Course Enrollment Recommender System

Hana Bydžovská
CSU and KD Lab Faculty of Informatics
Masaryk University, Brno
bydzovska@fi.muni.cz

ABSTRACT

One of the main problems faced by university students is to create and manage the semester course plan. In this paper, we present a course enrollment recommender system based on data mining techniques. The system mainly helps with students' enrollment decisions. More specifically, it provides recommendation of selective and optional courses with respect to students' skills, knowledge, interests and free time slots in their timetables. The system also warns students against difficult courses and reminds them mandatory study duties. We evaluate the usability of designed methods by analyzing real-world data obtained from the Information System of Masaryk University.

Keywords

Course enrollment recommender system, student performance, prerequisites, university information system.

1. INTRODUCTION

Recommender systems can be used in different fields including educational environment. Such systems are mainly focused on providing high educational standard and try to enhance the process of teaching and learning [13]. They help with searching for suitable web resources [8], recommend good solutions to improve students' knowledge [4], or analyze data obtained from quizzes and provide a feedback to instructor to modify a quiz [9].

Nowadays, researchers also try to improve personalized searching for beneficial courses. The aim of several projects was to select courses in order to obtain good exam results [12] or recommend elective course modules based on previous students' enrollments using collaborative filtering techniques [6]. Other option is to utilize association rules [1] or ant colony optimization [11].

In the last few years, recommendations became more complex. Besides selecting passable courses, it is essential to recommend beneficial courses [3]. The suitability of courses was determined by the importance in all fields of the university, the ratio of connectivity among courses and by the importance in the student's field of study. Association rules were utilized for searching relationships between courses. Another approach was presented in [7]. To graduate, all defined blocks of courses must be completed by finishing a pre-defined number of courses. They utilized a flow algorithm to find the minimal set of courses that students have to pass.

In this paper, we present a pilot version of the course enrollment recommender system designed at the Faculty of Informatics Masaryk University. All methods were validated on data originated from the Information System of Masaryk University (IS MU). The data contain information on courses, templates defining the mandatory and selective courses, students, study-related attributes, and social behavior data. The designed methods predict students' final grades and recommend them interesting courses with respect to their skills, interests, and free time-slots in the timetable.

2. COURSE ENROLLMENT RECOMMENDER SYSTEM

2.1 Motivation

All students have to follow the obligations and principles stated by their university. Especially at the beginning of the study, it is hard for students to cover all the mandatory duties. At Masaryk University, all semesters are preceded by a course enrollment process. All active students have to enroll a sufficient number of courses to achieve at least the minimal pre-defined amount of credits. If they do not reach the minimum limit, they cannot proceed to the next semester. Students have to pass many courses before finishing their studies successfully. All mandatory courses must be completed. Students have to also pass several selective and optional courses. Analyzing the enrollment statistics, we found out that students prefer interesting and passable courses. Universities usually offer a large number of courses and it is difficult for students to be familiarized with all of them. They are forced to search through the entire course catalog, read many abstracts and syllabi, and compare a large amount of success rate statistics. Naturally, they often discuss courses with other students who have their own personal experiences. Obviously, the decisions they have made during the course enrollment process could significantly influence the whole study progress and the final result.

