Y_j, \\ \min \begin{cases} d_{i-1,j} + w_{\text{del}}(X_i), \\ d_{i,j-1} + w_{\text{ins}}(Y_j), \\ d_{i-1,j-1} + w_{\text{sub}}(Y_j, X_i). \end{cases} & X_i \neq Y_j, \end{cases} \tag{2}$$

where the functions w_{del}, w_{ins}, and w_{sub} denote the weights of deletion, insertion, and substitution, respectively. Note that the weights are not necessarily constants for all i and j. The edit distance between X and Y is given by d_{NM}. As we mentioned earlier, adjusting the weights provides an extra handle on the resulting edit distance. When there are well-motivated reasons to choose a specific set of weights, one should go with it. For example, the keyboard layout makes certain typos more likely than others. Depending the purpose of the analysis, one can increase or decrease the weights associated with the editing operations relevant to those keys. But in general situations, such as the application discussed in this article, we usually do not have a clear motivation for a specific choice for the weights. From a Bayesian point of view, such weight choices can be considered a priori based on the edit distance.

2.3. Application to G/SBTs

The process data of a student's response to a G/SBT can be summarized as a sequence of time-stamped actions with corresponding attributes. Depending on the complexity of the specific game or scenario-based task, this sequence of actions can be very different in terms of length and variability. For a subset of G/SBTs, the time stamp plays a minor role, as the student's performance is mainly determined by the order of his or her actions during the task. Then, each student's performance is fully characterized by a sequence of actions whose order encodes the majority of information about the performance. As we show, the edit distance approach is well suited for this subset of G/SBTs.

For certain types of G/SBTs, we know the action sequences corresponding to the best practices, that is, the ideal action sequences. The ideal action sequences can be a single sequence from the beginning to the end of the task or a set of segmented sequences, each of which corresponds to evidence nuggets that support the proficiency of a certain skill. Conversely, the action sequences from students' actual responses can be very diverse, some of them following closely or being exactly the same as the ideal action sequences, whereas others may be very different. Therefore, how close the actual response is to the ideal action sequence provides a measure of how good the student's performance is, if one can appropriately specify the metric for "closeness." We suggest that edit distance, specifically the Levenshtein distance, can be considered a promising candidate for this purpose.

Now, the most challenging question comes. How do we quantify that the edit distance does its job properly? Is there a method that allows us to test whether the edit distance for a specific task really works properly? The answer is yes. What we propose is to compare edit distances calculated based on students' actual responses to the edit distances calculated based on fake responses after randomizing the orders of the actions. If the distribution of edit distances from actual responses is systematically lower than the distribution from random responses, it is a good indication that the edit distance is doing its job decently. If the distribution of the edit distance from actual responses is statistically indistinguishable from the distribution calculated based on random responses, then the edit distance won't provide much useful information. This also gives a way to winnow good ideal action sequences that can be used to calculate the edit distance.

To apply the edit distance as a measure of proximity between actual responses and ideal action sequences, another prerequisite is easily neglected. That is, we need to dummy code action names into single letters or numbers because the edit distance is operating directly on character strings. For the dummy coding, there are at least two different situations. In the first situation, one just codes each action as a unique single letter or number. In the second situation, instead of coding each action as a unique letter or number, one codes a class of actions to the same unique letter or number; that is, the actions can be classified into different groups first, and then the dummy coding is done at the group level. This has important implications in practice, because it will be possible that there are actions that play similar roles in terms of revealing the proficiency of certain constructs, and one may want to treat all these actions the same.

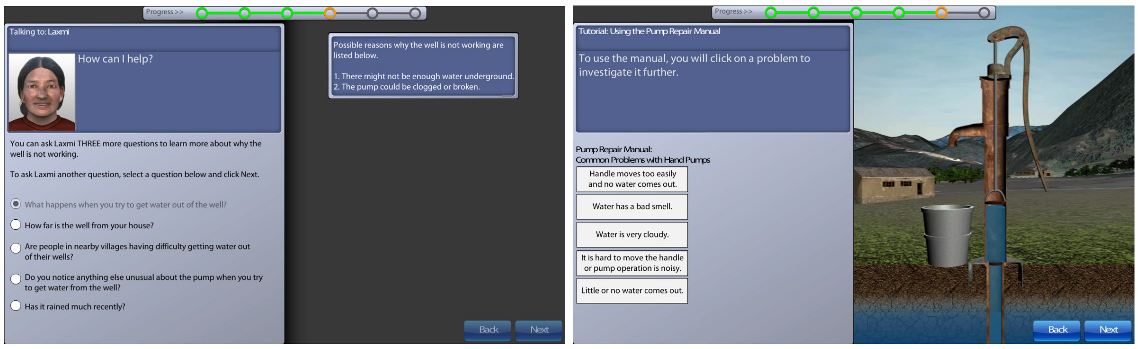
Last, but not least, if a task has multiple ideal action sequences (sequence segments), extra wisdom is needed to decide how to combine the edit distances corresponding to these segments. A confirmatory factor analysis can check an optimal concatenation of the edit distances if there is a clear cognitive motivation. If there is no clear motivation, principal component analysis (PCA) or cluster analysis can guide the combination. In the next section, we demonstrate all these ideas using a specific scenario-based task: the pump repair task from the NAEP TEL.

3. Pump repair task

3.1. The task and scoring rubrics

The TEL (TEL, 2013) is trying to measure whether students can apply what they learn about technology and engineering skills to real-life situations. TEL is implemented via "computer-based and interactive scenario-based tasks to gauge what students know and can do." Among the TEL tasks, the pump repair task was released to the public after the TEL pilot study was conducted in 2013. In the pump repair task, students are asked to play the role of an engineer to troubleshoot a water well that fails to work in a remote village in Nepal. There are two major parts in the pump repair task. In the first part, students can ask a set of questions about why the pump does not work and can get hints about the causes of the problems. In the second part, the students set out to fix the problem. According to the "pump manual," students are told that there are five common problems for the hand pump. The recommended cycle for troubleshooting each problem is check first, followed by repair, and then followed by a test to determine whether it is fixed. In Figure 1, we include two screenshots from the pump repair task, which correspond to the two major phases of the task.

Students' responses are recorded in the following way: the check, repair, and test operations are denoted by C, R, and P, respectively. The five common problems are indexed as 1 through 5. For example, if a student performs the operations of check the fourth common problem followed by repairing the pump for that problem, followed by testing the pump, his or her action sequences



Talking to: Laxmi

How can I help?

You can ask Laxmi THREE more questions to learn more about why the well is not working.

To ask Laxmi another question, select a question below and click Next.

[✓ What happens when you try to get water out of the well?]
How far is the well from your house?
Are people in nearby villages having difficulty getting water out of their wells?
Do you notice anything else unusual about the pump when you try to get water from the well?
Has it rained much recently?

Possible reasons why the well is not working are listed below.

1. There might not be enough water underground.
2. The pump could be clogged or broken.

Tutorial: Using the Pump Repair Manual

To use the manual, you will click on a problem to investigate it further.

Pump Repair Manual:
Common Problems with Hand Pumps

Handle moves too easily and no water comes out.
Water has a bad smell.
Water is very cloudy.
It is hard to move the handle or pump operation is noisy.
Little or no water comes out.

Figure 1: Screenshots of the pump repair task. (left) Questions students can ask the system to get hints about problems with the well system. (right) Troubleshooting part of the task, where students can choose to fix the problems.

Table 1: The scoring rules along the systematicity dimension

Score Level	Details
4	All checks performed before repairs Pump is checked immediately following each repair
3	All checks performed before repairs Pump is not checked immediately following each repair
2	One repair is performed before the associated check (or check is omitted) Pump may not be checked immediately following each repair
1	Two or more repairs performed before the associated check Pump may not be checked immediately following each repair

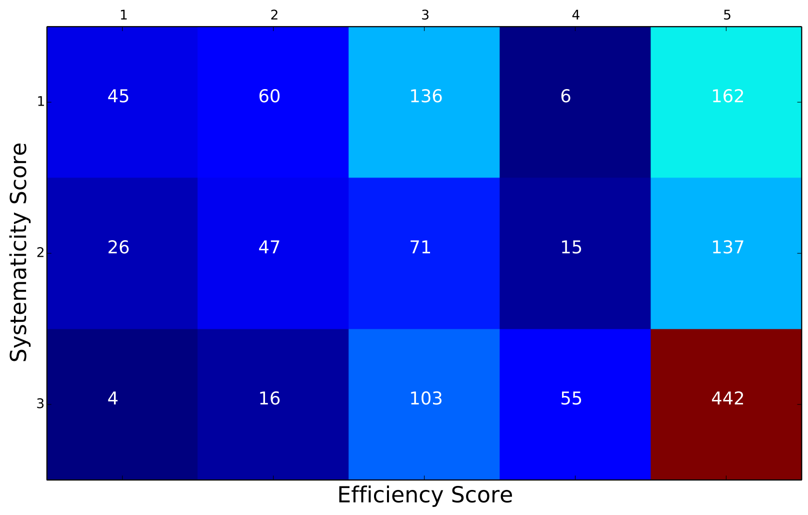
will be recorded as C4R4P. The scoring rubrics used in the pilot study defined two dimensions for the construct probed by this pump repair task: systematicity and efficiency. Systematicity refers to the idea that students need to follow a systematic operation cycle, that is, check, repair, and test, when they troubleshoot the well. Efficiency refers to the idea that students can quickly identify which problems the well actually has and repair them. According to the design of the task, if students are very careful in the first part of the task, they should be able to infer that only problems 4 and 5 are real trouble makers and need to be checked, fixed, and tested.

Each student's response is recorded in the log files as we described earlier. The scoring rubric (TEL, 2013) of the task defines the scoring rules for each of the two dimensions, as follows. For systematicity, the performances are classified into four levels, as shown in Table 1. Conversely, for efficiency, there are five levels, as shown in Table 2.

On the basis of the scoring rubrics, the responses from a total of 1,325 students in the pilot study are scored; a cross table of the score distribution is shown in Figure 2. Looking at the distribution, the two dimensions have low correlation, with a Spearman correlation of $r = 0.347$. Notably, one can observe from Figure 2 that 162 students got the highest level of efficiency score but the lowest level of systematicity score. Conversely, only four students are placed at the highest level of systematicity score but get the lowest efficiency score. This means that many students know where the problems are but cannot fix them systematically, but fewer people who can fix the problem systematically do not know where the problems are.

Table 2: The scoring rules along the efficiency dimension

Action definitions	Efficient actions: E = P, C4, R4, C5, R5 Unnecessary checks: C = C1, C2, C3 Unnecessary repair: R = R1, R2, R3
Score Level	**Details**
5	Only actions from set E
4	Actions from E + 1 action from C
3A 3B	Actions from E + 2–3 actions from C Actions from E + 0–1 action from C + 1 action from R
2	Actions from E + 2–3 actions from C + 1–2 actions from R
1	Actions from E + 3 actions from C + 3 actions from R



Systematicity Score \ Efficiency Score	1	2	3	4	5
1	45	60	136	6	162
2	26	47	71	15	137
3	4	16	103	55	442

Figure 2: Cross table of the efficiency score and systematicity score for the responses from the pilot study. Note that for the systematicity score, no responses get to level 4.

Table 3: Two dummy coding schemes

Action name	C1	C2	C3	C4	C5	R1	R2	R3	R4	R5	P
Coding scheme I	a	b	c	d	e	f	g	h	i	j	k
Coding scheme II	a	a	a	b	b	c	c	c	d	d	e

3.2. DUMMY CODING OF ACTION NAMES

Students' actions are recorded, for example, as "C4, R4, P, C5, R5, P." To apply the edit distance analysis, one needs to turn this into a string with each action recoded as a single letter; that is, one needs to recode the preceding response string by mapping each action or some actions to a single letter. In our implementation, we choose two coding schemes. In coding scheme I, we just convert each action name into a single and unique letter. In coding scheme II, we first classify the actions as necessary actions if they are associated with either 4 or 5 or P. The rest will be classified as unnecessary actions. Then, we code all the unnecessary actions of the same type (e.g., check, repair) to the same letter. The reason for considering coding scheme II is that we want to see whether the edit distance measure will lead to different results if we dummy code the actions in a different but meaningful way. In Table 3, we list the specific mappings of the two schemes.

It is worth noting that the way one chooses to code the action names is somewhat arbitrary in a certain sense. But comparing with random responses for each coding scheme will provide a way to test whether the coding scheme is reasonable.

3.3. EDIT DISTANCE OF THE RESPONSES

In this subsection, we present the results of the edit distances calculated with respect to the action sequences. For the pump repair task, we know the action sequences that correspond to the highest level of skill, for example, C4R5PC5R5P and C5R5PC4R4P. As these two ideal sequences are equally good, we will choose a final edit distance that is the minimum of the edit distances corresponding to each of them. In addition to the ideal action sequences, some other action sequence segments represent certain levels of system thinking, for example, C1R1P, C2R2P, C3R3P, C4R4P, and C5R5P. So we also calculate the edit distances between the actual responses and these action sequence segments. By doing this, we hope to capture all possible information via the edit distances and then explore the space these edit distances span via PCA and cluster analysis. To facilitate the discussion, we introduce the abbreviations corresponding to different edit distances in Table 4.

The first thing we want to check is a comparison of edit distances from actual responses to those from random responses. We create the random responses in the following way: after coding the actual responses into character strings, we build a set of string lengths by counting the length of each character string. Then, we randomly sample the letters based on the coding schemes to form random strings whose lengths are randomly sampled from the set of character string lengths. According to the task design, one must have the test operation (P) done before he or she can submit the results. So, when we sample the letters, we reserve the last letter of each string as that corresponding to action P. Through this process, we generated two sets of random responses corresponding to the two coding schemes. Then, we calculated the edit distances based on these random responses in the same way we did for the actual responses. In

Table 4: Abbreviations for edit distances corresponding to different action sequences

Abbreviation	Action sequence
dist0	C4R4PC5R5P
dist1	C5R5PC4R4P
dist2	C4R4P
dist3	C5R5P
dist4	C1R1P
dist5	C2R2P
dist6	C3R3P
dist	min(dist0, dist1)

Figure 3, we plot the distributions of the edit distances from both actual responses and random responses under the two coding schemes. Looking at these figures, one can observe that only the edit distances "dist" from actual responses are significantly less than those from the random responses; "dist2" and "dist3," from actual responses, are slightly less than those from random responses, whereas for the "dist4," "dist5," and "dist6," there are no clear differences between actual responses and random responses. This suggests that "dist" contains most of the useful information about students' performances.

