A Preprocessing Design Scheme for Sequential Pattern Analysis of a Student Database

R. Campagni, D. Merlini and M. C. Verri
Dipartimento di Statistica, Informatica, Applicazioni, Università di Firenze, Viale Morgagni 65, 50134, Firenze, Italia

Keywords: Educational Data Mining, Sequential Pattern Analysis, SQL Preprocessing.

Abstract: In a data mining project evolved on a relational database often a significant effort needs to be done to construct the data set for the analysis. In fact, usually the database contains a series of normalized tables that need to be joined, aggregated and processed in an appropriate way to build the data set. This process generates various SQL queries that are written independently of each other, in a disordered manner. In this way, the database grows with tables and views which are not present at the conceptual level and this can yield problems for the development of the database. In this paper we consider a typical database containing data about students, courses and exams and illustrate some SQL transformations to build a data set to perform a sequential pattern analysis eventually combined with clustering and classification. In particular, we introduce in the student database some interesting patterns representing relationship between the exams given by students in various periods and the career of each student. This is achieved by introducing a particular encoding of a the career of a student. The resulting table can be analyzed with clustering and classification algorithms. We present a case study following this organization.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

In the field of Educational Data Mining, we are often concerned with transactional data managed by a relational database that needs to be reorganized for the analysis with data mining algorithms, such as clustering, classification and association rules and frequent patterns mining. This very expensive preprocessing phase is crucial for the accuracy of the analysis but can generate a disorganized growth of the database (see (Baker, 2014; Peña-Ayala, 2014; Romero et al., 2014; Romero and Ventura, 2013) for recent surveys on the state of the art of educational data mining and on preprocessing educational data). In fact, in order to apply data mining techniques many join and aggregation operations need to be performed which result in a de-normalization of the database with new tables and/or views not present at the conceptual level. The ideal would be to have already at conceptual level a clear idea of the information that must be present in the data set that will be used for the data mining analysis. At the relational logic level this goal can be achieved directly through SQL language[1], organizing a clean SQL script with appropriate queries, views, and/or temporary tables that correspond to the creation of the data set of interest.

The most widely used tools for data mining provide filters to transform data for clustering, classification and the search of association rules, but as far as we know, it is less common to find features that allow to prepare data for analysis based on sequential patterns (see, e.g., (Dong and Pei, 2007; Tan et al., 2006)). This kind of analysis has been used in the context of educational data mining mainly in computer-based environments (see, e.g., (D'Mello et al., 2010; Martinez et al., 2011; Soundranayagam and Yacef, 2010)). A less common application is the use of sequential patterns to analyze data such as courses and exams of university students, with the aim of understanding how the way in which students implement their studies influences their results; a study of this type has been recently done in (Campagni et al., 2015a). Similar goals can also be found in previous work. For example, the analysis of the paths taken by students has been studied in (Ohland et al., 2004) to understand the factors that lead students to leave their study; this is done by using a model defined upon checkpoints in the curriculum in such a way that different curricula can be interpreted through a common framework. A regression model is

[1] In this paper we will refer to the MySQL (MySQL) implementation of the standard SQL

Campagni, R., Merlini, D. and Verri, M.
A Preprocessing Design Scheme for Sequential Pattern Analysis of a Student Database.
In *Proceedings of the 8th International Conference on Computer Supported Education (CSEDU 2016) - Volume 2*, pages 99-106
ISBN: 978-989-758-179-3
Copyright © 2016 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

used in (Zhang et al., 2004) to explore the relationship between graduation and demographic and academic characteristics.

In this paper, by following an approach similar to that recently presented in (Ordonez et al., 2014), we propose a methodology for obtaining a data set for a sequential pattern analysis, starting from a classical relational database about students, courses and exams and working at the SQL level.

In order to present our preprocessing design scheme, let us consider a typical relational database containing data about university students, courses and exams, as illustrated in Tables 1, 2 and 3. In particular, table Students contains the identification code of a student, studentid, the year of enrollment at the university, enrollment, the gender, gender, and the high school grade, hgrade; table Courses contains the identification code and the description of a course, courseid and description, as well as the semester in which the course is given by the teacher, semester, and the corresponding number of credits, cfu; finally, table Exams contains the information about the date and grade, date and grade, each student obtains in the examinations. Of course, other information could be added to the tables, for example the evaluation of courses taken by students and their results in the corresponding exams, as recently studied in (Campagni et al., 2014; Campagni et al., 2015b).