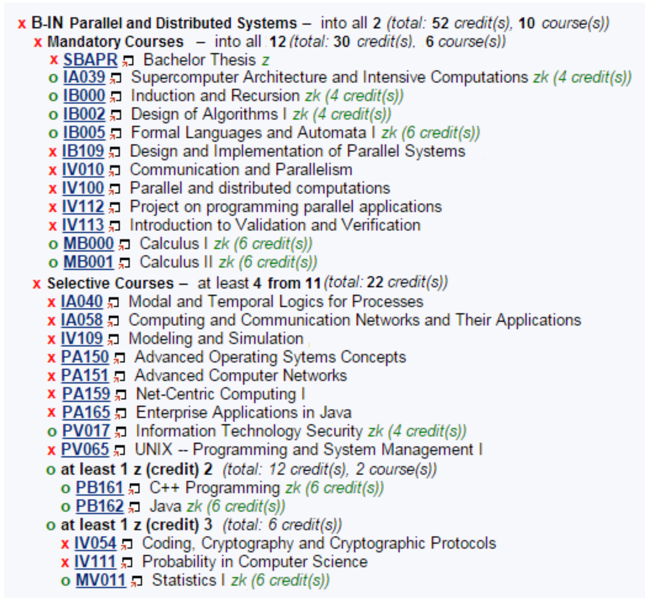
2.2 System Overview

The current version of the recommender system monitors the number of credits of enrolled courses to ensure successful progression to the next semester. It also reminds them to enroll all mandatory courses. Selective and optional courses are recommended according to the student's performance and interests with respect to free time slots in students' timetables. The system clarifies the decisions to students using notifications. The system also warns against enrolled courses that usually cause problems to students with similar characteristics. If the system identifies a difficult course in the student's enrollment, it informs the student about the potential issue. It allows students to focus more on this course or to revise the enrollment decision. Students can also assess each recommendation whether the recommended courses were interesting and adequately difficult. Based on these assessments, the recommendation algorithms will be modified in order to enhance the relevance of the further recommendations.

3. COURSE TEMPLATES

At our university, templates represent tree-like definitions of mandatory and selective courses for each field of study. The system allows checking the requirements that a student has already accomplished. The completed courses/nodes are marked with a green ring (o) and the uncompleted courses/nodes are marked with a red cross (x).

We examined 67 templates defining the study requirements for active students in the years of 2010-2013 at Faculty of Informatics. An example of a template can be seen in Figure 1.



- x **B-IN Parallel and Distributed Systems** – into all **2** *(total: **52** credit(s), **10** course(s))*
  - x **Mandatory Courses** – into all **12** *(total: **30** credit(s), **6** course(s))*
    - x SBAPR Bachelor Thesis *z*
    - o IA039 Supercomputer Architecture and Intensive Computations *zk (4 credit(s))*
    - o IB000 Induction and Recursion *zk (4 credit(s))*
    - o IB002 Design of Algorithms I *zk (4 credit(s))*
    - o IB005 Formal Languages and Automata I *zk (6 credit(s))*
    - x IB109 Design and Implementation of Parallel Systems
    - x IV010 Communication and Parallelism
    - x IV100 Parallel and distributed computations
    - x IV112 Project on programming parallel applications
    - x IV113 Introduction to Validation and Verification
    - o MB000 Calculus I *zk (6 credit(s))*
    - o MB001 Calculus II *zk (6 credit(s))*
  - x **Selective Courses** – at least **4 from 11** *(total: **22** credit(s))*
    - x IA040 Modal and Temporal Logics for Processes
    - x IA058 Computing and Communication Networks and Their Applications
    - x IV109 Modeling and Simulation
    - x PA150 Advanced Operating Sytems Concepts
    - x PA151 Advanced Computer Networks
    - x PA159 Net-Centric Computing I
    - x PA165 Enterprise Applications in Java
    - o PV017 Information Technology Security *zk (4 credit(s))*
    - x PV065 UNIX -- Programming and System Management I
    - o **at least 1 z (credit) 2** *(total: 12 credit(s), 2 course(s))*
      - o PB161 C++ Programming *zk (6 credit(s))*
      - o PB162 Java *zk (6 credit(s))*
    - o **at least 1 z (credit) 3** *(total: 6 credit(s))*
      - x IV054 Coding, Cryptography and Cryptographic Protocols
      - x IV111 Probability in Computer Science
      - o MV011 Statistics I *zk (6 credit(s))*

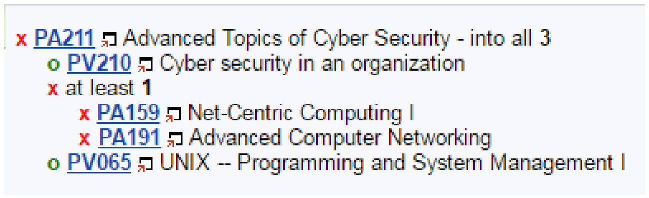
Figure 1. Template of mandatory and selective courses

However, the structure of the templates is often more complicated. Each node defines how many child nodes have to be completed (all, defined by the number of credits, or defined by the number of children). The template does not enforce in which semester courses should be enrolled.