The second thing we want to check is how well the edit distances are associated with the systematicity and efficiency scores defined in the scoring rubrics. Here we mention association rather than correlation because the edit distances' relation may not be linear, whereas correlation mainly captures the linear association. We calculate both the Spearman correlation and the adjusted mutual information (Vinh et al., 2009) between the scores and the edit distances. The results are presented in Figure 4. On the basis of the results, one can observe that the trends of the associations for the Spearman correlation and adjusted mutual information are very close. To make the discussion more intuitive, we focus on the Spearman correlation in the following discussion. For the Spearman correlation, coding scheme I leads to higher absolute correlations. All edit distances show relatively high correlations with the efficiency score, and the highest correlation is from the "dist," which is about -0.82. For the systematicity score, only "dist" shows a significant correlation of -0.56, while the other edit distances do not correlate to the systematicity score significantly. Similar conclusions held true based on the adjusted mutual information. Given that "dist" contains most of the information, it will be interesting to see the cross tables between "dist" and the efficiency and systematicity scores for the two coding schemes. We present the results for both coding schemes in Figure 5.

By carefully examining all these results, the following conclusions can be drawn. First, the two different coding schemes do not lead to significantly different edit distances in terms of the correlations and adjusted mutual information with the systematicity and efficiency scores, though some variabilities have been introduced. Clearly one can also introduce other coding schemes based on specific motivations and compare the results in a similar fashion. In this article, our goal is to demonstrate the methodology rather than give an exhaustive analysis of various coding schemes under different motivations, so we conclude our discussion along this line. Second, the edit distance to the ideal action sequences encodes the most information about students' performance, as revealed both by correlations and adjusted mutual information with the rubric-based scores and by the distribution comparison with random responses. This in-

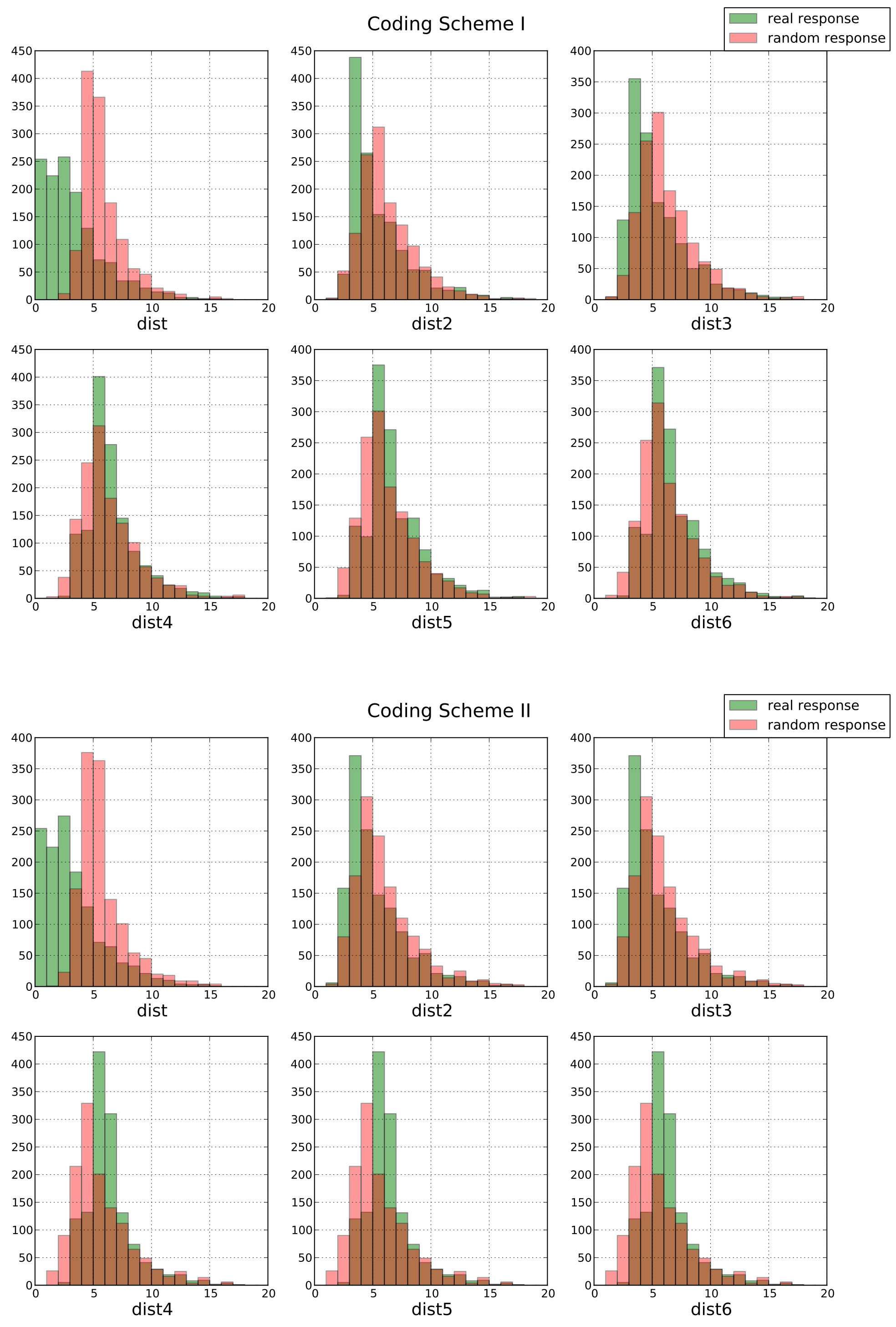
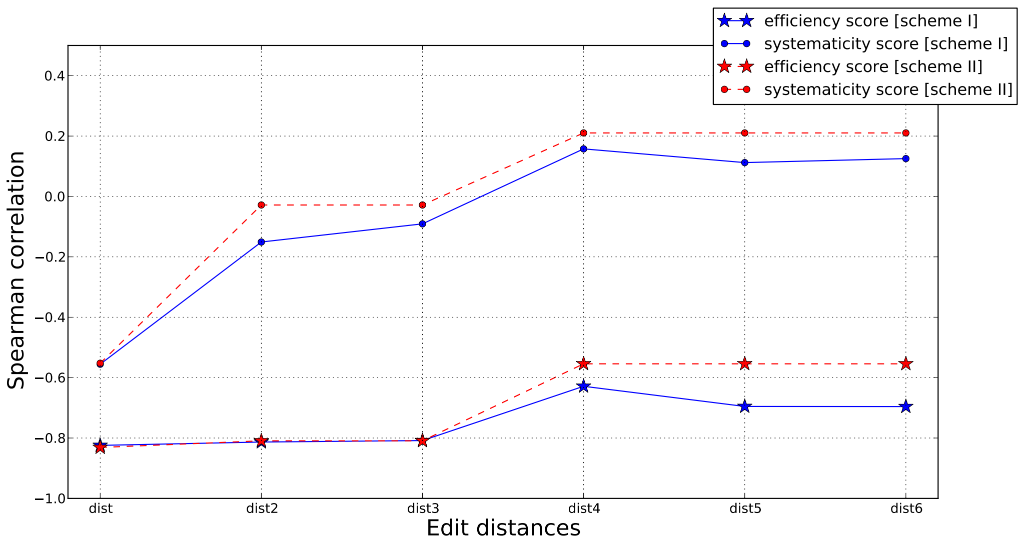


Figure 3: Distributions of the edit distances from actual responses and from random responses for coding schemes I and II. The specific definitions of the labels along the x axis are given in Table 4.



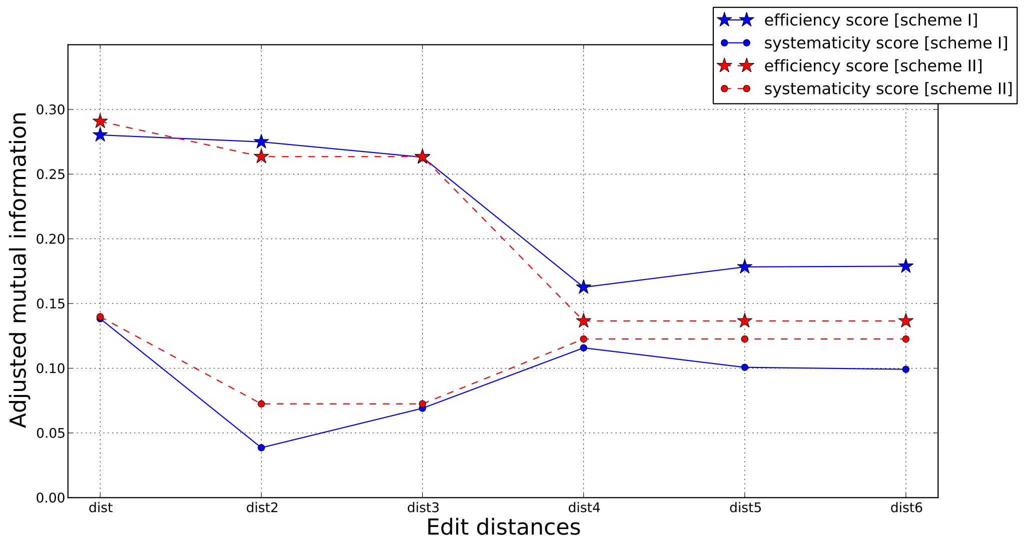


Figure 4: (top) Spearman correlation between the edit distances and the systematicity and efficiency scores from the scoring rubrics. (bottom) Adjusted mutual information between the edit distances and the systematicity and efficiency scores.

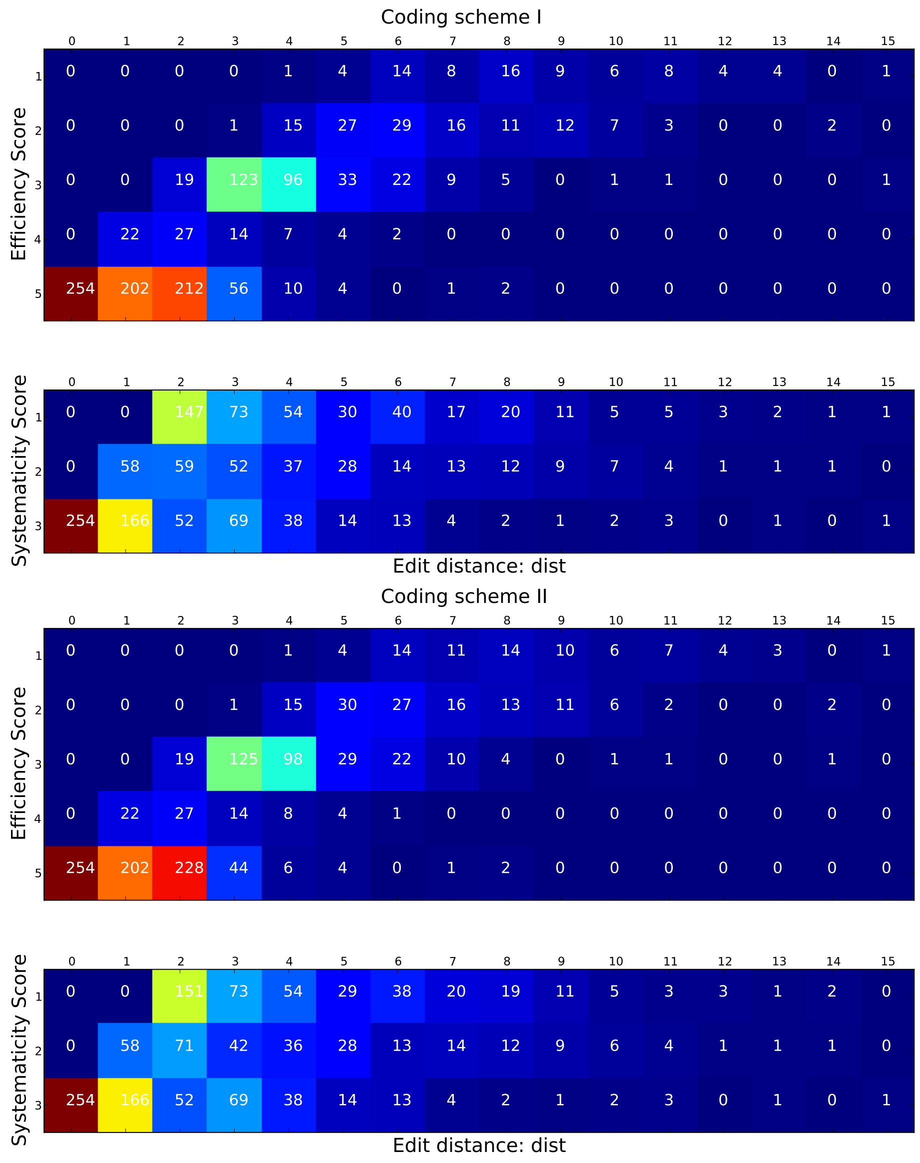
forms us that the edit distances to the ideal action sequences are the best first guess to create an edit distance–based measure to quantify students' performance in G/SBTs, and thus can serve as an informative feature variable that characterizes the performance. Third, most of the edit distances other than "dist" do not display distinct distributions compared with those calculated from random responses, whereas most of them do show high correlations with the efficiency score; this implies that the efficiency score may not be a well-defined dimension of the construct as it highly correlates with those edit distances that do not manifest clear distinctions between real responses and random responses. Finally, the information encoded in the edit distances is different from the information contained in the rubric-based scores, though there are reasonably high (negative) correlations. Each of these approaches probes the process data from a specific perspective, and the vector space all these features span needs to be analyzed to extract complete information about the students' performances in G/SBTs. In the next subsection, we focus on the vector space spanned by the edit distances corresponding to different action sequences and explore the ways to combine them.