Table 1: A sample Students table.

studentid	enrollment	gender	hgrade
10	2012	M	85
20	2012	F	98
30	2012	F	75

Table 2: A sample Courses table.

courseid	description	semester	cfu
1	Calculus	1	12
2	Programming	1	12
3	Algorithms	1	9
4	Databases	2	9
5	Statistics	2	6
6	Data Mining	2	6

If we wanted to study the correlations among the exams given by students through the research of association rules with the Apriori algorithm (Tan et al., 2006), we would probably need to organize data as in Table 4, for example if we were interested in showing exams conditions that occur frequently together (e.g., many students who gave exams Statistics and Database then gave Data Mining). The organization in Table 4 corresponds to the input format for the Apriori implementation in the Weka system

Table 3: A sample Exams table.

studentid	courseid	date	grade
10	1	2013-06-10	28
10	2	2013-09-24	26
20	1	2013-01-21	28
20	2	2013-02-11	30
20	3	2013-02-25	27
20	5	2013-06-10	28
20	4	2013-07-15	30
20	6	2014-01-22	30
30	2	2013-01-21	24
30	4	2013-09-20	26

Table 4: A sample table for association rule mining. C: Calculus, P: Programming, A: Algorithms, DB: Databases, S: Statistics, DM: Data Mining.

studentid	C	P	A	DB	S	DM
10	1	1	0	0	0	0
20	1	1	1	1	1	1
30	0	1	0	1	0	0

(Witten et al., 2011), which provides filters to menage the original data. However, such organization can be achieved directly with a quite simple SQL query which performs a natural join among the tables Students, Exams and Courses and aggregates on each student by using the appropriate aggregate operator and conditional function. In Table 5 we illustrate the query which select data for Table 4.

The search of association rules in a table organized as in Table 4 allows to determine some possible conditions existing between groups of exams taken by students. These rules, by their nature, do not take into account any temporal information and the involved exams may have been taken by students at any time and in any order. If we want to monitor the order in which the exams are taken, it is necessary to search rules corresponding to sequential patterns by introducing the temporal information associated with the events of interest, in our case the exams. As we will illustrate in the next sections, in order to prepare data for sequential pattern analysis it is necessary to make a non trivial preprocessing on the database. We will show how to achieve this goal by working at SQL level on the original relational scheme. We wish to point out that the approach presented in this paper could be easily adapted to contexts different from those of university education.

2 THE PREPROCESSING SCHEME

Before introducing our preprocessing scheme, we

Table 5: A MySQL query corresponding to Table 4.

```
select studentid,
sum(if(description="Calculus", 1, 0)) as Calculus,
sum(if(description="Programming", 1, 0)) as Programming,
sum(if(description="Algorithms", 1, 0)) as Algorithms,
sum(if(description="Databases", 1, 0)) as Databases,
sum(if(description="Statistics", 1, 0)) as Statistics,
sum(if(description="Data Mining", 1, 0)) as DataMining
from Students natural join Exams natural join Courses
group by studentid;
```

recall the basic concepts related to sequential patterns analysis (see, e.,g., (Tan et al., 2006)). A *sequence* is an ordered list $s = \langle s_1 s_2 \dots s_m \rangle$, where each s_j is a collection of one or more *events*, i.e., $s_j = (e_1, e_2, \dots, e_{k_j})$. The events in s_j correspond to the same temporal information, that is, they occur at the same time. We say that sequence s has length m, while a k-sequence is a sequence that contains k events. A sequence t is a *subsequence* of another sequence s if each ordered element in t is a subset of an ordered element in s. Given a data set containing one or more sequences, the *support* of a sequence s is the fraction of sequences that contain s. The aim of sequential pattern analysis is to find all sequences with support greatest or equal to a user-specified minimum support threshold; those sequences are called *frequent patterns*. We illustrate these concepts by referring to a relational database organized as illustrated in Tables 1, 2 and 3.