3.1 Which courses do students have to pass before enrolling a certain course?

Some courses have prerequisites that define what a student must meet before he or she can enroll in a certain course. At our university, prerequisites are composed of terms $p_1 \dots p_n$ that are associated with logical operators AND(&&), OR(||). A term p_i can be a course or a compound term. Prerequisites can be transformed into the template subtree by the following rules:

- p_i && p_j → new node containing p_i and p_j with the rule of fulfillment: all nodes
- p_i || p_j → new node containing p_i and p_j with the rule of fulfillment: at least one of nodes



- x PA211 Advanced Topics of Cyber Security - into all 3
  - o PV210 Cyber security in an organization
  - x at least 1
    - x PA159 Net-Centric Computing I
    - x PA191 Advanced Computer Networking
  - o PV065 UNIX -- Programming and System Management I

Figure 2. PA211 prerequisites: PV210 && (PA159 || PA191) && PV065

Example of such transformation can be seen in Figure 2. Each template could be extended by prerequisites courses for students to be able to count on them when creating their study plans.

3.2 When do students have to enroll a certain course?

Students can decide in which semester they enroll in a certain course. All graduate students that completed the template requirements were selected and the semester in which the most of them enrolled in the particular mandatory course was calculated by Algorithm 1. Therefore, we remind courses in the proper semesters with respect to students' completed semesters.

Algorithm 1. Semester Selection

```
Function select_semester(course, template):
sem_max = {sem ∈ semesters | ¬∃sem_2: number_students (sem_2, course, template) > number_students (sem, course, template)}
if (|sem_max| == 1) then
        return sem_max[0];
else if (|sem_max| > 1) then
        return min(sem_max);
else
        return 1;
end if;

Function number_students(semester, course, template):
return the number of students having completed the given template enrolled in the given course in the specific semester;
```

3.3 Which courses are passable for a certain student?

We focused on the problem of predicting the final grade at the beginning of the semester with the emphasis on identifying unsuccessful students. The following grade scale was used: 1 (excellent), 1.5 (very good), 2 (good), 2.5 (satisfactory), 3 (sufficient), 4 (failed or waived). The value 4 represents students' failure; the others represent a full completion.

We present two different approaches in [2]. Both approaches are validated on 138 courses which were offered to students of the Faculty of Informatics of Masaryk University between the years of 2010 and 2013. The first approach is based on classification and regression algorithms that search for patterns in study-related data and also data about students' social behavior. We prove that students' social behavior characteristics improve prediction for a quarter of courses. The second approach is based on collaborative filtering techniques. We predict the final grades based on previous achievements of similar students. We also present the novel approach how to find out which approach is better for which courses. Finally, we are able to correctly identify half of all failures (that constitute less than a quarter of all grades) and predict the final grades only with the error slightly higher than one degree in the grade scale.

Due to the prediction error, we decided to lower the granularity of predictions to the following three classes: excellent (1, 1.5), good (2, 2.5), and bad (3, 4) to prevent the recommendation of difficult courses. As it can be seen in Table 1, the mean absolute error was below 0.5 and due to the high value of sensitivity the most of unsuccessful students were revealed.

The approaches are beneficially utilized in the presented course enrollment recommender system to warn students against difficult courses and to recommend only passable optional courses. Courses with predicted grade better than bad grade are considered as passable for a student.

Table 1. Prediction Evaluation on Test set

Task	MAE	Sensitivity
Grade prediction	0.609	0.436
Excellent / good / bad prediction	0.474	0.899

4. SELECTIVE COURSES

Students can select different sets of selective courses from the template with respect to their skills and the course content. They have to select enough courses to fulfill the node requirements. We were interested in the student behavior, e.g. information about the most preferred courses.

4.1 Designed Recommendation Methods

We defined a course *c* for a student *a* as interesting by the following function:

$$f(a, c) \begin{cases} 1 \text{ if the student } a \text{ attended course } c \text{ or marked it as favorite} \\ 0 \text{ otherwise} \end{cases}$$

This characteristic defined the student's interest in the course. Therefore, each student can be characterized by a set of his or her interesting courses. We designed the following 4 algorithms to recommend courses:

S1. The most selected courses by students with the same template. We were interested in the student behavior, e.g. information about the most preferred courses. We computed the most frequent path of graduate students in the template. We were inspired by a simple ant colony algorithm and marked each node with the number of students that passed through. The path was computed by universal path finding Algorithm 2.

S2. Courses enrolled by similar students. We calculated the similarity between sets of interesting courses for each student and all graduate students that already completed the template. We utilized Jaccard's coefficient. For each student, we selected the most similar students and recommended their courses. We were searching for the proper size of the neighborhood and evaluated $n \in [1; 25]$. When we sorted the courses in the list by their frequency of occurrence in similar student's lists, we also explored how many of them were suitable to be recommended. We examined $x \in [1; 10]$.