3.4. Principal component analysis and hierarchical cluster analysis

The various edit distances corresponding to different action sequences lead to a multidimensional (six, in our case) vector space. Given the possible correlations among the edit distances, a natural question is, what is the actual dimension of this space, and what are the ways to combine these edit distances that are on the same dimension? To tell how many actual dimensions the vector space really has, probably the best approach is to perform a PCA. In Figure 6, we show results from the PCA for the two coding schemes. The results show that both coding schemes lead to two effective dimensions (PCs) based on the scree plot for the vector spaces spanned by the six edit distances. The first PC is predominant, explaining over 90% of the variability of the data.

It is interesting to see how these PCs correlate with the scores from the scoring rubrics, because the results will tell us whether the space spanned by the edit distances covers the space spanned by the systematicity and efficiency scores. We show the results of the correlation in Figure 7. One can see that the PCs from the two coding schemes correlate differently with the two rubric-based scores. For coding scheme I, the first PC highly correlates (positively) with the efficiency score, while the second PC highly correlates (negatively) with the systematicity score. The other PCs are not strongly correlated with either score. Similar conclusions are applied to coding scheme II, though with some adjustments. For a more intuitive picture about what the correlation numbers mean, we present the scatter plots between the two rubric-based scores and the first and second PCs corresponding to coding scheme I in Figure 8. All these results show that the space spanned by the edit distances covers the space spanned by the rubric-based scores. Moreover, given that the first PC explains most of the variability in the data, and it is highly correlated with the efficiency score, one can infer that among the two rubric-based scores, the efficiency score is subject to higher variability and therefore might not be a very well defined dimension of the construct. This observation echoes our previous conclusion.

Conversely, one may want to check how we should group the edit distances. If there is a strong cognitive motivation, we can do a confirmatory factor analysis to check the grouping. However, in our case, the cognitive motivation is not yet very clear, and therefore we perform a hierarchical cluster analysis on the edit distances to see whether the results are consistent with our general understanding of the task. In Figure 9, we show the results for the two different



Coding scheme I

Efficiency Score	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	1	4	14	8	16	9	6	8	4	4	0	1
2	0	0	0	1	15	27	29	16	11	12	7	3	0	0	2	0
3	0	0	19	123	96	33	22	9	5	0	1	1	0	0	0	1
4	0	22	27	14	7	4	2	0	0	0	0	0	0	0	0	0
5	254	202	212	56	10	4	0	1	2	0	0	0	0	0	0	0

Systematicity Score	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	147	73	54	30	40	17	20	11	5	5	3	2	1	1
2	0	58	59	52	37	28	14	13	12	9	7	4	1	1	1	0
3	254	166	52	69	38	14	13	4	2	1	2	3	0	1	0	1

Edit distance: dist

Coding scheme II

Efficiency Score	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	1	4	14	11	14	10	6	7	4	3	0	1
2	0	0	0	1	15	30	27	16	13	11	6	2	0	0	2	0
3	0	0	19	125	98	29	22	10	4	0	1	1	0	0	1	0
4	0	22	27	14	8	4	1	0	0	0	0	0	0	0	0	0
5	254	202	228	44	6	4	0	1	2	0	0	0	0	0	0	0

Systematicity Score	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	151	73	54	29	38	20	19	11	5	3	3	1	2	0
2	0	58	71	42	36	28	13	14	12	9	6	4	1	1	1	0
3	254	166	52	69	38	14	13	4	2	1	2	3	0	1	0	1

Edit distance: dist

Figure 5: Cross table of the scores from the scoring rubrics vs. the edit distance for coding schemes I and II. Here the edit distance refers to the minimum of the edit distance with respect to C4R4PC5R5P and C5R5PC4R4P.

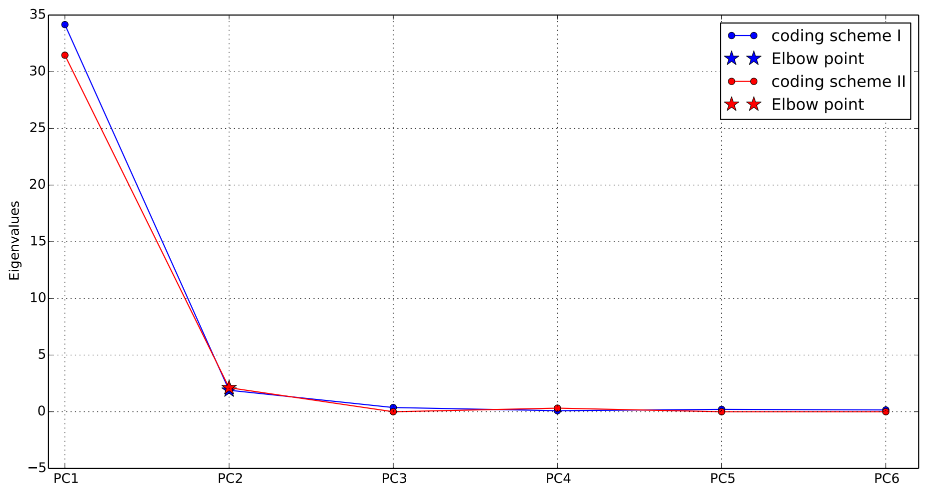


Figure 6: PCA on the space spanned by dist, dist2, dist3, dist4, dist5, and dist6. The results show that different coding schemes lead to slightly different results. Both PCAs indicate that the spaces spanned by the edit distances have two dimensions based on the scree test.

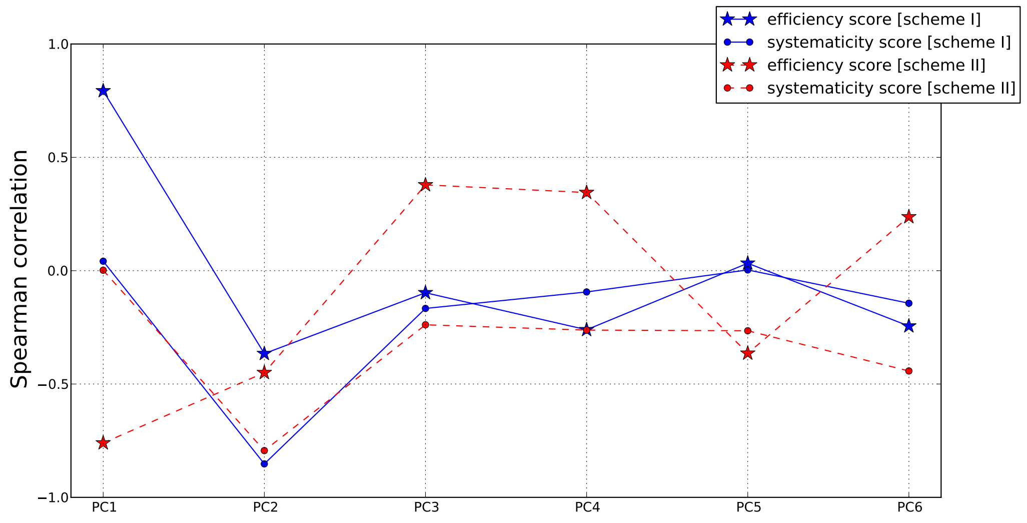


Figure 7: Spearman correlation between the PCs and the systematicity and efficiency scores.

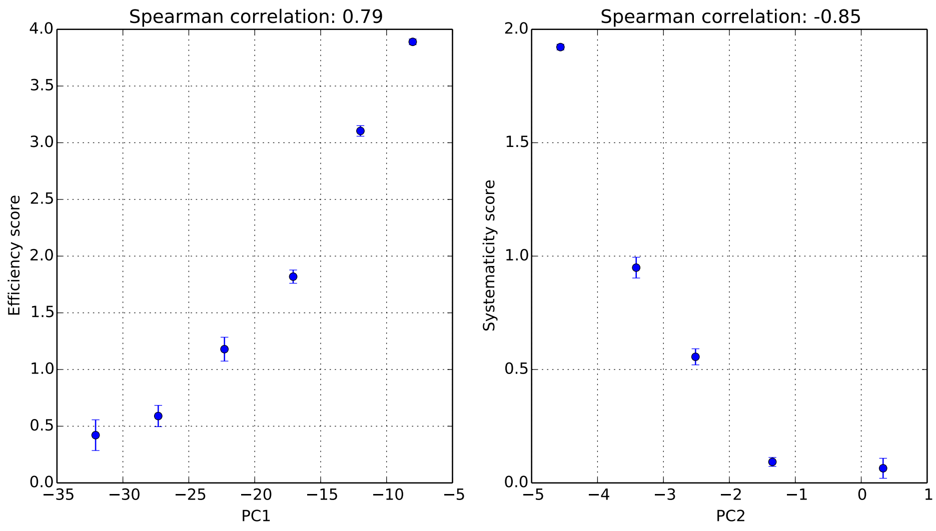


Figure 8: Scatter plots between the first two PCs and the rubric-based scores for coding scheme I. Each dot is the mean value of the bin, and the error bar is the standard deviation of the mean.

coding schemes. In coding scheme I, one can observe that there are three clusters: dist; dist2, dist3; and dist4, dist5, dist6. Such a clustering pattern is very consistent with our understanding of the task. The "dist" corresponds to the ideal action sequence and should be different from others; "dist2" and "dist3" correspond to the main action sequence segments in the ideal action sequence and is different from the rest (e.g., "dist4," "dist5," and "dist6"). So, a simple hierarchical cluster analysis further confirms the consistency of our choice of the ideal action sequences and action sequence segments. Similar conclusions are held for coding scheme II.

4. Discussion

In this article, we propose an edit distance approach to analyzing the process data from certain G/SBTs, where only the order of the actions is of primary interest. By considering a specific task, the pump repair task, from NAEP TEL, we lay down step-by-step procedures for applying the edit distance approach to the process data. By comparing the edit distances with the scores from the existing scoring rubrics of the pump repair task, we conclude that the information contained in the two scores is also reflected by the edit distances, though the latter is obtained from quite a different perspective. Moreover, PCA on edit distances, together with the internal consistency among the edit distances from actual responses and from random responses, suggests that the efficiency score based on the existing scoring rubrics needs further scrutinization.

However, the edit distance approach in its current form does have restrictions on what types of tasks it is applicable to. One of the major restrictions is that the temporal information as well as the properties of different actions are not included in the current scheme. This additional information about the process data may be accommodated into the edit distance scheme by assigning appropriate weights to the edit operations based on the time interval or action attributes. In addition to this restriction, how to properly interpret the "edit" operations is also challenging.

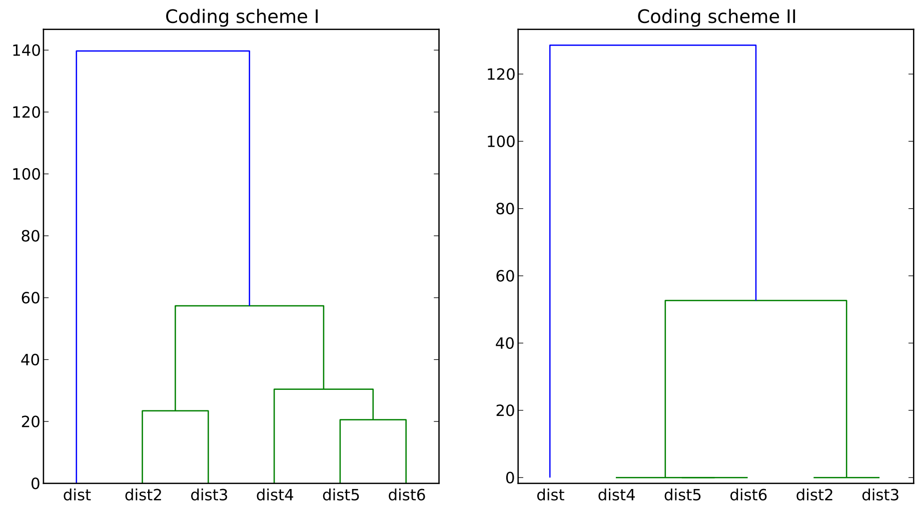


Figure 9: Hierarchical clustering analysis using complete linkage for edit distances dist, dist2, dist3, dist4, dist5, dist6.

The interpretation may be different for different G/SBTs, which leaves great room for imagination. In an ongoing study, we are applying the approach to a number of other G/SBTs, and we hope we can find more generalizable ways to interpret the "edit" operations after comparing across different tasks. Nevertheless, the edit distance approach remains a useful tool for exploring the process data from G/SBTs, allowing quick construction of meaningful feature variables to help to refine the scoring rules.

ACKNOWLEDGMENT

The work is supported by allocation funding from Educational Testing Service. J.H. thanks Shelby Haberman for helpful conversations.

REFERENCES

ALMOND, R., STEINBERG, L., AND MISLEVY, R. 2002. Enhancing the design and delivery of assessment systems: A four-process architecture. *The Journal of Technology, Learning and Assessment 1,* 5.

BERGNER, Y., SHU, Z., AND VON DAVIER, A. 2014. Visualizing and clustering sequence data from a simulation-based assessment task. *Journal of Educational Data Mining*.

BRYANT, V. 1985. *Metric spaces: iteration and application*. Cambridge University Press.

CHIEU, V. M., LUENGO, V., VADCARD, L., AND TONETTI, J. 2010. Student modeling in orthopedic surgery training: Exploiting symbiosis between temporal bayesian networks and fine-grained didactic analysis. *International Journal of Artificial Intelligence in Education 20,* 3, 269–301.

DESMARAIS, M. C. AND LEMIEUX, F. 2013. Clustering and visualizing study state sequences. In *Proceedings of 6th International Conference on Educational Data Mining*, pp. 224–227.

GABADINHO, A., RITSCHARD, G., STUDER, M., AND MÜLLER, N. S. 2009. Mining sequence data in r with the traminer package: A users guide for version 1.2. *Geneva: University of Geneva*.

GEE, J. P. 2007. *What video games have to teach us about learning and literacy.: Revised and Updated Edition*. Macmillan.

GUTIERREZ-SANTOS, S., MAVRIKIS, M., AND MAGOULAS, G. 2010. Sequence detection for adaptive feedback generation in an exploratory environment for mathematical generalisation. In *Artificial Intelligence: Methodology, Systems, and Applications*, pp. 181–190. Springer.