Each exam is typically associated with the date on which it was taken by the student, but this temporal information does not allow to compare the careers of different students, as, potentially, the exams can be taken on different dates. We need to perform a time discretization to give the same temporal information to exams taken in the same dates interval, taking into account the organization of the university context under consideration. For example, referring to Tables 1, 2 and 3, we can introduce the discretization illustrated in Table 6 to associate the interval between the attributes `fromdate` and `todate` to the temporal information `semester`. In this way, the exam corresponding to `studentid` 10 and `courseid` 1 is matched with the `semester` 1 while that of `studentid` 20 and `courseid` 6 is matched with `semester` 3. We wish to point out that, in this example, the `semester` of the course in Table 2 corresponds to a value of `semester` in Table 6, that is, these two values correspond to the same dates interval.

By using the formalism of sequential patterns just introduced, the exams of students identified by 10, 20 and 30 correspond to the sequences $\langle(1\ 2)\rangle$, $\langle(1\ 2\ 3)(4\ 5)(6)\rangle$ and $\langle(2)(4)\rangle$ of length 1, 3 and 2, respectively. This means that student 10 has taken exams 1 and 2 in the same `semester`, while student 20 has given exams 1, 2 and 3 in the same `semester`, then has given exams 4 and 5 in a later `semester` and, finally, exam 6 in another `semester`. In this data set, the pattern $(1\ 2)$ is verified by two students, that is, students 10 and 20 gave exams 1 and 2, each in a same semester. In other words, $(1\ 2)$ is a subsequence of sequences $\langle(1\ 2)\rangle$ and $\langle(1\ 2\ 3)(4\ 5)(6)\rangle$, therefore it is a frequent pattern with support 66%.

There are several algorithms implementing techniques for finding frequent patterns based on the *Apriori* principle (Tan et al., 2006). In this paper, we refer to the `CloSpan` algorithm (CLOSPAN; Yan et al., 2003) which finds *closed patterns*, that is, those patterns containing no super-sequence with the same support. In the previous example, the algorithm does not find the sequence (1) which is a subsequence of $(1\ 2)$ with the same support. These algorithms generally take as input a data set similar to that illustrated in Table 7, containing records which correspond, in our example, to the attributes `studentid`, `period` and `courseid`. In particular, the blue values correspond to the discretization of the attribute `date` in Table 3 via Table 6; the red values are an alternative temporal information, which give the delay with which a student takes an exam with respect to the period in which the corresponding course has been given by the teacher, and can be computed by the difference between the `period` of the exam and the `semester` of the corresponding course. Once the data set illustrated in Table 7 has been processed, we can use the `CloSpan` algorithm specifying a value of support to find the frequent patterns verified by a percentage of students greater than the given support. An expert of

Table 6: A sample `Semesters` table.

semester	enrollment	fromdate	todate
1	2012	2012-10-01	2013-02-28
2	2012	2013-03-01	2013-09-30
3	2012	2013-10-01	2014-02-28
4	2012	2014-03-01	2014-09-30
5	2012	2014-10-01	2015-02-28
6	2012	2015-03-01	2015-09-30

Table 7: A sample ExamsSequentialAnalysis table: in red the alternative to the blue temporal information.

studentid	period	courseid	
10	2	1	1
10	2	2	1
20	1	1	0
20	1	2	0
20	1	3	0
20	2	5	0
20	2	4	0
20	3	6	1
30	1	2	0
30	2	4	0

the context under examination will be then responsible to select the most interesting patterns and we will have to decide how to use this information for data mining analysis on students.

In this paper, we start from a quite general relational scheme and describe how to implement, by using SQL, the actions just introduced as well as how to use the most interesting patterns determined by the algorithm. Figure 1 illustrates the logical scheme of our student database. Compared to the example shown above, we can see that the tables Students and Courses contain new attributes and that many others might be considered, as already observed in the Introduction. In particular, the attribute hschool corresponds to the type of high school and the attribute testgrade is the grade obtained by a student in the entrance test at the University. In table Courses, the attribute number is a number assigned by an expert of the domain, taking into account the organization of the course of study. For example, the courses can be numbered in the order in which they are given by teachers and by taking into account the prerequisites among them. Generally, this number does not match the identification code of the course. For the sake of simplicity, in the example of Table 2, those attributes coincide.