S3. Courses taught by favorite teacher. Students' interesting courses were examined and favorite teachers were revealed. We considered all course lecturers and only student's tutors. The teacher's popularity was defined as the sum of all his or her courses which were considered as interesting. Considering the teacher's popularity, we recommended other teacher's courses if his or her popularity was above the threshold (2).

S4. Courses enrolled by friends. We examined students' social behavior characteristics and their mutual cooperation. We focused on statistical data that represented the interaction among students: explicitly expressed friendship, posts and comments in discussion forums, e-mails statistics, publication co-authoring, or files sharing. This information served as the basis for computing social ties among students by means of a sociogram [2]. From this sociogram, we were able to reveal friends ties among students. We recommended courses that friends considered as interesting and belonged to the template.

The algorithms also observed the following rules:

- Courses recommended for a particular student were limited to courses that should be enrolled in the certain semester:
  $$course's\ semester \leq student's\ semester$$
  S*tudent's semester* was defined as the number of commenced semesters and the course's semester was defined as the semester in which other students usually enrolled in the course calculated by Algorithm 1.
- We also did not recommend courses that belonged to the subtree of the template which students had already completed.
- Only courses that could be enrolled in the actual semester were recommended.

Algorithm 2. Finding Path in Template

```
Function process_node (node, template, student):
children ← children of the node;
for each child in children do
        unless (child_computed) then
                process_node(child, template, student);
        end if;
end for;
path; # calculated path
sort children in descending order by the value in the node;
for each child in children do
        path ← child;
        if (node_fulfilled(node, student)) then
                return path;
        end if;
end for;

Function node_fulfilled (node, student):
if (the given node is fulfilled by the given student) then
        return true;
else
        return false;
```

4.2 Recommendation Methods Evaluation

We can assume that students are familiar with the offer of selective courses. Therefore, offline experiments [10] can be suitable approach to evaluate previously mentioned algorithms. All students that enrolled in the semester autumn 2014 and did not complete their templates were selected: 1,444 students in total.

4.2.1 Settings for the algorithm S2

Firstly, we had to evaluate suitable settings for the algorithm S2. Our task was to select suitable courses for students and subsequently detect if they enrolled in them or not. Therefore, the suitable evaluation metrics were precision and recall. To find a balance between precision and recall, the F1 score was also calculated.

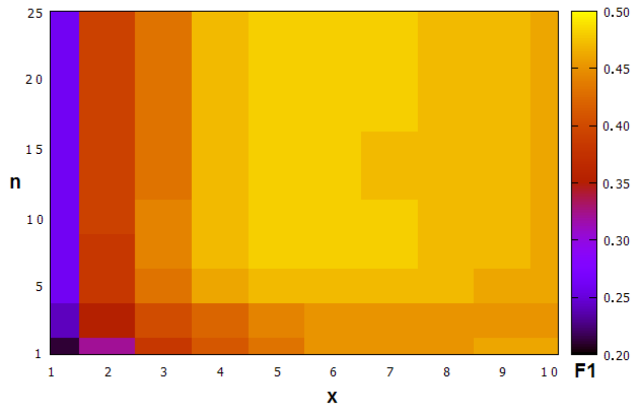


Figure 3. Relationship among the size of the neighborhood *n*, number of selected courses *x* and the value of F1 score

We selected 90% of examined students and calculated the F1 score of the recommendations. Figure 3 shows the relationship among variables *n*, *x*, and the value of F1. Based on these findings, the following setting was selected for algorithm S2 as the most suitable: $n = 8$, $x = 5$. This conclusion was also verified on the test set (the rest 10% of students).

4.2.2 All algorithms' evaluation

We utilized all previously described algorithms to recommend courses for each student. The coverage determines the percentage of students for whom we were able to recommend at least one course.