JURAFSKY, D. AND MARTIN, J. H. 2000. *Speech & Language Processing*. Pearson Education India.

KLOPFER, E., OSTERWEIL, S., GROFF, J., AND HAAS, J. 2009. Using the technology of today, in the classroom today. *The Education arcade*.

KÖCK, M. AND PARAMYTHIS, A. 2011. Activity sequence modelling and dynamic clustering for personalized e-learning. *User Modeling and User-Adapted Interaction 21,* 1-2, 51–97.

LEVENSHTEIN, V. I. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, Volume 10, pp. 707.

MISLEVY, R., ORANJE, A., BAUER, M. I., VON DAVIER, A. A., HAO, J., CORRIGAN, S., HOFFMAN, E., DICERBO, K., AND JOHN, M. 2014. *Psychometric considerations in game based assessments*. CreateSpace Independent Publishing Platform.

MISLEVY, R. J. AND RICONSCENTE, M. 2006. Evidence-centered assessment design. *Handbook of test development*, 61–90.

PUMPREPAIR 2013. *Pump Repair Sample Task*. `http://nces.ed.gov/nationsreportcard/tel/wells_item.aspx`.

SAO PEDRO, M. A., DE BAKER, R. S., GOBERT, J. D., MONTALVO, O., AND NAKAMA, A. 2013. Leveraging machine-learned detectors of systematic inquiry behavior to estimate and predict transfer of inquiry skill. *User Modeling and User-Adapted Interaction 23,* 1, 1–39.

SCHMIT, M. J. AND RYAN, A. M. 1992. Test-taking dispositions: A missing link? *Journal of Applied Psychology 77,* 5, 629.

SUNDRE, D. L. AND WISE, S. L. 2003. Motivation filtering: An exploration of the impact of low examinee motivation on the psychometric quality of tests. In *annual meeting of the National Council on Measurement in Education, Chicago, IL.*

TEL 2013. *Technology and Engineering Literacy Assessments*. `https://nces.ed.gov/nationsreportcard/tel/`.

USMLE 2014. *United States Medical Licensure Examinations*. `http://www.usmle.org/pdfs/step-3/2014content_step3.pdf/`.

VINH, N. X., EPPS, J., AND BAILEY, J. 2009. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1073–1080. ACM.

Understanding Student Language: An Unsupervised Dialogue Act Classification Approach

Aysu Ezen-Can
North Carolina State University
aezen@ncsu.edu

Kristy Elizabeth Boyer
North Carolina State University
keboyer@ncsu.edu

Within the landscape of educational data, textual natural language is an increasingly vast source of learning-centered interactions. In natural language dialogue, student contributions hold important information about knowledge and goals. Automatically modeling the dialogue act of these student utterances is crucial for scaling natural language understanding of educational dialogues. Automatic dialogue act modeling has long been addressed with supervised classification techniques that require substantial manual time and effort. Recently, there is emerging interest in unsupervised dialogue act classification, which addresses the challenges related to manually labeling corpora. This paper builds on the growing body of work in unsupervised dialogue act classification and reports on the novel application of an information retrieval technique, the Markov Random Field, for the task of unsupervised dialogue act classification. Evaluation against manually labeled dialogue acts on a tutorial dialogue corpus in the domain of introductory computer science demonstrates that the proposed technique outperforms existing approaches to education-centered unsupervised dialogue act classification. Unsupervised dialogue act classification techniques have broad application in educational data mining in areas such as collaborative learning, online message boards, classroom discourse, and intelligent tutoring systems.

1 Introduction

Natural language constitutes a vast portion of educational data. Recent years have seen a surge of educational data mining research aimed at modeling natural language data and leveraging those models to further student learning or to support effective teaching. Natural language has been mined within sources including lecture notes (Atapattu, Falkner, and Falkner, 2014), MOOC discussion forums (Wen, Yang, and Rosé, 2014), traditional class message boards (Yoo and Kim, 2014), computer-supported collaborative learning technologies (Kumar, Beuth, and Rosé, 2011), collaborative learning transcripts (D'Mello, Olney, and Person, 2010), tutorial dialogue systems (Graesser, VanLehn, Rosé, Jordan, and Harter, 2001), online discussion forums (Ferguson, Wei, He, and Buckingham Shum, 2013) and student peer-reviews (Xiong and Litman, 2014). Models of natural language have been used to achieve goals including generating questions (Niraula, Rus, Stefanescu, and Graesser, 2014), assessing students' prior knowledge (Stefanescu, Rus, and Graesser, 2014), identifying social deliberative behavior (Xu, Murray, Woolf, and Smith, 2013), predicting task completion (González-Brenes, Mostow, and Duan, 2011), and detecting and predicting affect in educational games (Forsyth, Graesser, Pavlik Jr, Cai, Butler, Halpern, and Millis, 2013).

A primary focus of educational data mining of natural language interactions is to identify highly effective teaching strategies and implement them within educational systems that engage in dialogue (e.g., (Mostow, Beck, Cen, Cuneo, Gouvea, and Heiner, 2005)). Research suggests that dialogue-rich interactions may foster improved learning because of the adaptive collaboration mechanisms within dialogue (Graesser, Person, and Magliano, 1995), the available naturalness of expression (Litman, Rosé, Forbes-Riley, VanLehn, Bhembe, and Silliman, 2006), and by supporting students' self-explanation (Aleven, Popescu, and Koedinger, 2001) even for ill-defined domains (Weerasinghe, Mitrovic, and Martin, 2009).

Natural language dialogue can be modeled at many levels of granularity, and full natural language understanding involves a multi-step pipeline that links raw utterances to their semantics, intent, and context (Jurafsky and Martin, 2000). One of the most useful levels of dialogue modeling is *dialogue act classification* which identifies the communicative action or intent of each utterance, such as questions, hints, or statements (Allen, Schubert, Ferguson, Heeman, Hwang, Kato, Light, Martin, Miller, Poesio, et al., 1995; Core and Allen, 1997; Serafin and Di Eugenio, 2004; Traum, 1999; Stolcke, Ries, Coccaro, Shriberg, Bates, Jurafsky, Taylor, Martin, Van Ess-Dykema, and Meteer, 2000). Understanding student dialogue acts is a central challenge. For example, in a tutorial dialogue system, distinguishing whether the student is asking a question, requesting feedback, proposing a plan, or expressing affect is critical for subsequent tutorial move selections. Similarly, when modeling dialogues within message boards of MOOCs, student dialogue acts such as indirect questions, commitments, and social coordination may be of particular importance. In these and other contexts, dialogue act modeling provides key information to understanding student natural language contributions.

While dialogue act classification has been studied extensively for decades and reliable techniques exist for *supervised* dialogue act modeling, there are substantial challenges for scaling these for educational data mining. First, supervised dialogue act models rely on handcrafted dialogue act tag sets which are often highly corpus-specific and require substantial consideration of the domain and of dialogue theory. Second, the manual effort required to label corpora so that supervised models can be trained is high. For these reasons, our work has focused for the past several years on unsupervised dialogue act modeling, which has only recently emerged as a research focus within the dialogue systems research community (Ezen-Can and Boyer, 2014b; Crook, Granell, and Pulman, 2009; Higashinaka, Kawamae, Sadamitsu, Minami, Meguro, Dohsaka, and Inagaki, 2011; Joty, Carenini, and Lin, 2011; Ritter, Cherry, and Dolan, 2010; Lee, Jeong, Kim, Ryu, and Lee, 2013) and to a very limited extent within educational data mining research (Ezen-Can and Boyer, 2013, 2014a; Rus, Moldovan, Niraula, and Graesser, 2012). The goal is to build well-formed models characterized by cohesive dialogue act groups that facilitate interpretation and subsequent use within systems to support teaching and learning.

This paper presents a novel approach to unsupervised natural language dialogue modeling for educational data mining, with application to tutorial dialogue. Leveraging highly effective techniques from the computational linguistics subfield of information retrieval, we propose a clustering approach based on a graph-based Markov Random Field (MRF) framework to group dialogue utterances with the same dialogue act in an unsupervised way, that is, without requiring the dialogue acts to be labeled manually ahead of time. We compare this new approach to prior unsupervised approaches in the literature and find that it outperforms all previously reported approaches, including our earlier work on query-likelihood clustering for dialogue act modeling (Ezen-Can and Boyer, 2013).

The organization of this article is as follows. We first give an overview of supervised and unsupervised classifiers in the Related Work section, followed in Section 3 by a description of the corpus used for modeling student utterances in this work. The steps undertaken for leveraging information retrieval techniques for dialogue act classification are described in Section 4. Section 5 presents experiments comparing MRF-based clustering with our prior query-likelihood approach and other unsupervised dialogue act modeling work in the educational data mining literature, detailing the results both quantitatively and qualitatively. Section 6 provides both quantitative and qualitative evaluation results with a held-out test set for the best performing MRF model. In Section 7 the performance of the dialogue act classifier is evaluated in terms of its capability for understanding students: we evaluate which dialogue acts are harder to distinguish by comparing the classifier's performance for specific dialogue acts. Finally in Section 8, we summarize the work presented in this article with some final remarks and future work.

2 Related Work

The idea that human conversation contains *speech acts* originated with sociolinguistic theorists (Austin, 1975; Searle, 1969). Dialogue act theory suggests that humans not only communicate factual information within natural language utterances, they often express underlying intended action (e.g., to ask a question, to give a command). The practical value of dialogue acts for computational linguistics has been well demonstrated, with an extensive literature on automated dialogue act classification approaches. Most of these approaches rely on supervised dialogue act classifiers. Hidden Markov models (Stolcke et al., 2000; Boyer, Ha, Phillips, Wallis, Vouk, and Lester, 2010), maximum entropy models (Rangarajan Sridhar, Bangalore, and Narayanan, 2009), conditional random fields (Quarteroni, Ivanov, and Riccardi, 2011), decision trees (Shriberg, Stolcke, Jurafsky, Coccaro, Meteer, Bates, Taylor, Ries, Martin, and Van Ess-Dykema, 1998) and support vector machines (Sadohara, Kojima, Narita, Nihei, Kamata, Onaka, Fujita, and Inoue, 2013) are some of the methods proposed by researchers for supervised dialogue act classification. For tutorial dialogue, promising approaches have included an extension of latent semantic analysis (Serafin and Di Eugenio, 2004), a syntactic parser model (Marineau, Wiemer-Hastings, Harter, Olde, Chipman, Karnavat, Pomeroy, Rajan, Graesser, Group, et al., 2000) and vector-based classifiers (Boyer et al., 2010), which typically achieve higher than 75% accuracy (Bangalore, Di Fabbrizio, and Stent, 2008; Forbes-Riley and Litman, 2005; Serafin and Di Eugenio, 2004).

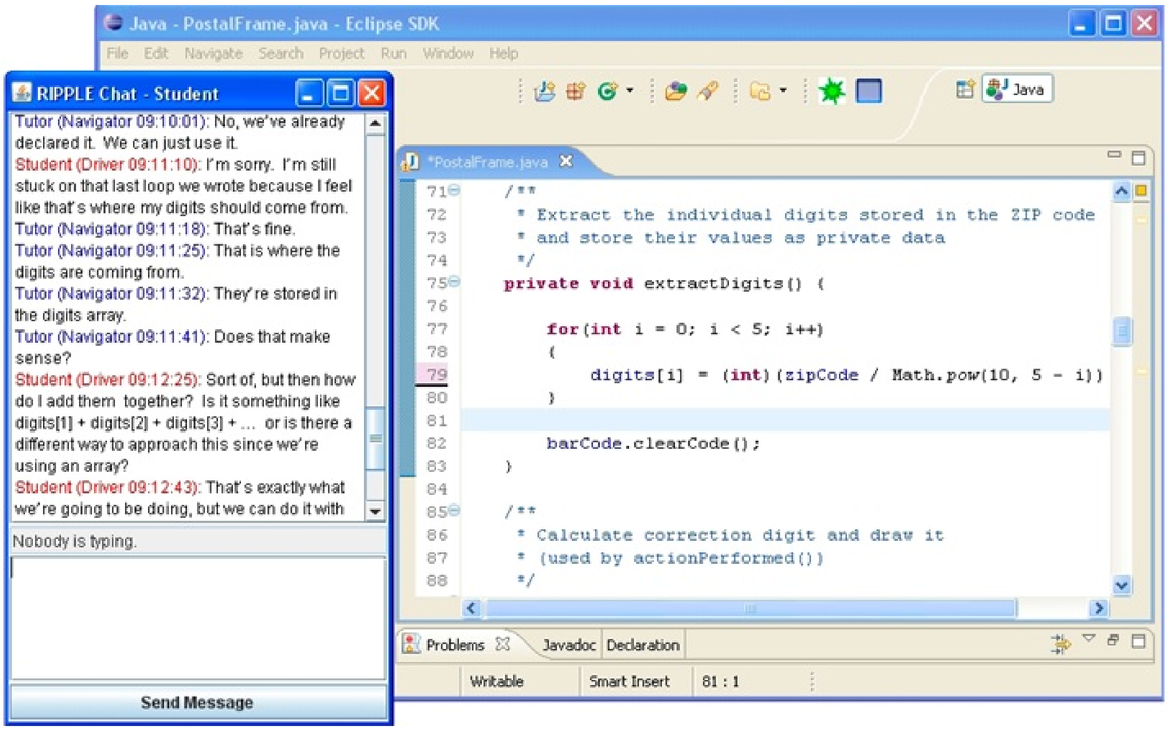
Following this line of investigation, recent years have witnessed a growing body of work on *unsupervised* dialogue act modeling. Ritter et al. (2010) utilized Hidden Markov Models (HMMs) with topic information for modeling Twitter conversations. They separated content words (topic words) with the help of a Latent Dirichlet Allocation framework. Other research has followed this direction by proposing a variation of HMM for asynchronous conversations such as e-mails and forums, defining dialogue act emission distributions as mixture models and adding dialogue structure features (Joty et al., 2011). Dirichlet Process Mixture Models with a non-parametric Bayesian approach for train fares and timetables have also been explored (Crook et al., 2009), and a subsequent improvement on that work used a hierarchical Dirichlet Process with Hidden Markov Models for extracting semantics from utterances (Lee et al., 2013). Lee and colleagues used a three-step approach: dialogue act, intent and slot entity recognition applied on spoken language. Similar to Ritter and colleagues, Lee et al. assume that each word is generated by one of three sources: words in the current dialogue act, general words and domain words. Another non-parametric Bayesian method, infinite HMM, has been explored within a Japanese discussion domain, and the results were compared with those obtained using the Chinese Restaurant Process showing that the infinite HMM performed better in terms of purity and F-measure (Higashinaka et al., 2011). There have also been attempts at clustering dialogue acts on educational corpora using k-means (Rus et al., 2012) as well as our prior work on combining query-likelihood with clustering (Ezen-Can and Boyer, 2013). Overall, the prior unsupervised approaches have substantially underperformed current supervised approaches. Additionally, with the exception of one hidden Markov Modeling approach (Joty et al., 2011), unsupervised models have not exploited word-ordering information, instead using a bag-of-words representation of utterances. The current work leverages word order for dialogue act clustering. We propose a dialogue act modeling framework that takes word-ordering information into account by representing the relationships between utterances as a Markov random field utilizing the probabilities obtained from the graph for clustering. This approach advances the state of the art for understanding students in terms of classifying dialogue acts.