In Figure 2 we give the scheme of the tables which are used for transforming the initial scheme. Table Semesters is organized exactly like the sample table described in Table 6 and, as already observed, is important to associate a discrete value to the date of each exam. Table ExamsProcessed adds to the table Exams the attributes cfu, period and delay. The first one is the attribute cfu of table Course and can be added with a simple join operation. The second attribute correspond to the period of examination and is computed by using a query similar to that illustrated in Table 8[2]. Finally, attribute delay is computed as the difference between the period just computed and the semester in Courses. This table could be analyzed on its own, if we want to focus on courses and their characteristics or can be used as the starting point for computing the next tables of Figure 2. Table ExamsSequentialAnalysis is organized like the sample table described in Table 7 and can be obtained by a simple join between ExamsProcessed and Courses followed by a projection. This is the table used as input for the CloSpan algorithm.

After selecting the most significant frequent patterns, we want to keep track of such patterns by inserting in the Students table a Boolean value that allows to determine if the pattern is verified by the student or not. To manage this operation through SQL language, we introduce a representation of student careers that allows us to easily select the students who verify a certain pattern. For example, the career of the student corresponding to studentid 20, is represented by the string (1: 1 2 3)(2: 4 5)(3: 6) which highlights the exams taken in semesters 1, 2 and 3. To achieve this encoding, we use the service table ExamsInPeriod, of which we have an example in Table 9; in practice, it can be realized through a temporary table or a view. The encoding for the student career can be finally obtained aggregating data of table ExamsInPeriod with respect to the studentid and by concatenating appropriately the sequences of exams taken in the same period. With MySQL this can be achieved by the GROUP_CONCAT operator, as described in Table 10.

The most important table for the analysis is represented by the StudentsProcessed table, that enriches Students table with the following attributes: the average grade obtained by the student on all exams, weighted with respect to the credits, avggrade, the total number of credits, credits, the encoding of the career in the form just described, career, and one or more attributes associated with one or more patterns that indicate if the student verifies the patterns or not. In Figure 3, to simplify, we reported only a pattern attribute of this type. In Table 11 we give the StudentsProcessed table for our initial sample tables (a_i for $i = 1, \cdots, 7$ are abbreviations for attributes studentid, enrollment, gender, hgrade, avggrade, credits and career, respectively, while p_1 corresponds to the pattern (1 2)). The values of the attribute pattern may be calculated using the operator rlike, as shown in the query of Table 12. In particular, the jolly operator . matches any character and $*$ corresponds to the Kleene star operator for regular expressions.

We wish to emphasize the step described in Tables 10 which corresponds to a crucial encoding of the ca-

[2]Tables from now on are displayed at the end of the paper

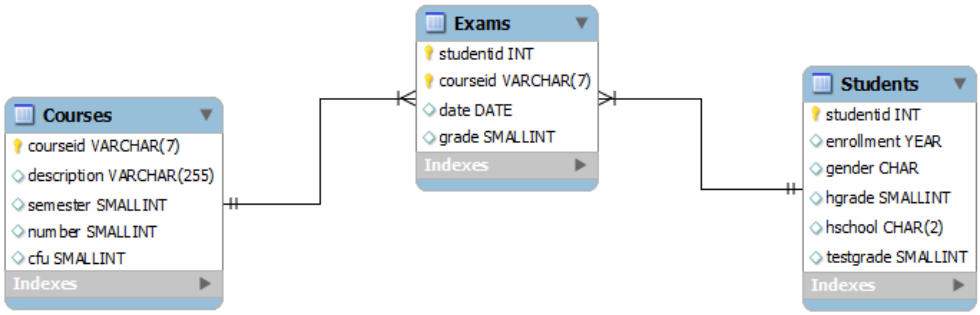

Figure 1: The initial logical schema.

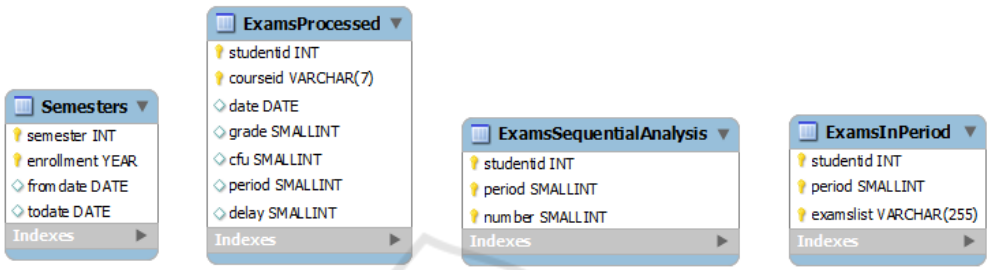

Figure 2: The transformation tables.