Table 2. Results of selective courses recommendation

Algorithm	S1	S2	S3	S4
Coverage	0.97	0.63	0.60	0.54
Offered courses	2.97	4.81	3.85	4.43
Enrolled courses in the semester autumn 2014	1.63	2.08	1.81	1.88
Enrolled courses anytime	2.82	3.15	2.49	2.85
Precision	0.81	0.56	0.48	0.47
Recall	0.55	0.42	0.28	0.39
F1	0.66	0.48	0.35	0.43
Rank	**1**	**2**	**4**	**3**

The coverage of approaches differs as it can be seen in Table 2. S1 covered almost all students. In contrary, the rest of approaches recommended courses for only 60% of selected students. The average number of courses offered by each algorithm can be seen in the second row. Algorithms recommended 3-5 courses on average. The average number of courses that students really enrolled in autumn 2014 can be seen in the third row. Because the university does not define when students have to enroll courses, we extend the searching for enrollment also to the next semesters. The average number of courses that students really enrolled anytime from autumn 2014 till now can be seen in the fourth row. As it can be seen, the number of enrolled courses almost doubled in all cases. Finally, we also calculated precision and recall for all algorithms. The algorithm S1 reached the best results.

4.2.3 Which courses are selected the most often?

H1: We supposed that students select easier selective courses.

For finding the easiest way to complete the template, we assessed each course using its success rate (the percentage of successful students to all students in the course). However, we had to penalize courses with a small number of students and also the courses with smart students only (with excellent average grade). Therefore, the adjusted success rate (ASR) was defined as:

$$ASR = CSR \cdot \log_4 ESAG \cdot \frac{NES}{\text{MAX\_ENR}}$$

where CSR defined the course success rate, ESAG defined the average grade of enrolled students, NES defined the number of enrolled students in a course, and MAX_ENR was a constant for the template and defined the maximum number of students enrolled in any course from the template. We calculated the minimal adjusted success rate of courses that have to be passed in the subtree for each node of the template. Subsequently, we employed the Algorithm 2 that selected the easiest courses till the node requirements were met.

For each template $t \in T$ we constructed the easiest path (EP) and also the most frequented path (MFP). Both paths can be represented as a set of selected courses on the path. Jaccards' coefficient (JC) was calculated to compare these sets of courses. The similarity of paths was 0.8 on average for all templates.

$$\frac{\sum_{t \in T} JC(EP, MFP)}{|T|} = 0.8$$

H1 was confirmed. Correlation of EP and MFP over all templates confirmed our hypothesis that students usually select easier selective courses.

5. OPTIONAL COURSES

To fulfill all study requirements, students have to obtain the pre-defined number of credits in their studies. Except credits obtained from mandatory and selective courses, they have to select optional courses. Optional courses for each student were defined as courses that do not belong to the student's template.

5.1 Designed Recommendation Methods

We utilized the same methodology as described in Section 4 for recommendation of selective courses. The main difference was that algorithms did not restrict courses from templates. The courses recommended by algorithms were limited to only passable courses (the predicted grade was not bad) according to the method introduced in Section 3.3.

S1. **The most selected courses by students with the same field of study.** All optional courses of all students of a certain field of study were selected. The number of students that were interested in each course was calculated and the sorted list of all courses based on the calculated value was created from the most interesting.

S2. **Courses enrolled by similar students.** We computed the student similarity with all active students and also students graduated in the last five years. The revealed courses were sorted into a list by the number of occurrences in similar students' sets of optional courses.

S3. **Courses taught by favorite teacher.** Courses were sorted into a list in decreasing order by the popularity of a teacher.

S4. **Courses enrolled by friends.** Courses were sorted into a list by the number of occurrences in friends' sets of optional courses.

5.2 Recommendation Methods Evaluation

As a contrary to the selective course recommendation, we supposed that students are not familiarized with all the optional courses. Therefore, the offline experiments were not sufficient evaluation technique in this case and we had to conduct a user study [10]. We contacted only selected group of students to request them to assess our recommendations.

We could approach 607 students enrolled in one of our courses in the last semester. Considering the number of students and expecting the lower response rate of students, we selected 5 top rated courses by each algorithm for each student. The coverage of approaches when the algorithm found at least one course to offer is presented in Table 3 in the first row. Only for a half of students, we revealed friends who could inspire students with interesting courses. The average number of offered courses by each algorithm can be seen in the second row. The approach which uses social ties (S4) offered only 4 courses on average.

In our experiment, we offered 10 courses at maximum selected using the 2 our algorithms S_i and S_j for each student. We sorted the students in the list by their average grade in order to be independent of students' characteristics and nearly randomly selected 2 algorithms that offered its top 5 courses each at maximum to students. We balanced the number of occurrence of each algorithm due to the low coverage of S4. We also merged the list of courses of S_i and S_j in order to not prioritize one of them in the following order: S_{i1}, S_{j1}, S_{i2}, S_{j2}, S_{i3}, S_{j3}, S_{i4}, S_{j4}, S_{i5}, and Sj_5. When both algorithms selected the same course, the course appeared only once in the list. The assessment of the course was added to results for both algorithms.