3 Tutorial Dialogue Corpus

Our goal in this study is to understand students better by utilizing educational data mining for the task of dialogue act classification. Therefore, we mine a task-oriented tutorial dialogue corpus collected in 2007 for an introductory Java programming project. The corpus consists of student-tutor interactions in a computer-mediated environment while collaborating on computer programming problems (Boyer, Vouk, and Lester, 2007; Boyer et al., 2010; Boyer, Phillips, Ingram, Ha, Wallis, Vouk, and Lester, 2011). Students were allowed to ask questions and make fully unrestricted dialogue moves to their tutors via the textual communication channel in the course of solving programming tasks (see Figure 1).

The corpus consists of 1,525 student utterances from 43 distinct students (averaging 7.54

words per utterance) and 3,332 tutor utterances from two paid tutors (averaging 9.04 words per utterance). The corpus was manually annotated in prior work (with 0.80 Kappa) for both tutor and student dialogue acts that analyzed relationship between tutoring modes and student learning outcomes (Boyer et al., 2011). The tagging scheme consists of nine dialogue acts, the distribution of which is depicted in Table 1. The most frequently appearing dialogue act is EQ (evaluation questions) with 27.3%, which is the majority baseline chance constituting performance of a model that is equal to chance. Excerpts from the corpus can be seen in Table 2. For this article, the manual annotations are used only for evaluation purposes because unsupervised approaches assumes that manual labels are not accessible to the model building phase.



Tutor (Navigator 09:10:01): No, we've already declared it. We can just use it.
Student (Driver 09:11:10): I'm sorry. I'm still stuck on that last loop we wrote because I feel like that's where my digits should come from.
Tutor (Navigator 09:11:18): That's fine.
Tutor (Navigator 09:11:25): That is where the digits are coming from.
Tutor (Navigator 09:11:32): They're stored in the digits array.
Tutor (Navigator 09:11:41): Does that make sense?
Student (Driver 09:12:25): Sort of, but then how do I add them together? Is it something like digits[1] + digits[2] + digits[3] + ... or is there a different way to approach this since we're using an array?
Student (Driver 09:12:43): That's exactly what we're going to be doing, but we can do it with

Nobody is typing.

```
    /**
     * Extract the individual digits stored in the ZIP code
     * and store their values as private data
     */
    private void extractDigits() {

        for(int i = 0; i < 5; i++)
        {
            digits[i] = (int)(zipCode / Math.pow(10, 5 - i))
        }

        barCode.clearCode();
    }

    /**
     * Calculate correction digit and draw it
     * (used by actionPerformed())
     */
```

Figure 1: Screenshot from the human-human tutorial dialogue interface.

4 Unsupervised Dialogue Act Modeling

The goal of this work is to model student dialogue acts in an unsupervised manner. Our novel approach adapts information retrieval techniques combined with clustering for modeling student dialogue acts. In this section, the framework utilized for dialogue act classification is described.

Student Dialogue Act	**Example**	**Percent of dialogue messages**
Evaluation Question (EQ)	*can I parse a character to an int*	27.3
Question (Q)	*What is the next step?*	18.1
Statement (S)	*ohh let me try something else then*	13.8
Grounding (G)	*ok*	12.6
Extra-Domain (EX)	*Ok I have read over the first page*	8.7
Positive Feedback (PF)	*that works great*	7.6
Negative Feedback (NF)	*I'm having trouble figuring out how to pass the parameter to the other methods though*	6.0
Greeting (GRE)	*hi*	3.7
Lukewarm Feedback (LF)	*we've sort of learned them but im not exactly sure how to use them*	2.1

Table 1: Student dialogue act tags and their frequencies.

First we describe the natural language preprocessing, followed by the information retrieval techniques used to calculate utterance similarity and finally the clustering phase, which utilized calculated similarities to group utterances.

Information retrieval is the process of searching available resources to retrieve results that are similar to the query (Ricardo and Ribeiro-Neto, 1999). Probabilistic models such as language models are commonly used for this purpose (Manning, Raghavan, and Schütze, 2008). For example, in document search, documents that are expected to be relevant to a query are sorted according to their relevance to the query based on a similarity score. We adapt this information retrieval approach to compute similarity between utterances. This similarity information can then be used for clustering relevant utterances together, forming groups of utterances that we hypothesize represent the same dialogue act.

This section presents a novel approach to unsupervised dialogue act modeling: Markov Random Field based clustering. First we describe the technique that MRF-based clustering will be compared against, query-likelihood clustering, which is also based on information retrieval techniques. Query-likelihood clustering treats each student utterance as a query and retrieves other similar utterances from the corpus. MRF-based clustering builds on this approach with one key difference: MRF takes word ordering into account, unlike prior unsupervised dialogue act modeling approaches (Metzler and Croft, 2005). We hypothesize that MRF will perform better

Excerpt 1:
Student: will there be a loop in this? [*Q*]
Tutor: well, we need to add up all of the values in DigitArray [*S*]
Tutor: the easiest way to do that is with another loop [*S*]
Tutor: so yes [*S*]
Student: ok [*G*]
Student: do i declare a new in within this class or do i need to declare toward the beginning of the code? [*Q*]
Tutor: within this method is fine [*S*]
Student: cool [*G*]
Excerpt 2:
Student: what did i do wrong on this line [*Q*]
Tutor: Z isn't declared anywhere [*S*]
Tutor: so you want to declare Z as an array of int's [*S*]
Tutor: if you hit Save as you go those errors will update (Ctrl+S on the keyboard) [*EX*]
Student: how am i doing so far? [*EQ*]
Tutor: looks good [*PF*]
Tutor: keep going then we'll talk about how to shorten this up with a for-loop [*S*]
Student:how do i shorten this? [*Q*]
Tutor: well first let's try running it to make sure it's right [*S*]
Tutor: try putting a system.out.println in your method and run it [*S*]
Student: what is it meant to print out? [*Q*]
Tutor: if you call say System.out.println("5"); then that would print a 5 in the console when you ran it [*S*]

Table 2: Excerpts of dialogue from the corpus with the corresponding dialogue act tags in brackets.

than all prior techniques. The experiments described in the next section demonstrate this with empirical results.

In Section 4.1., we first explain the preprocessing required for modeling, and in Section 4.2., we detail the query-likelihood and MRF-based similarity calculations. Then, in Section 4.3. we describe the clustering.

4.1. Natural Language Preprocessing

When modeling natural language data, a series of natural language processing steps are often required prior to proceeding with further modeling. The types of features that are most useful to produce in the initial natural language processing steps are often not known in advance and is the subject of preliminary experimentation, as is the case in this work. In its raw form, natural language dialogue utterances are a series of *tokens* which include words and punctuation. Some dialogue modeling approaches work best when given raw word-level tokens while other models benefit from abstracting the actual words in some way, for example to their parts of speech (POS). This work presents an example of this phenomenon. POS tagging labels each word according to their grammatical part of speech such as noun, verb, and adjective (Marcus, Santorini, and Marcinkiewicz, 1993). Because POS tagging represents words by their function in sentences, it provides a level of generalization that can be useful in dialogue modeling (Becker, Basu, and Vanderwende, 2012; Boyer et al., 2010; Di Eugenio, Xie, and Serafin, 2010). Because POS tagging is so useful, there are many available automatic POS taggers. In this work we utilize the Stanford Parser (Klein and Manning, 2003). We experimented with both actual words and with full part-of-speech backoff. While the best results for MRF-based clustering were achieved with raw features (actual words), query-likelihood clustering reached its best performance with a combination of raw features and POS tags. The hybrid approach utilized for query-likelihood clustering replaces function words such as determiners ("the", "a"), conjunctions ("and", "but"), and prepositions ("in", "after") with their POS tags.

A second important preprocessing choice is whether to use content words themselves or whether to stem them. Stemming is the process of generalizing words to their roots. For query-likelihood clustering, content words were retained but stemmed (e.g., "parameter" becomes "paramet", "completely" becomes "complet") to reduce the number of distinct words in the vocabulary of the corpus under consideration. We use the Snowball stemmer in this work[1].

Another consideration for preprocessing raw natural language dialogue data is how to represent special entities within the utterances. In our domain of computer science learning, the natural language contains special characters that indicate semantically important entities related to the domain, such as short bits of programming code. Students often incorporate some code such as function names or variable names into their text messages to tutors within the course of the dialogue. Although they are important with regard to the tutoring task, they require additional preprocessing in order to be handled appropriately by automated natural language processing techniques. Therefore, code segments in the corpus were manually replaced with meaningful tags representing them. For instance, segments about array indexing, which may originally have appeared as "$x[i]$" and been mishandled, were replaced with the text "ARRAY_INDEXING". *If*

[1]http://snowball.tartarus.org

statements, loops and arithmetic operations were all replaced in the corpus using similar conventions.[2]

4.2. Utterance Similarity Calculation

Having preprocessed the corpus, we now have a set of utterances that will be used for dialogue act classification. The next step is to calculate how similar these utterances are to each other so that we can cluster them. We report on the novel MRF-based model and compare it against query-likelihood modeling, both of which adapt information retrieval techniques as described in the following subsections.[3]

4.2.1. Query-Likelihood Model

We have previously reported on a technique to calculate similarities between utterances using query-likelihood (Ezen-Can and Boyer, 2013), and we describe this technique here for comparison. Query-likelihood is based on a language model which originates from the Bayesian assumption that each token is independent from every other token (Manning et al., 2008). This technique is widely used in search engines. Given a query, a list of documents that are expected to be relevant to the query are returned. We adapt this information retrieval goal to our purposes where we focus on finding similar utterances instead of documents. The process can be summarized as follows: *given a target utterance, find a list of utterances that are similar to the target.* The similar utterances are obtained from our corpus of student utterances. Table 3 presents two sample queries and top three most similar utterances retrieved by the query-likelihood model.

To achieve this, the query-likelihood model searches for words that are shared between the query and the utterances in the corpus. We use query-likelihood modeling to calculate proximity between utterances in a similarity space. For every target utterance whose dialogue act is to be classified, query-likelihood produces similarities to every other utterance in the corpus. In this way, we obtain a list of similar utterances for each utterance in the corpus. Because we would like to obtain groups of utterances that share the same dialogue act, having lists of similarity information is not sufficient. Therefore we need to use these lists for clustering. Using each

[2]Performing this annotation automatically is the focus of ongoing research. Differentiating programming code within natural language utterances is a challenging problem.

[3]The Lemur Project's information retrieval implementation (Indri) was used in this work (Strohman, Metzler, Turtle, and Croft, 2005).

Target utterance	Top three most similar student utterances
I am confused	- here's the part I am really confused on is this where I have to call up another class - and if so, then I guess I am sort of confused about how to retrieve the appropriate values from the table array - I'm confused on what I am going to put inside the loop
how can I solve	- how can I pull values out of an array or can I reference them with code like ARRAY_INDEXING - Is it like I have it on my screen And can I also set how long my array will be when I say private int PARAMETER -yea but i just can't remember to how to use the METHOD_CALL to get each individual number

Table 3: Sample queries and their top three query-likelihood results.

produced list, we create a vector representation that shows each utterance in the list as a 1 indicating presence of that utterance in the list of similar utterances and others as a 0 indicating absence of the utterance, to perform clustering. Figure 2 illustrates this process.

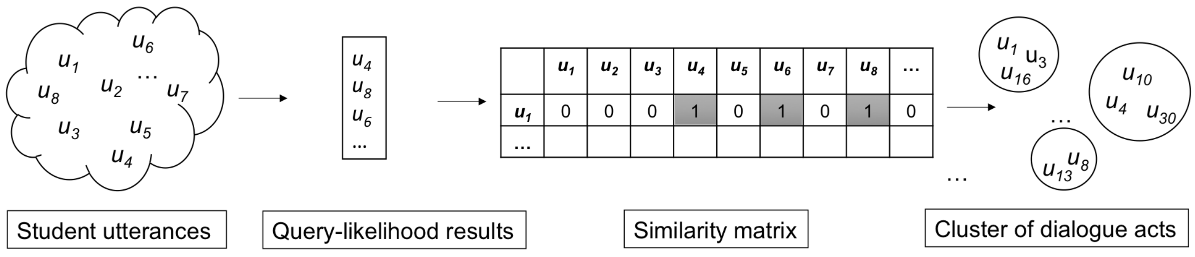


Figure 2: The query-likelihood clustering framework.