Table 8: A MySQL query corresponding to Table 7.

```
select studentid,
(select semester from Semesters where exams.date>= fromdate
and exams.date<= todate) as period, courseid
from Exams;
```

Table 9: The table ExamsInPeriod for Table 7.

studentid	period	esamslist
10	2	1 2
20	1	1 2 3
20	2	4 5
20	3	6
30	1	2
30	2	4

Table 10: A MySQL query corresponding to Table 9.

```
select studentid,
group_concat('(',convert(period,char(2)),':', examslist,')'
order by period SEPARATOR ' ') as career
from ExamsInPeriod group by studentid;
```

Table 11: The sample table StudentsProcessed.

a_1	a_2	a_3	a_4	a_5	a_6	a_7	p_1
10	2012	M	85	27	24	(2: 1 2)	1
20	2012	F	98	29	54	(1: 1 2 3) (2: 4 5) (3: 6)	1
30	2012	F	75	25	21	(1: 2) (2: 4)	0

reer of a student and the next step described in 12 , which allows to search if a given frequent pattern occurs in the career.

The table StudentsProcessed can be analyzed using clustering techniques and/or classification algorithms. In the next section we will show an analysis of this type on a real case study.

Table 12: The MySQL to insert a value in the attribute pattern of Table 11.

```
update StudentsProcessed
set pattern=1 where studentid in
(select studentid from StudentsProcessed
where career rlike '.*(.*:1 2.*).*');
```

Table 13: The 21 careers of students verifying pattern (1 3 4)(11 12).

(2:1 2 3 4 5 6) (3:7 8 9 10) (4:11 12 13 14)
(2:1 2 3 4 5 6) (3:7 8 9 10) (4:11 12 13 14)
(2:1 2 3 4 5 6) (3:7 8 9 10) (4:11 12 14)
(2:1 2 3 4 5 6) (3:7 8 9 10) (4:11 12 14)
(2:1 2 3 4 5 6) (3:8 9 10) (4:7 11 12 14)
(2:1 2 3 4 5) (3:6 7 8 9 10) (4:11 12 14)
(2:1 2 3 4 5) (3:7 8) (4:9 10 11 12 14)
(2:1 2 3 4 5) (3:8 9 10) (4:7 11 12 13)
(2:1 2 3 4 6) (3:5 7 8 10) (4:9 11 12 14)
(2:1 2 3 4 6) (3:5 8 9) (4:11 12 14)
(2:1 2 3 4 6) (3:8 9 10) (4:5 11 12 14)
(2:1 2 3 4) (3:5) (4:9 11 12)
(2:1 3 4 5 6) (3:2 9) (4:7 10 11 12 13 14)
(2:1 3 4 5 6) (3:8 9 10) (4:11 12 14)
(2:1 3 4 5) (3:2 8 9 10) (4:7 11 12 14)
(2:1 3 4 6) (3:8) (4:10 11 12)
(2:1 3 4) (3:10) (4:11 12)
(2:1 3 4) (3:2 5 8) (4:6 11 12)
(2:1 3 4) (3:2 8 10) (4:9 11 12)
(2:1 3 4) (3:5 10) (4:2 9 11 12 14)
(2:1 3 4) (4:8 11 12)

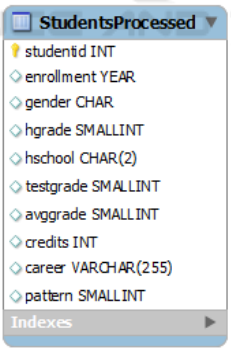

Figure 3: The final table.