Table 3. Algorithms coverage

Algorithm	S1	S2	S3	S4
Coverage	1	1	0.96	0.49
Offered Courses	4.98	4.98	4.47	4.02

Subsequently, students were asked for assessing the recommendation during their course enrollment process to increase the possibility of their reaction. Students could mark courses using the following attributes: like, do not like or leave it unanswered. Overall, 172 students responded. The most of them responded in one week since the invitation (see Figure 4).

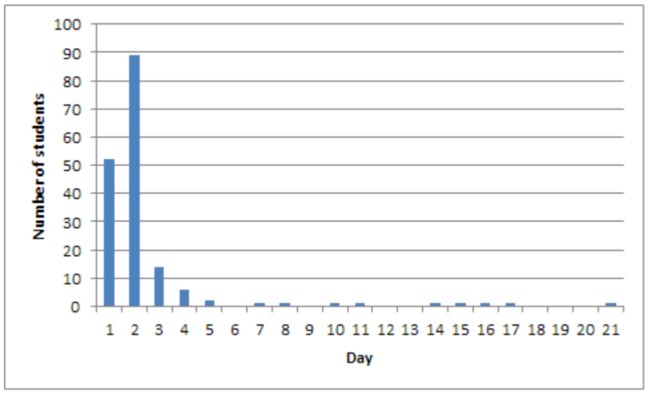


Figure 4. Students' reaction period

The distribution of students' reactions is shown in Figure 5. The best recommendation was offered by the algorithm S2. The algorithm is based on the similarity of students' sets of interesting courses.

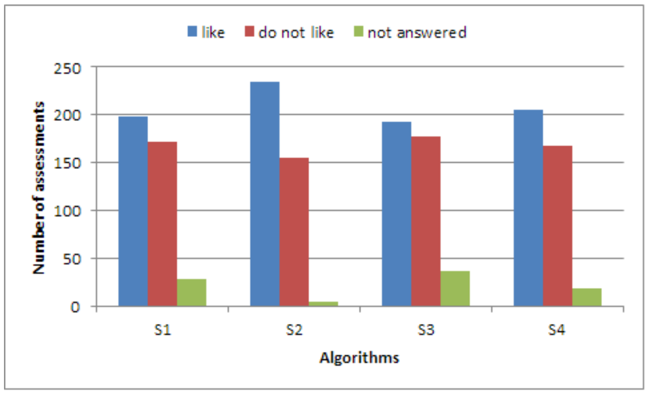


Figure 5. Assessed courses

The number of students assessed (NSA) our algorithms was almost in balance. Each student was included twice: for each of algorithms that assessed. As it can be seen in Table 4, we obtained more assessments of courses inspired by friends' selections (S4). It can mean that students with more social ties in the system are more active. We omitted recommendations that were not assessed. For all algorithms we obtained enough assessments to be able to properly evaluate them. We utilized the same evaluation metrics as for selective courses besides recall because we could not compute false negatives. On average for all algorithms, students liked 2-3 of 4-5 offered courses.

Table 4. Algorithms evaluation

Algorithm	S1	S2	S3	S4
NSA	79	79	82	99
Liked Courses	2.52	2.97	2.35	2.07
Offered Courses	5	5	4.8	3.9
Precision	0.53	0.60	0.52	0.55
Rank	**3**	**1**	**4**	**2**

Considering all evaluation methods, we determined the ranking of algorithms' success rate. Algorithm based on similarity of interesting courses (S2) reached the best results. However, the final solution will combine all algorithms to achieve best results.

6. RECOMMENDATIONS

We have designed new elements for Registration Application which might be available to all students of Masaryk University in the future. The first enhancement presents the predicted difficulty of courses to students. The predictions are computed by the method described in Section 3.3. The predicted grades correspond to the following color:

- Excellent grade – green color.
- Good grade – yellow color.
- Bad grade – red color.

All predictions are presented as the icons of corresponding color. When we have no predictions, there is no icon. We try to predict grades of courses that students enrolled or courses that we recommend to them (see Figure 6). Based on these warnings, students can concentrate on difficult courses or revise their choices depending on the planned workload in the semester.