4.2.2. Markov Random Field Model

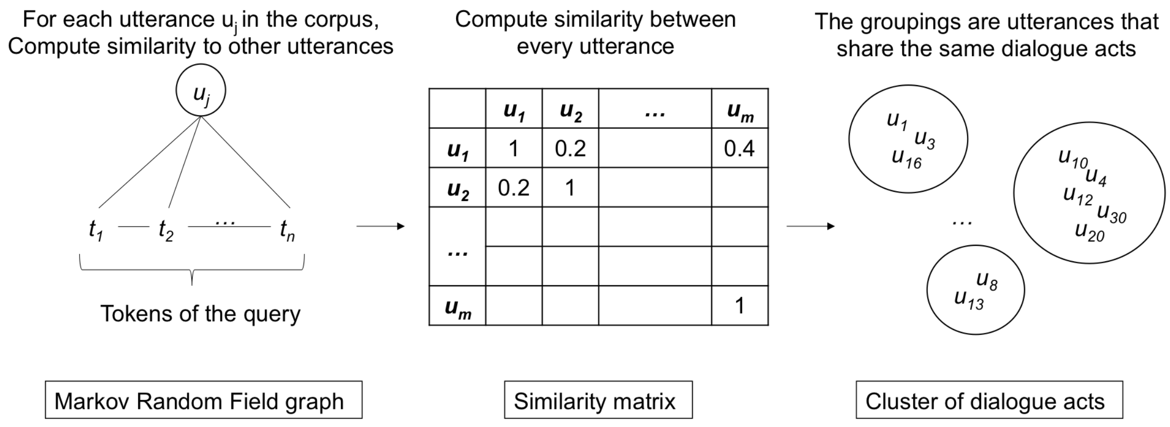
The previous section described the query-likelihood technique. The new approach presented here explicitly models the token ordering within a graphical representation, MRF graphs. In this way, while finding similarities between two utterances, we aim to consider the sequential order of each token rather than using a bag-of-words approach. For example, the utterances "I am correct" and "am I correct" have higher similarity scores to each other using query-likelihood clustering than MRF because all tokens are the same without considering the token ordering.

However, in this case a lower similarity is desirable as one of the utterances is a statement and the other is a question in terms of their communicative intentions. This distinction is the main motivation of MRF-based clustering. We hypothesize that if the similarities of utterances utilized in clustering technique are calculated more accurately by taking the token ordering into consideration, the clustering performance will improve as well. We therefore use a formulation of MRF that takes word ordering into account (Metzler and Croft, 2005).

MRF has been shown to be one of the best performing algorithms in the information retrieval community (Metzler and Croft, 2005). Therefore, proving its theoretical background is beyond the scope of this article. However, we would still like to provide motivation for using MRF for similarity calculation by discussing some differences with widely used metrics in the natural language processing literature. *Skip n-grams* are a generalization of *n-grams* where the words need not be consecutive, but there may be gaps within a window size. For example, if the window size is w, a bigram (n-gram with n=2) will consider a sequence of four tokens with the window of 2 tokens changing in between. Although this technique is widely used, skip n-grams require a window size to be set before computation whereas MRF does not, which is a motivating factor for using MRF. In addition, the use of n-grams gets more computationally expensive as n increases. While representing n-grams as feature vectors, the length of the vector is given by the whole set of n-grams in the corpus, making the vector large and the effect of the non-zero values lower, whereas in MRF likelihood summation, the non-zero values do not affect the similarity calculation. This property of MRF enables the nonzero values to be given more importance as desired. Further, MRF does smoothing for frequently occurring tokens, taking into account more important and representative words.

Recall that the purpose of using MRFs is to identify similar utterances. Another way to do this would be *longest common subsequence* to compute the similarity of not necessarily contiguous sequences of words. This technique is costly because all window sizes are computed, whereas it is possible to limit the window size (e.g., w=2) in MRF. In addition, it is possible to weight some values of w and n more than others in MRF by giving different weights to edges. The ordered tokens can be weighted more than single words ($n \geq 2$ more than $n = 1$) and phrases with one skipped token weighed more than multiple skipped tokens ($w = 1$ more than $w \geq 2$), whereas it is very difficult to do weighting with the longest common subsequence metric. It would require weighting some of the columns in the feature vector compared to a better structured way provided by MRF. Motivated by the flexibility of MRF models, we apply this technique to calculate similarities between utterances.

First, MRF models are drawn and the probabilities obtained from the undirected graph serve as inputs to the clustering algorithm (i.e., k-medoids clustering) as shown in Figure 3. Suppose y_i is an utterance that we would like to calculate the similarities to every other utterance u_j in the corpus. We draw an MRF model with u_j being the root and $t_1...t_k...t_n$ the leaves where t_k are the tokens of the target utterance y_i. This graph is drawn for every utterance in the corpus for which we would like to calculate proximity to y_i. The edges represent how well the token t_k describes the utterance u_j. Using the cliques formed via edges between one utterance and each token, we create a similarity matrix. For instance, the first row of the similarity matrix shows the similarities of utterance u_1 to every utterance in the corpus and similarly the second row for utterance u_2. This produces a symmetric matrix, therefore it is sufficient to compute only the half above the diagonal. Then, these similarity values are input into the clustering technique that outputs groups of utterances that are hypothesized to share the same dialogue act.


For each utterance u_j in the corpus, Compute similarity to other utterances
u_j
t_1 t_2 ... t_n
Tokens of the query
Markov Random Field graph
Compute similarity between every utterance
Similarity matrix
The groupings are utterances that share the same dialogue acts
u_1 u_3 u_{16}
u_{10} u_4 u_{12} u_{30} u_{20}
u_8 u_{13}
Cluster of dialogue acts

	u_1	u_2	...	u_m
u_1	1	0.2		0.4
u_2	0.2	1		
...				
u_m				1

Figure 3: The MRF-based clustering framework.

To compute the needed similarities between every pair of utterances in the corpus, we consider 3-node cliques and 4-node cliques which are analogous to bigrams (n-grams with $n = 2$) and trigrams ($n = 3$) considering the root. Additionally, we consider all values of window size w, which is analogous to skip bigrams and trigrams. In the MRF, a 2-clique is formed when an edge exists between two nodes (a token t_k and utterance u_j). This represents a bag-of-words approach where we calculate how well token t_k describes utterance u_j. For calculating the edge values of a 3-clique, we search for skip bigrams for all values of w limited by the length of u_j. We use the edge values obtained from the graphs as similarity measures between utterances. This approach is a generalization of all three techniques previously mentioned: bag-of-words, skip n-grams and longest common subsequence.

For the bag-of-words approach ($n = 1$, unigrams), we calculate a smoothed language model estimate $P(t_k|u_j)$ where the probability measures the likelihood of token t_k describing u_j. In other words, we wish to estimate the importance of this token in the utterance compared to the importance of this token in the entire corpus. This probability can be estimated as follows:

$$P(t_k|u_j) = [(1 - \alpha_d)\frac{freq(t_k, u_j)}{|u_j|} + \alpha_d \frac{freq(t_k, C)}{|C|}] \tag{1}$$

where $freq(t_k, u_j)$ is the frequency of token t_k in u_j, $|u_j|$ is the total number of tokens in u_j, $freq(t_k, C)$ is the frequency of token t_k in the corpus, and $|C|$ is the total number of tokens in the corpus. The smoothing is included to assign a non-zero probability to unseen words in utterance u_j that are present in the corpus, a common technique for natural language distributions where many words occur with low frequency (Zhai and Lafferty, 2001).

In addition to unigrams, we also wish to consider the more flexible skip n-grams with $w \geq 0$. To do this, we estimate the probability that tokens t_k and t_m, possibly separated by w other tokens, describe u_j. This probability $P(t_k, t_m|u_j)$ for 3-cliques is computed as follows:

$$P(t_k, t_m|u_j) = [(1 - \alpha_d)\frac{freq(t_k * t_m, u_j)}{|u_j|} + \alpha_d \frac{freq(t_k * t_m, C)}{|C|}] \tag{2}$$

where $freq(t_k * t_m, u_j)$ is the frequency of the phrases in utterance u_j that start with t_k and end with t_m with w tokens between them, and similarly $freq(t_k * t_m, C)$ is the count of such phrases in the entire corpus. By varying w, we identify all phrases in which the tokens occur in that order.

The process described above uses cliques to compute similarities among individual constituents t_k of each target utterance $y_i = t_1 t_2 ... t_n$ and all other utterances $u_j \in C$. We need an overall similarity between the target utterance y_i and all other utterances u_j. To compute this overall similarity, we sum over the 2-clique and 3-clique similarities using the following formula adapted from information retrieval (Metzler and Croft, 2005):

$$M(u_j, y_i) = \sum_{c \in G} \lambda_i P(t_k|u_j) + \sum_{c \in G} \lambda_d P(t_k, t_m|u_j) \tag{3}$$

where λ_i is the weight of 2-cliques and λ_d is the weight of 3-cliques. These similarities between utterances are then placed into a matrix M.

4.3. Clustering

The similarity results obtained as described above are used as the distance metrics for clustering dialogue acts. For query-likelihood clustering, each utterance that is present in the similarity list (above the defined similarity threshold) is represented as a 1, while the others are represented with a value of 0 (Ezen-Can and Boyer, 2013). For MRF-based clustering, each utterance is represented by a vector where similarities are obtained from probabilities in the Markov Random Field model, the matrix M described above. In this way, each target utterance in the corpus is represented by a vector indicating the utterances that are similar to it. Then the clustering takes the produced matrix as an input to group utterances that are similar to each other. We utilize a widely used clustering algorithm k-medoids (Ng and Han, 1994) for MRF-based clustering. The entire unsupervised dialogue act classification algorithm for MRF-based clustering is depicted in Table 4.

```
Let $D$ be a corpus of utterances $D = u_1, u_2, ..., u_n$. Then the goal is:
$\forall\ u_j \in D$, identify $l_j$ as dialogue act label of $u_j$
Procedure:
    For each utterance $u_j$
        1. Set target utterance to $y_i$
           so that the similarities of $y_i$ to every other utterance will be calculated
        2. Build the Markov Random Field graphs $G$ with the tokens of $y_i$
            as the leaf nodes such that every other utterance in the corpus
            is represented as roots of $G$
        3. Create vector of similarity results indicator variables from $G$ as
           $V_j = (v_1, v_2, ..., v_t, ...v_n)$ such that $v_t$ is obtained from cliques
           formed by target utterance $y_i$ and an utterance $u_j$ from the corpus in G
    Let the total vector be $V_T = (V_1, V_2, ..., V_j, ..., V_n)$
    Return clusters $C = c_1, c_2, ..., c_k$ such that C is the result of $kmedoids(V_T)$
```

Table 4: Markov Random Field-based clustering algorithm.

5 Experiments with Comparisons

The goal of the experiments is to determine whether MRF-based clustering outperforms query-likelihood clustering as hypothesized. Additionally, we compare our implementation against that of the recent approach of Rus et al. (2012), which clusters utterances in an educational corpus via word similarity with Euclidean distance, using a specified number of leading tokens of each utterance. We use accuracy to compare these three models. How to best measure accuracy is a non-trivial question for unsupervised models, but we follow the standard practice of comparing to manual labels, though those labels were not used during model training.

For evaluating the MRF-based dialogue act classification approach, we follow an isomorphic approach to the one used in our prior work on query-likelihood dialogue act classification (Ezen-Can and Boyer, 2013): we first retrieve the similarity results for each utterance, and then send the matrix M of similarities to the clustering algorithm. Then using majority voting, we label each utterance with the most frequent dialogue act tag in its cluster.

Training set accuracy evaluates the ability of the models to match the manual labels for the data on which they were trained. Following standard practice for unsupervised model evaluation (Higashinaka et al., 2011; Joty et al., 2011; Rus et al., 2012), we utilize training set accuracy for comparison of the models. The training set accuracy is computed as the number of utterances correctly classified divided by the total number of utterances in the training set.

We explore varying numbers of clusters and provide accuracies for each of them separately. Figure 4 depicts the training set accuracy results for MRF-based clustering compared to query-likelihood clustering, the Rus et al. approach with 5 leading tokens, as well as the random chance baseline. This random chance baseline is the most frequently occurring dialogue act, Evaluation Question (EQ), at 27.3%. We use this highest frequency class as the random chance baseline, rather than $1/9 = 11.1\%$ which would be the accuracy of random guesses among all tags disregarding their frequencies. Choosing the highest frequency tag is a more stringent baseline because a classifier that always guesses EQ will achieve 27.3% accuracy, higher than 11.1%. As shown in Figure 4, MRF-based clustering outperforms its counterparts substantially, confirming our hypothesis.

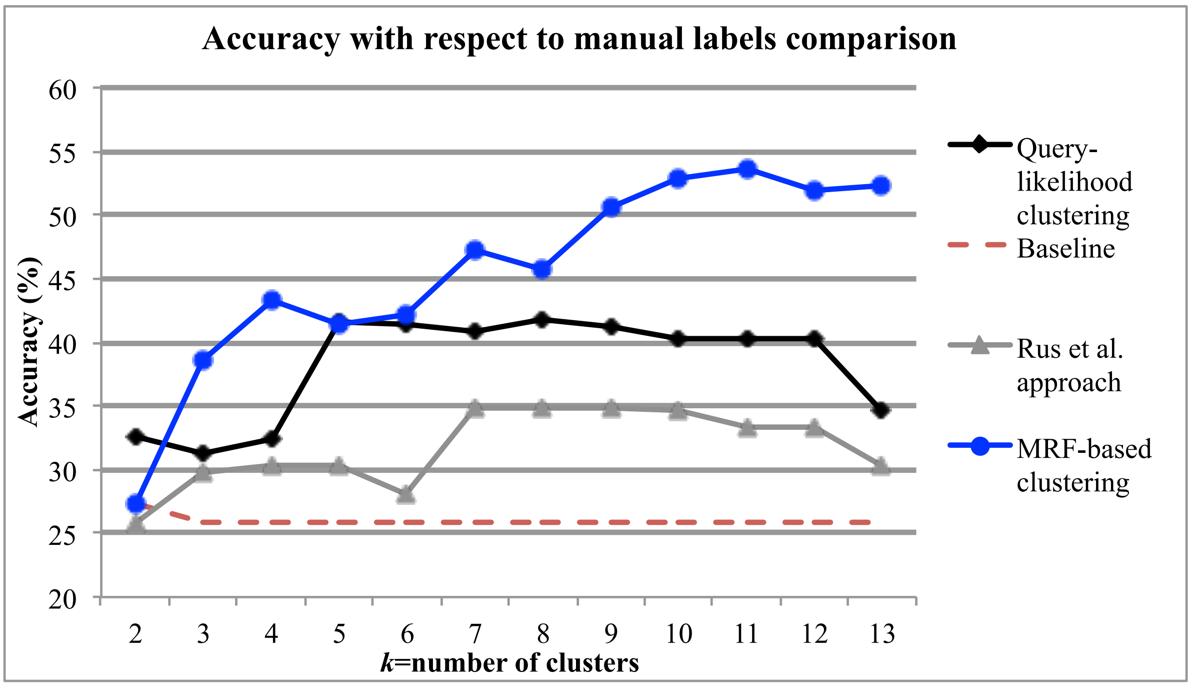


Figure 4: Comparison of two information retrieval inspired methods: query-likelihood clustering and MRF-based clustering as well as the only education-centered unsupervised dialogue act classification work by Rus et al.

6 Evaluation of MRF-Based Clustering

In addition to the training set accuracy used for comparison of different models, we are also interested in how well MRF-based clustering performs on unseen test utterances. To investigate this, this section reports on the selection of number of clusters k, followed by quantitative analyses of *test set accuracy, precision and recall*. Additionally, we examine the distribution of manual dialogue acts across the unsupervised clusters.

In order to determine the number of clusters for the MRF-based model, the Bayesian Information Criterion (BIC) is computed for each value of k. BIC penalizes the number of parameters, which is the number of clusters in our case. Lower BIC values represent a better fit to the data (Chen and Gopalakrishnan, 1998). In our corpus, a model with seven clusters achieved the lowest BIC value (see Figure 5). Note that, the lowest BIC value does not necessarily correspond to the model with highest accuracy because BIC does not consider manual labels, instead it only measures how coherent the clusters are considering distances between data points. The remainder of this section analyzes the seven-cluster model.

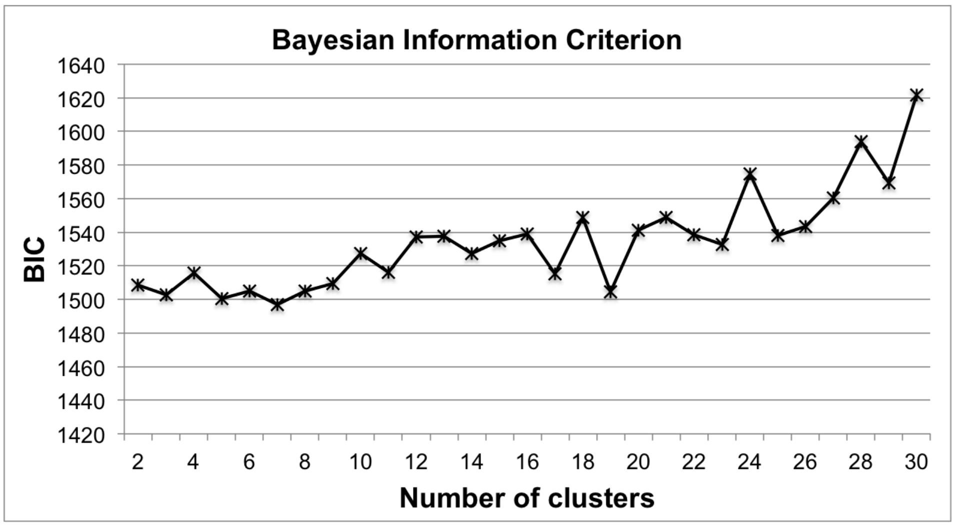
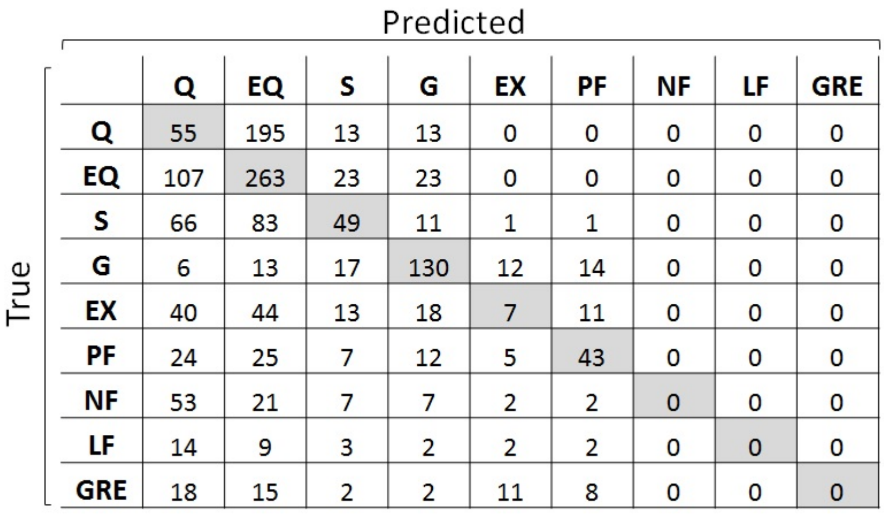


Figure 5: BIC values for varying number of clusters.

6.1. Test Set Accuracy, Precision and Recall

For analyzing the model's performance on unseen test data, we conduct leave-one-student-out cross-validation so that each student's utterances are included in the held-out test set once. In this way, we avoid providing our model with an unfair advantage because the utterances of the same student are not present both for training and testing. To label each held-out utterance, it is assigned to its closest cluster by taking the minimum average distance to all clusters. The majority label of the closest cluster is assigned as the dialogue act label of the test utterance. Then the test set accuracy is calculated as the number of utterances in the test set that are classified correctly, divided by the total number of utterances in the test set. For the MRF-based clustering, the overall accuracy is 36%. The F-measure for MRF-based clustering is 23.2%, with 24.5% precision and 24.0% recall. Because some dialogue acts have never become majority in any of the clusters, they were never predicted by the model (i.e., NF, LF and GRE). Considering only the dialogue act tags that were predicted by the model, the F-measure is 34.8%, with 36.8% precision and 36.1% recall. This performance is still well above baseline. Note that test set accuracy and F-measure were not reported for query-likelihood clustering nor by Rus et al. The confusion matrix in Figure 6 depicts the correct and incorrect predictions for the whole test set.

Similar to the training set evaluations, questions (Q and EQ) are among the most accurately predicted acts in the model. In contrast, there are some dialogue acts (NF, LF, GRE) that are not



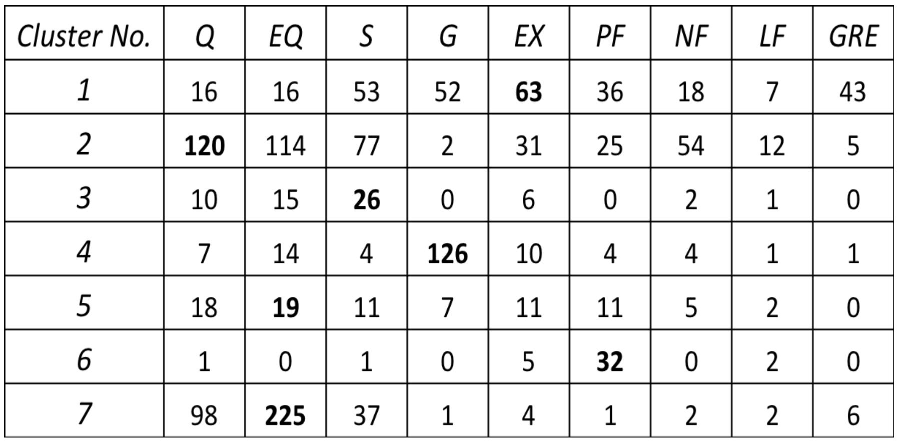
Predicted

True	Q	EQ	S	G	EX	PF	NF	LF	GRE
Q	55	195	13	13	0	0	0	0	0
EQ	107	263	23	23	0	0	0	0	0
S	66	83	49	11	1	1	0	0	0
G	6	13	17	130	12	14	0	0	0
EX	40	44	13	18	7	11	0	0	0
PF	24	25	7	12	5	43	0	0	0
NF	53	21	7	7	2	2	0	0	0
LF	14	9	3	2	2	2	0	0	0
GRE	18	15	2	2	11	8	0	0	0

Figure 6: Confusion matrix for leave-one-student-out cross validation.

predicted in the model. The reason for this is that these acts are so infrequent in the corpus that they were not assigned as the majority label of any cluster and therefore were never assigned to any test utterance.

To understand the performance seen above, we examine the distribution of dialogue acts among clusters for one of the folds of the MRF-based clustering (Figure 7). The majority dialogue act of each cluster is shown in bold.



Cluster No.	*Q*	*EQ*	*S*	*G*	*EX*	*PF*	*NF*	*LF*	*GRE*
1	16	16	53	52	**63**	36	18	7	43
2	**120**	114	77	2	31	25	54	12	5
3	10	15	**26**	0	6	0	2	1	0
4	7	14	4	**126**	10	4	4	1	1
5	18	**19**	11	7	11	11	5	2	0
6	1	0	1	0	5	**32**	0	2	0
7	98	**225**	37	1	4	1	2	2	6

Figure 7: Distribution of dialogue acts over clusters in one of the folds. Bold face indicates the majority dialogue act in each cluster.

The first cluster is mostly composed of extra-domain utterances (utterances that are off-topic) as well as statements and groundings. Because the clustering approaches investigated here do not explicitly use topic information, they group on the surface-level features of utterances, which makes it challenging to distinguish utterances that are not related to the task. The second cluster and seventh cluster are mainly questions, both in the form of Q, a general question, and EQ, an "evaluation" question more specific to the task. In examining the second cluster we see that the model has difficulty differentiating these two question types. For the difficulty in differentiating questions and statements, we observed an interesting point about the corpus. Because the data collection was with novice students, it is very common for these students to use a declarative with a question mark attached such as "while declaring the loop I can use *i* again since it's local right?" and "no wait we need *if* for the conditions right". Given the features used within the models, these questions and statements appear similar in structure, which may explain the model's difficulty in differentiating them. The third cluster has mostly statements, and the model is successful in grouping grounding dialogue acts in the fourth cluster. The fifth cluster is highly impure and its interpretation is not straightforward. Finally, the sixth cluster groups positive feedback utterances together.

6.2. Qualitative Evaluation

In this section, we evaluate the resulting MRF-based clusters qualitatively by examining the utterances grouped in each cluster. Table 5 presents a sampling of utterances from each cluster of the model with seven clusters.

As shown by the examples of cluster 1, many extra-domain utterances and statements are grouped together. From surface-level features, it is difficult to distinguish dialogue acts that may be labeled differently by human coders; for example, "oh" was tagged as extra-domain and "oohhhh" as positive feedback. (Dialogue history is highly influential on different labels of these utterances and as discussed in Section 7, it is important to leverage within dialogue act models.) Likewise, the utterance "beginning now" which was labeled as off-topic is syntactically a statement.

Cluster 2 is mostly composed of questions and evaluation questions. The same finding we noted when examining the distribution of dialogue act tags within clusters is visible here, as both questions and evaluation questions are similar in structure.

The utterances in cluster 3 are generally in the form of statements regardless of their dialogue act tags. For instance, the two utterances, “in a while loop” (which is tagged as question) and “so it would be like assignment” (which is tagged as evaluation question) are syntactically closer to statements than questions when the utterances are considered alone. Once again incorporating dialogue structure so that the model benefits more from the whole dialogue rather than the surface-level features alone is an important challenge moving forward, to address this issue.

It is clear that some clusters are influenced by words: cluster 4 sees many words like “ok” while cluster 6 sees many occurrences of “yes.” In cluster 5, in addition to the questions, some utterances which seem like statements but which were manually tagged as questions appear. For example, “well this array is taking ints and we are putting in characters from a string” has a question tag from manual labeling.

7 Discussion and Limitations

Automatically understanding natural language that students exchange as they are learning is an increasingly prominent problem for educational data mining research. To achieve truly scalable models, unsupervised approaches hold great promise as they aim to address the labor-intensive nature of engineering taxonomies and manual labeling. We described an unsupervised model aimed at modeling student dialogue acts without requiring labor-intensive efforts for annotations and showed that 36% cross-validated test set accuracy is achievable by MRF-based clustering. Compared to a supervised model utilizing an extensive set of features including manually labeled task activities and hidden dialogue states learned by a Hidden Markov Model, which achieved 62.8% accuracy (Boyer et al., 2010), the results obtained by the proposed unsupervised model is promising. As promising as unsupervised models are, they pose important challenges. The dialogue act classifiers presented here have exemplified both the great promise and challenges of unsupervised dialogue modeling.

First, the results illustrate that while dialogue acts are a very useful distinction for educational dialogues, we also need other distinctions such as topic. For example, the dialogue act tag EX indicating off-topic utterances was not distinguished successfully by MRF-based clustering or the prior approaches used for comparison. We have begun to explore combinations of unsupervised dialogue act models with unsupervised topic models, which may address this challenge.

Cluster 1
-oh [*EX*]
-oohhhh [*PF*]
-let me fix this real fast [*S*]
-beginning now [*EX*]
-fair enough [*G*]
Cluster 2
-exactly how would i make an array an instance parameter [*Q*]
-do i set it equal to the conditions of parameter [*EQ*]
-could i just do that with a string [*EQ*]
-ok so how would i add up each digit [*Q*]
-all i remember how to do is convert strings into ints [*LF*]
Cluster 3
-in a while loop [*Q*]
-this should be a string [*EQ*]
-we want it to equal zero so should this be ten [*S*]
-go on [*Q*]
-so it would be like assignment [*EQ*]
Cluster 4
-ok [*G*]
-oh ok [*G*]
-ok like loop [*EQ*]
-ok however [*NF*]
Cluster 5
-is that already declared somewhere [*Q*]
-but since parameter is already given as an int [*Q*]
-is this different than my parameter thing [*Q*]
-and also if my parameter is correct [*EQ*]
-it looks like it is giong to come to that [*S*]
Cluster 6
-yes [*PF*]
-yes giving me definitions to various commands and such [*EX*]
-ohh yes [*EX*]
Cluster 7
-how can you get the sum of parameter [*EQ*]
-ok so it would be like assignment and so on until the last digit [*EQ*]
-what is the name of the postal code in the program [*Q*]
-can the chararray not be used outside because it is in a private method? [*EQ*]
-where are the drawing functions provided [*Q*]

Table 5: Example utterances from each cluster.