3 THE CASE STUDY

Our case study corresponds to a photo shoot in September 2014 of the students enrolled on 2012 at the Computer Science degree of an Italian University. This degree is organized into three years and six semesters, that is, two semesters for year. The semesters are organized as described in Table 6, therefore, we are dealing with students who attended the courses of the first two years, just before the beginning of the third. In the first year students have 6 annual courses which are given by teachers from October to May and at the end they can take the corresponding exams; therefore, in practice, students start to take exams in semester 2, according to Table 6. The second year includes 4 courses both in the first and second semesters (number 3 and 4 of Table 6, respectively), for a total of 14 courses. In general, with our choice of semesters, students can start to take an exam in the same semester in which the course was held by the teacher, with a null delay, or take the exam in a following semester, accumulating a delay that, in principle, can grow indefinitely. We numbered the exams from 1 to 14 according to the year, the semester and the more or less explicit prerequisites among them. At the time of the photo, 56 students took at least an exam and we analyzed their career by using the methodology illustrated in the previous section. The data set is not large, however, as focused in (Natek and Zwilling, 2014), a data mining analysis is useful also in such small contexts. Moreover, the case study allows us to describe the methodology on a real situation.

The data of the students were first organized in a database organized as in Figure 1, then we con-

structed table ExamsSequentialAnalysis by following the steps previously described and used the CloSpan algorithm to identify the possible frequent patterns existing among the exams taken by students. By running the algorithm with support 30%, we found 66 patterns which we ordered according to the length and to the number of students verifying them.

As an example, we considered the pattern which maximizes the product between the number of exams and the number of students, that is the sequence (1 3 4) (11 12) verified by 21 students. Numbers 1, 3, 4, 11 and 12 correspond to *Calculus I*, *Programming*, *Algorithms*, *Calculus II* and *Databases*, respectively. In particular, courses 1, 3 and 4 are annual courses of the first year, while 11 and 12 are two courses of the second semester of the second year. The pattern tells us that the 21 involved students took the exams 1, 3 and 4 in the same semester and exams 11 and 12 in a following semester, but it does not tell us nothing about the semester; in principle, each student could correspond to a different pair of semesters. However, by coding the career of the students as described in Table 11 and adding the pattern to the table StudentProcessed, we can easily verify that all 21 students gave the exams 1, 3 and 4 in the semester 2 and exams 11 and 12 in semester 4, that is, without delay, as illustrated in Table 13.

The pattern shows that courses 1, 3, 4, 11 and 12 seem to be organized in such a way to bring the students to take the exams immediately at the end of the course, while there is a greater dispersion for the exams of the first semester of the second year, as shown in Table 13. Previous facts also mean that if we used the delay as temporal information (red values in the Table 7) we would find the pattern (1 3 4 11 12) verified by 21 students with zero delay.

After choosing the most interesting patterns, with the help of an expert of the domain, and computing the table StudentProcessed, we proceeded with the data mining analysis. First, we used the Weka implementation of the K-means algorithm, with $K = 2$, and obtained the following two clusters, according to attributes testgrade, avggrade and credits.

Attribute	Full Data (56)	0 (29)	1 (27)
testgrade	14.2321	15.1724	13.2222
avggrade	24.9286	26.1034	23.6667
credits	73.5179	97.6552	47.5926

Cluster 0 contains students which have reached better results, in terms of exam grades and number of credits, than those in cluster 1; moreover, students in cluster 0 attained a better grade in the entrance test (with possible values in the range $[0..25]$).

All students which verify the pattern are in cluster 0; belong to the same cluster also students who do not verify the pattern but obtained a quite good grade at the high school, independently from the typology of the school. Analyzing in more details the careers of students who do not verify the pattern but belong to cluster 0 we observed that they had some difficulties with the first year exams *Calculus I* and *Programming*, taking them with one semester of delay but then recovering in the subsequent semesters.

The results obtained in this small case study allow us to point out that the frequent patterns analysis introduced in this paper can bring out some unusual relationships related to the way in which students face exams. Obviously, once table StudentProcessed has been processed other data mining techniques can be used to analyse data.