The second improvement is the panel on the right (see Figure 6) where the recommended courses are presented. For each student we remind mandatory courses, recommend selective and optional courses selected by methods introduced in Sections 4 and 5, and also recommend their prerequisite courses. After clicking the wrench icon, the short explanation of each recommendation is displayed to increase students' trust to the system [5]. They can also assess each recommendation. Based on assessments we continuously improve our algorithms.

7. CONCLUSION

We presented a pilot version of course enrollment recommender system that reminds students their duties, warns them against difficult courses and recommends them potentially beneficial courses. Therefore, the system helps students with their decisions during the enrollment process at the beginning of each semester.

More specifically, we designed four algorithms suitable for the course recommendation. The first algorithm searches for the most frequently enrolled courses. The second algorithm utilizes similarities of students based on courses of their interests. The third algorithm recommends courses of students' favorite teachers. The last algorithm calculates the social ties among students and selected courses which were interested for students' friends.

The most suitable algorithm for the selective course recommendation was the first described algorithm. Students usually selected easier courses defined in their templates. In contrary, the best results for the optional courses recommendation achieved the second algorithm utilizing students' similarities. However, we decided to employ all methods in the system due to the high students' satisfaction with recommendations. Optional courses were also recommended only if we predicted that students could pass the course and they had free time slots in the timetable for the course. We validated all designed methods on data originated from students of the Faculty of Informatics Masaryk University stored in the university information system.

We also introduced the environment that presents recommendations to students, offers them the explanations why the courses were selected, allows them to leave a feedback, warns them against difficult courses, and reminds them important events that should be accomplished, e.g. enroll in mandatory courses or enroll enough credits. The designed course enrollment recommender system will be a part of the university information system in the future.

8. REFERENCES

[1] Bendakir, N. and Aimeur, E. 2006. Using Association Rules for Course Recommendation. In *Proceedings of the AAAI Workshop on Educational Data Mining*, pp. 31-40.

[2] Bydžovská, H. 2016. A Comparative Analysis of Techniques for Predicting Student Performance. *Proceedings of the 9th International Conference on Educational Data Mining 2016* (Accepted).

[3] Lee, J. Ch. Y. and Lee, K.-W. 2011. An intelligent course recommendation system. *Smart Computing Review*, 1(1).

[4] Loll, F. and Pinkwart, N. 2009. Using collaborative filtering algorithms as elearning tools. *In Proceedings of the 42nd Hawaii International Conference on System Sciences*.

[5] O'Donovan, J. and Smyth, B. 2005. Trust in Recommender Systems. *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pp. 167-174.

[6] O'Mahony, M. P., and Smyth, B. 2007. A recommender system for on-line course enrolment: an initial study. *In Proceedings of the ACM conference on Recommender systems (RecSys '07)*, pp. 133–136.

[7] Parameswaran, A., Venetis, P., and Garcia-Molina, H. 2011. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Trans. Inf. Syst.*, 29(4):20:1–20:33.

[8] Recker, M. M., Walker, A., and Wiley, D. 2004. Collaborative information filtering: A review and an educational application. *International Journal of Artificial Intelligence in Education*, Volume 14 Issue 1, pp. 3-28.

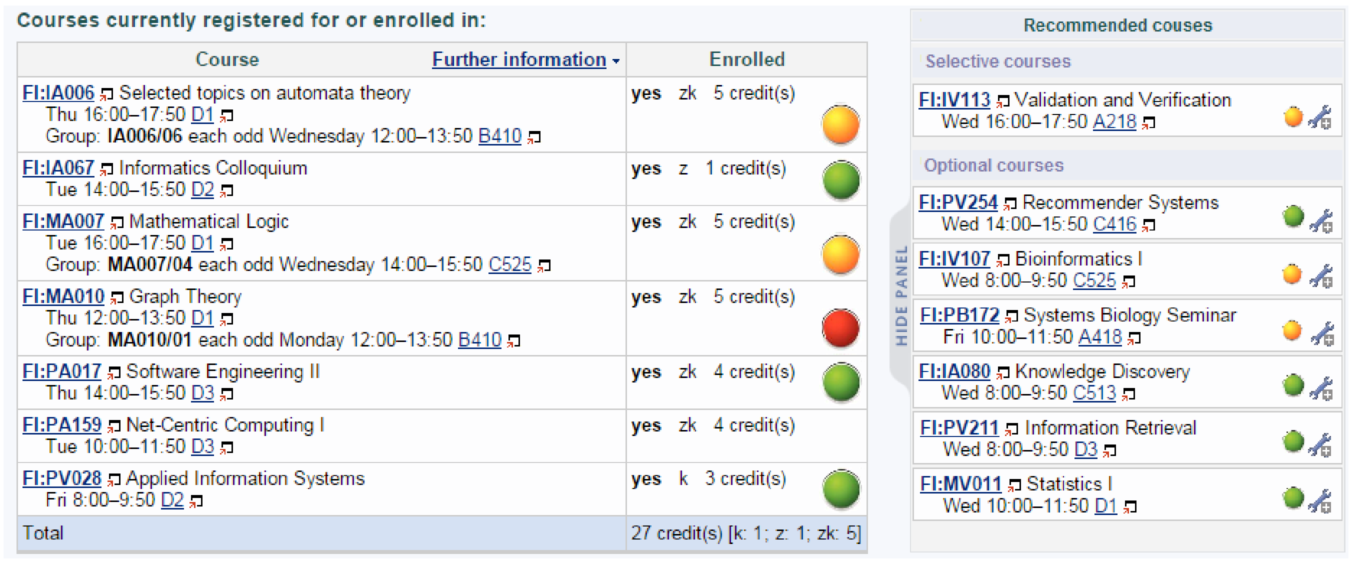
[9] Romero, C., Zafra, A., Luna, J. M., Ventura, S. 2013. Association rule mining using genetic programming to provide feedback to instructors from multiple-choice quiz data. *Expert Systems* 30(2): 162-172.

[10] Shani, G. & Gunawardana, A. 2011. Evaluating recommendation systems. Recommender Systems Handbook, pp. 257-297.

[11] Sobecki, J. and Tomczak, J. M. 2010. Student courses recommendation using ant colony optimization. In *Proceedings of the Second international conference on Intelligent information and database systems*: pp. 124-133.

[12] Vialardi, C., Chue, J., Peche, J. P., Alvarado, G., Vinatea, B. Estrella, J., and Ortigosa, A. 2011. A data mining approach to guide students through the enrollment process based on academic performance. *User Modeling and User-Adapted Interaction*, Volume 21, Issue 1, pp. 217-248.

[13] Vuorikari, R., Hummel, H., Manouselis, N., Drachsler, H., and Koper, R. 2011. Recommender Systems in technology enhanced learning. In Recommender systems Handbook, pp. 387-415. Spriger Verlag.



Courses currently registered for or enrolled in:

Course	Further information	Enrolled			
FI:IA006 Selected topics on automata theory Thu 16:00–17:50 D1 Group: **IA006/06** each odd Wednesday 12:00–13:50 B410		yes	zk	5 credit(s)	🟠
FI:IA067 Informatics Colloquium Tue 14:00–15:50 D2		yes	z	1 credit(s)	🟢
FI:MA007 Mathematical Logic Tue 16:00–17:50 D1 Group: **MA007/04** each odd Wednesday 14:00–15:50 C525		yes	zk	5 credit(s)	🟠
FI:MA010 Graph Theory Thu 12:00–13:50 D1 Group: **MA010/01** each odd Monday 12:00–13:50 B410		yes	zk	5 credit(s)	🔴
FI:PA017 Software Engineering II Thu 14:00–15:50 D3		yes	zk	4 credit(s)	🟢
FI:PA159 Net-Centric Computing I Tue 10:00–11:50 D3		yes	zk	4 credit(s)	
FI:PV028 Applied Information Systems Fri 8:00–9:50 D2		yes	k	3 credit(s)	🟢
Total		27 credit(s) [k: 1; z: 1; zk: 5]			

Recommended couses

Selective courses	
FI:IV113 Validation and Verification Wed 16:00–17:50 A218	🟠
Optional courses	
FI:PV254 Recommender Systems Wed 14:00–15:50 C416	🟢
FI:IV107 Bioinformatics I Wed 8:00–9:50 C525	🟠
FI:PB172 Systems Biology Seminar Fri 10:00–11:50 A418	🟠
FI:IA080 Knowledge Discovery Wed 8:00–9:50 C513	🟢
FI:PV211 Information Retrieval Wed 8:00–9:50 D3	🟢
FI:MV011 Statistics I Wed 10:00–11:50 D1	🟢

Figure 6. Demonstration of Interface