Second, the discrimination of dialogue act tags according to whether they referred specifically to the task (e.g., for questions and evaluation questions) was hard for the dialogue act classifier. For example, the data-driven groupings did not distinguish "exactly how would I make an array an instance parameter," which is manually annotated as a question, and "do i set it equal to the conditions of parameter," which is labeled as an evaluation question. In order to address this challenge, we have begun to explore both unsupervised and supervised semantic mapping from utterances to the learning task as a second stage to dialogue act classification. By doing this we may successfully group all questions together in one step, and then determine whether they refer to the task in a second step.

In addition, in this work we showed the promise of MRF-based clustering using only the current student utterance. Because each dialogue utterance is related to its predecessor, considering prior dialogue acts is an important way of representing dialogue history as shown in a supervised dialogue act classification task (Samei, Li, Keshtkar, Rus, and Graesser, 2014). Incorporating context-based features to the model presented in this work is a promising future direction.

8 Conclusion and Future Work

A tremendous amount of educational data is in the form of textual natural language. Whether in tutorial dialogue systems, textual collaborations, or MOOCs, these textual data capture intentions, goals, emotions, and other rich dimensions of human learning interactions. Dialogue act classification is an important step in understanding this natural language dialogue. This article proposed a new unsupervised dialogue act classifier inspired by information retrieval techniques, MRF-based clustering, and compared it to the previous best-performing unsupervised dialogue act classifiers for educational data. Experimental results showed that MRF-based clustering was more successful than its predecessor query-likelihood clustering, likely due to its ability to capture word ordering information.

Several future directions are promising. First, several features successfully utilized in supervised classification techniques remain unexplored for unsupervised models: these include multimodal features when available such as facial expression, posture, gesture, and speech signal. In addition, evaluation of unsupervised models constitutes an important research challenge. Most of the work to date has utilized manual labels as the gold standard. However, the underlying assumption of accepting manual labels as gold-standard imposes unwanted restrictions on unsupervised models. Evaluation techniques that do not depend on manual labels should be investigated in the future; for example, our own future work is placing our best-performing unsupervised dialogue act models within a deployed tutorial dialogue system for end-to-end system evaluation. It is hoped that by moving the field of unsupervised dialogue act modeling forward we will enable better adaptive systems, and better automated understanding of human learning interactions.

9 Acknowledgments

The authors wish to thank to the members of the JavaTutor project, especially James Lester, Eric Wiebe, Bradford Mott, Eunyoung Ha, Christopher Mitchell, Joseph Grafsgaard, and the LearnDialogue group at NC State University. This work is supported in part by the National Science Foundation through Grant DRL-1007962 and the STARS Alliance, CNS-1042468. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the participants, and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

References

Aleven, V., Popescu, O., and Koedinger, K. R. 2001. Towards tutorial dialog to support self- explanation : Adding natural language understanding to a Cognitive Tutor. *Proceedings of Artificial Intelligence in Education*, 246–255.

Allen, J. F., Schubert, L. K., Ferguson, G., Heeman, P., Hwang, C. H., Kato, T., Light, M., Martin, N., Miller, B., Poesio, M., et al. 1995. The TRAINS project: A case study in building a conversational planning agent. *Journal of Experimental & Theoretical Artificial Intelligence 7,* 1, 7–48.

Atapattu, T., Falkner, K., and Falkner, N. 2014. Acquisition of triples of knowledge from lecture notes: A natural language processing approach. In *Proceedings of the International Conference on Educational Data Mining*. 193–196.

Austin, J. L. 1975. *How to do things with words*. Vol. 1955. Oxford university press.

Bangalore, S., Di Fabbrizio, G., and Stent, A. 2008. Learning the structure of task-driven human–human dialogs. *IEEE Transactions on Audio, Speech, and Language Processing 16,* 7, 1249–1259.

Becker, L., Basu, S., and Vanderwende, L. 2012. Mind the gap: learning to choose gaps for question generation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 742–751.

Boyer, K. E., Ha, E. Y., Phillips, R., Wallis, M. D., Vouk, M. A., and Lester, J. C. 2010. Dialogue act modeling in a complex task-oriented domain. In *Proceedings* *of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 297–305.

BOYER, K. E., PHILLIPS, R., INGRAM, A., HA, E. Y., WALLIS, M., VOUK, M., AND LESTER, J. 2011. Investigating the relationship between dialogue structure and tutoring effectiveness: a hidden Markov modeling approach. *International Journal of Artificial Intelligence in Education 21,* 1, 65–81.

BOYER, K. E., VOUK, M. A., AND LESTER, J. C. 2007. The influence of learner characteristics on task-oriented tutorial dialogue. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education (AIED)*. 365–372.

CHEN, S. S. AND GOPALAKRISHNAN, P. S. 1998. Clustering via the Bayesian information criterion with applications in speech recognition. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vol. 2. 645–648.

CORE, M. G. AND ALLEN, J. 1997. Coding dialogs with the DAMSL annotation scheme. In *Proceedings of the AAAI Fall Symposium on Communicative Action in Humans and Machines*. 28–35.

CROOK, N., GRANELL, R., AND PULMAN, S. 2009. Unsupervised classification of dialogue acts using a Dirichlet process mixture model. In *Proceedings of the SIGDIAL 2009 Conference*. Association for Computational Linguistics, 341–348.

DI EUGENIO, B., XIE, Z., AND SERAFIN, R. 2010. Dialogue act classification, higher order dialogue structure, and instance-based learning. *Dialogue & Discourse 1,* 2, 1–24.

D'MELLO, S., OLNEY, A., AND PERSON, N. 2010. Mining collaborative patterns in tutorial dialogues. *Journal of Educational Data Mining 2,* 1, 2–37.

EZEN-CAN, A. AND BOYER, K. E. 2013. Unsupervised classification of student dialogue acts with query-likelihood clustering. In *Proceedings of the International Conference on Educational Data Mining*. 20–27.

EZEN-CAN, A. AND BOYER, K. E. 2014a. A Preliminary Investigation of Learner Characteristics for Unsupervised Dialogue Act Classification. In *Proceedings of the 7th International Conference on Educational Data Mining (EDM)*. 373–374.

EZEN-CAN, A. AND BOYER, K. E. 2014b. Combining task and dialogue streams in unsupervised dialogue act models. In *Proceedings of the 15th Annual SIGDIAL Meeting on Discourse and Dialogue*. 113–122.

FERGUSON, R., WEI, Z., HE, Y., AND BUCKINGHAM SHUM, S. 2013. An evaluation of learning analytics to identify exploratory dialogue in online discussions. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge*. ACM, 85–93.

FORBES-RILEY, K. AND LITMAN, D. J. 2005. Using bigrams to identify relationships between student certainness states and tutor responses in a spoken dialogue corpus. In *Proceedings of the 6th SIGDIAL Workshop on Discourse and Dialogue*. 87–96.

FORSYTH, C. M., GRAESSER, A. C., PAVLIK JR, P., CAI, Z., BUTLER, H., HALPERN, D., AND MILLIS, K. 2013. Operation aries!: Methods, mystery, and mixed models: Discourse features predict affect in a serious game. *Journal of Educational Data Mining 5,* 1, 147–189.

GONZÁLEZ-BRENES, J. P., MOSTOW, J., AND DUAN, W. 2011. How to classify tutorial dialogue? comparing feature vectors vs. sequences. In *Proceedings of the International Conference on Educational Data Mining*. 169–178.

Graesser, A., Person, N. K., and Magliano, J. P. 1995. Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied cognitive psychology 9,* 6, 495–522.

Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W., and Harter, D. 2001. Intelligent tutoring systems with conversational dialogue. *AI magazine 22,* 4, 39.

Higashinaka, R., Kawamae, N., Sadamitsu, K., Minami, Y., Meguro, T., Dohsaka, K., and Inagaki, H. 2011. Unsupervised clustering of utterances using non-parametric Bayesian methods. In *INTERSPEECH*. 2081–2084.

Joty, S., Carenini, G., and Lin, C.-Y. 2011. Unsupervised modeling of dialog acts in asynchronous conversations. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. 1807–1813.

Jurafsky, D. and Martin, J. H. 2000. *Speech & language processing*. Pearson Education.

Klein, D. and Manning, C. D. 2003. Accurate unlexicalized parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 423–430.

Kumar, R., Beuth, J. L., and Rosé, C. P. 2011. Conversational strategies that support idea generation productivity in groups. In *Proceedings of the Computer Supported Collaborative Learning (CSCL) Conference*. 398–405.

Lee, D., Jeong, M., Kim, K., Ryu, S., and Lee, G. 2013. Unsupervised spoken language understanding for a multi-domain dialog system. In *IEEE Transactions On Audio, Speech, and Language Processing*. Vol. 21. 2451–2464.

Litman, D. J., Rosé, C. P., Forbes-Riley, K., VanLehn, K., Bhembe, D., and Silliman, S. 2006. Spoken versus typed human and computer dialogue tutoring. *International Journal of Artificial Intelligence in Education 16,* 2, 145–170.

Manning, C. D., Raghavan, P., and Schütze, H. 2008. *Introduction to information retrieval*. Vol. 1. Cambridge University Press.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics 19,* 2, 313–330.

Marineau, J., Wiemer-Hastings, P., Harter, D., Olde, B., Chipman, P., Karnavat, A., Pomeroy, V., Rajan, S., Graesser, A., Group, T. R., et al. 2000. Classification of speech acts in tutorial dialog. In *Proceedings of the Workshop on Modeling Human Teaching Tactics and Strategies at the Intelligent Tutoring Systems 2000 Conference*. 65–71.

Metzler, D. and Croft, W. B. 2005. A Markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and development in information retrieval*. 472–479.

Mostow, J., Beck, J., Cen, H., Cuneo, A., Gouvea, E., and Heiner, C. 2005. An educational data mining tool to browse tutor-student interactions: Time will tell. In *Proceedings of the Workshop on Educational Data Mining, National Conference on Artificial Intelligence*. 15–22.

Ng, R. T. and Han, J. 1994. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*. 144–155.

Niraula, N. B., Rus, V., Stefanescu, D., and Graesser, A. C. 2014. Mining gap-fill questions from tutorial dialogues. In *Proceedings of the International Conference on Educational Data Mining*. 265–268.

Quarteroni, S., Ivanov, A. V., and Riccardi, G. 2011. Simultaneous dialog act segmentation and classification from human-human spoken conversations. In *IEEE International*

Conference on Acoustics, Speech and Signal Processing. 5596–5599.

RANGARAJAN SRIDHAR, V. K., BANGALORE, S., AND NARAYANAN, S. 2009. Combining lexical, syntactic and prosodic cues for improved online dialog act tagging. *Computer Speech & Language 23,* 4, 407–422.

RICARDO, B.-Y. AND RIBEIRO-NETO, B. 1999. *Modern information retrieval*. Vol. 463. ACM press, New York.

RITTER, A., CHERRY, C., AND DOLAN, B. 2010. Unsupervised modeling of Twitter conversations. In *Proceedings of the Association for Computational Linguistics*. 172–180.

RUS, V., MOLDOVAN, C., NIRAULA, N., AND GRAESSER, A. C. 2012. Automated discovery of speech act categories in educational games. In *Proceedings of the International Educational Data Mining Society*. 25–32.

SADOHARA, K., KOJIMA, H., NARITA, T., NIHEI, M., KAMATA, M., ONAKA, S., FUJITA, Y., AND INOUE, T. 2013. Sub-lexical dialogue act classification in a spoken dialogue system support for the elderly with cognitive disabilities. In *Proceedings of the Fourth Workshop on Speech and Language Processing for Assistive Technologies*. 93–98.

SAMEI, B., LI, H., KESHTKAR, F., RUS, V., AND GRAESSER, A. C. 2014. Context-based speech act classification in intelligent tutoring systems. In *Proceedings of International Conference on Intelligent Tutoring Systems*. 236–241.

SEARLE, J. R. 1969. *Speech acts: An essay in the philosophy of language*. Cambridge university press.

SERAFIN, R. AND DI EUGENIO, B. 2004. FLSA: Extending latent semantic analysis with features for dialogue act classification. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. 692–699.

SHRIBERG, E., STOLCKE, A., JURAFSKY, D., COCCARO, N., METEER, M., BATES, R., TAYLOR, P., RIES, K., MARTIN, R., AND VAN ESS-DYKEMA, C. 1998. Can prosody aid the automatic classification of dialog acts in conversational speech? *Language and speech 41,* 3-4, 443–492.

STEFANESCU, D., RUS, V., AND GRAESSER, A. C. 2014. Towards assessing students' prior knowledge from tutorial dialogues. In *Proceedings of the International Conference on Educational Data Mining*. 197–200.

STOLCKE, A., RIES, K., COCCARO, N., SHRIBERG, E., BATES, R., JURAFSKY, D., TAYLOR, P., MARTIN, R., VAN ESS-DYKEMA, C., AND METEER, M. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics 26,* 3, 339–373.

STROHMAN, T., METZLER, D., TURTLE, H., AND CROFT, W. B. 2005. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*. Vol. 2. 2–6.

TRAUM, D. R. 1999. Speech acts for dialogue agents. In *Foundations of Rational Agency*. Springer, 169–201.

WEERASINGHE, A., MITROVIC, A., AND MARTIN, B. 2009. Towards individualized dialogue support for ill-defined domains. *International Journal of Artificial Intelligence in Education 19,* 4, 357–379.

WEN, M., YANG, D., AND ROSÉ, C. P. 2014. Sentiment analysis in MOOC discussion forums: What does it tell us? In *Proceedings of the International Conference on Educational Data Mining*.

XIONG, W. AND LITMAN, D. 2014. Evaluating topic-word review analysis for understanding student peer review performance. In *Proceedings of the International Conference on Educational Data Mining*.

XU, X., MURRAY, T., WOOLF, B. P., AND SMITH, D. 2013. Mining social deliberation in online communication–if you were me and I were you. In *Proceedings of the International Conference on Educational Data Mining*.

YOO, J. AND KIM, J. 2014. Capturing difficulty expressions in student online Q&A discussions. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 208–214.

ZHAI, C. AND LAFFERTY, J. 2001. A study of smoothing methods for language models applied to ad-hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 334–342.