4 CONCLUSIONS

In any data mining project a large part of the work is related to process the data in order to obtain one or more data sets on which applying the data mining algorithms. In particular, our attention focused on the preparation of data for the analysis and the use of frequent patterns. We presented a preprocessing design scheme based on SQL queries and on a particular encoding of the student career that, starting from a traditional university database, allows to obtain the data set which can later be analyzed by techniques of clustering and classification in order to highlight possible relations among the exams taken by students; these relationships take into account the associated temporal information. The frequent pattern analysis can highlight some behaviors which may seem counterintuitive, for example, course e_i is scheduled before course e_j while many students take exam e_j before e_i or can put in evidence which exams tend to be given with greater delay than others. The introduction in the database of the Boolean information about the most significant patterns and clustering techniques can help to understand if students satisfying the patterns have some common characteristics. Another interesting application could be the identification of patterns corresponding to students at risk of dropping out. The design scheme and the encoding of the student career presented in this paper can help to achieve these goals. A delicate point remains the selection of the pattern, for this reason it is crucial the presence of an expert of the context under analysis, also for the validation of the models obtained.

REFERENCES

Baker, R. S. J. D. (2014). Educational data mining: an advance for intelligent systems in education. *IEEE Intelligent Systems*, 29(3):78–82.

Campagni, R., Merlini, D., Sprugnoli, R., and Verri, M. C. (2015a). Data mining models for student careers. *Expert Systems with Applications*, 42(13):5508–5521.

Campagni, R., Merlini, D., and Verri, M. C. (2014). Finding regularities in courses evaluation with k-means clustering. In *Proceedings of CSEDU 2014 - the 6th International Conference on Computer Supported Education*, volume 2, pages 26–33.

Campagni, R., Merlini, D., and Verri, M. C. (2015b). An analysis of courses evaluation through clustering. In Zvacek, S., Restivo, M., Uhomoibhi, J., and Helfert, M., editors, *Computer Supported Education*, volume 510 of *Communications in Computer and Information Science*, pages 211–224. Springer International Publishing.

CLOSPAN. http://www.cs.ucsb.edu/∼xyan/software/Clospan.htm.

D'Mello, S., Olney, A., and Person, N. (2010). Mining Collaborative Patterns in Tutorial Dialogues. *Journal of Educational Data Mining*, 2(1):1–37.

Dong, G. and Pei, J. (2007). *Sequence Data Mining*, volume 33 of *Advances in Database Systems*. Springer.

Martinez, R., Yacef, K., Kay, J., Al-Qaraghuli, A., and Kharrufa, A. (2011). Analysing frequent sequential patterns of collaborative learning activity around an interactive tabletop. In *Proceedings of EDM 2011, 4th International Conference on Educational Data Mining*, pages 111–120, Eindhoven, the Netherlands.

MySQL. http://www.mysql.com/.

Natek, S. and Zwilling, M. (2014). Student data mining solution-knowledge management system related to higher education institutions. *Expert Systems with Applications*, 41:6400–6407.

Ohland, M. W., Zhang, G., Thorndyke, B., and Anderson, T. J. (2004). The creation of the multiple-institution database for investigating engineering longitudinal development. In *Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition*.

Ordonez, C., Maabout, S., Matusevich, D. S., and Cabrera, W. (2014). Extending ER models to capture database transformations to build data sets for data mining. *Data & Knowledge Engineering*, 89:38–54.

Peña-Ayala, A. (2014). Educational data mining: a survey and a data mining-based analysis. *Expert Systems with Applications*, 41:1432–1462.

Romero, C., Romero, J. R., and Ventura, S. (2014). A survey on pre-processing educational data. In *Educational Data Mining. Studies in Computational Intelligence*, volume 524, pages 29–64, A. Peña-Ayala (Ed.), Springer.

Romero, C. and Ventura, S. (2013). Data mining in education. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 3(1):12–27.

Soundranayagam, H. and Yacef, K. (2010). Can order of access to learning resources predict success? In *Proceedings of EDM 2010, 3rd International Conference on Educational Data Mining*, pages 323–324, Pittsburgh, PA, USA.

Tan, P. N., Steinbach, M., and Kumar, V. (2006). *Introduction to Data Mining*. Addison-Wesley.

Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition*. Morgan Kaufmann.

Yan, X., Han, J., and Afshar, R. (2003). Clospan: Mining closed sequential patterns in large databases. In *Proceedings of the Third SIAM International Conference on Data Mining*, San Francisco, CA, USA.

Zhang, G., Anderson, T. J., Ohland, M. W., and Thorndyke, B. (2004). Identifying factors influencing engineering student graduation: a longitudinal and cross-institutional study. *Journal of Engineering Education*, 93(4):313–320.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS