Masters' Courses Recommendation: Exploring Collaborative Filtering and Singular Value Decomposition with Student Profiling

Fábio Carballo
fabio.carballo@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

November 2014

Abstract

The decision that students have to make on which masters courses to enrol is decisive: this choice may have a direct impact on their academic and personal goals and may constrain their future professional area. A bad choice of courses may demotivate a student, which can cause the student to drop out or to not take advantage of the totality of his skills. However, usually no support is offered to contest this problem. It is crucial that understanding over students particularities is acknowledged, so as to recommend courses that besides interesting, are also adequate to their capabilities. The use of recommendation systems to suggest items to users has well-known success in several domains, as in e-commerce and movies recommendation. Some of the most successful recommendation techniques are singular value decomposition (SVD) and collaborative filtering. In this work, we propose the combination of these two techniques, along with as-soon-as-possible (ASAP) classifiers, so as to capture the hidden factors that explain courses marks. With this knowledge, we aim to predict students masters courses marks, in order to be able to recommend the masters courses that are more adequate for their skills. Through ASAP classifiers is possible to anticipate the recommendation, requiring less data from the target student. Our results show that our approach to predict the masters courses marks has potential to serve as a starting point for the recommendation production.

Keywords: Recommendation Systems, Singular Value Decomposition, Educational Data Mining, Course Recommendation, Classification

1. Introduction

After bachelor, many students strive to select the masters courses that are most likely to meet their interests and skills. Although this decision may have a strong impact on students motivation and future achievements, usually no support is offered to help them on this decision. A bad choice of courses may cause demotivation, which can result in a student to drop out or to not take advantage of the fullness of his skills. Hence, it is urgent that students are given the so much needed support to help them to select the best set of master courses, according their skills and interests, which will keep them motivated and with a clear path to achieve their goals. Educational Data Mining is concerned on developing methods that use several types of data originated from educational contexts aiming to improve the learning process. Although several interesting results have been achieved on student modeling and performance prediction, when it comes to recommend courses we are not able to find a lot of diversity on the existing solutions [13]. However, the good results that recommendation systems have on other areas are acclaimed [10] and, despite some few approaches, their (natural) application to this problem is roughly explored. The fact is that this problem is suitable to be seen as a recommendation problem: we can easily define the object of recommendation (courses), our users (students), and the most important, we should be able to get a lot of heterogeneous data to explore using recommendation techniques, so as to produce our recommendations.

Current solutions on education have a tendency to recommend courses based on their contents or potential interest to the students, not considering how those courses can affect the students overall academic performance [6][12][17]. Other solutions demand too much participation by the students [5][14][19]. Therefore, we propose the creation of a system that, given a student's bachelor results, automatically recommends the master

courses that are more adequate to the student's skills. To achieve it, we propose to combine two successful recommendation techniques: *singular value decomposition* (SVD) and *user-based collaborative filtering* (UB-CF), so as to explore historical students bachelor and master results towards the prediction of master courses marks. This follows our belief that we will be able to produce good master courses recommendations if our predictions unveil themselves accurate. We also will show how to use ASAP classifiers [2] so as to complete profiles of students whose bachelor data is not totally available. This allows to anticipate recommendation, as well as to recommend courses to students that for any reason have completed only a subset of the bachelor courses.

The rest of the paper is organized as follows: in section 2, we introduce some important concepts and techniques in the area of recommendation systems, approaching both *singular value decomposition* and *user-based collaborative filtering*. We also overview existing literature about courses recommendation and the systems that address it. We then present our proposal in section 3, explaining how we perform recommendations in the educational context using SVD and UB-CF, taking into account how will the courses recommendation suit students skills. In section 4, we propose the combination of *ASAP classifiers* to our recommendation process. The paper ends with the evaluation of our recommendations on section 5.

2. Background

Recommendation Systems have become an important multidisciplinary research field in the mid-1990's, and many people have exhaustively dedicated large amounts of time and effort to the problem of recommending items from some fixed databases. A recommendation system on a common formulation can be seen as a set of tools and techniques used with the goal of providing suggestions of items to individuals who lack sufficient competence to evaluate the potentially overwhelming number of alternative items available. They have grown to become fundamental applications in electronic commerce (Amazon [10]), information access, entertainment (Netflix [9]) and various types of services, providing suggestions that effectively prune large information spaces so that users are directed to those items that best meet their needs and preferences.

The majority of recommendation systems are based on Collaborative Filtering (CF). In pure CF, one identifies users whose tastes are similar to those of the target user and recommends items they have liked, never doing any analysis over the items at all. For this reason it is said that it uses the common principle of word of mouth [3]. By using other users recommendations, it is possible to deal with any kind of content and receive items with dissimilar content to those seen in the past. Since other users' feedback influences what is recommended, there is the potential to maintain effective performance given fewer ratings from any target user. However, this approach is not ridden of problems: a user profile for a new user will only be valid after he have done some ratings, in order to enable the identification of the most similar users to the new user. Other weakness is that if a new item appears in the database there is no way it can be recommended to a user until more information about it is obtained through another user either rating it or specifying which other items it is similar to. If the number of users is small, relative to the volume of information in the system, then there is a danger of the coverage of ratings becoming very sparse. Another problem is that it will create poor recommendations for a user whose tastes are not according to the majority of the population. Usually, these kinds of systems make use of a users-items matrix R - the ratings matrix - where R_{ij} stands for the rating that user i gave to item j.

Moreover, collaborative filtering techniques can be split in two categories: *user-based* and *item-based*. User-based systems employ techniques to find a set of neighbors that are the most similar users to the target user. As soon as the neighbors are defined, the system combines the preferences of neighbors to produce a prediction for the active user. These methods are popular due to their simplicity, interpretability and ability to produce accurate recommendations. But there are several disadvantages also: they depend on human ratings and their performance is poor when there is too much data sparsity, since it prevents the scalability of the solution. Item-based techniques first analyze the user- item matrix to identify relationships between different items, and then use the relations to indirectly compute recommendations for users. Despite the fact that these techniques handle the sparsity better than the user-based ones, they also require an expensive model construction [15][1]. After the Netflix challenge [7], there was a huge trend to use the so-called latent factor models, aiming to reveal the hidden latent features that somehow explain the observed ratings. One of the most applied techniques with these models is a matrix factorization technique - *singular value decomposition* - due to its accuracy and scalability. This technique factors the $m \times n$ ratings matrix R, into three matrices as in equation (1),

$$R = U\Sigma V' \tag{1}$$

where U and V are two orthogonal matrices of size $m \times m$ and $n \times n$ respectively. Matrix Σ is a

$r \times r$ diagonal matrix whose values σ_i are the so-called singular values of this decomposition. These values are stored in decreasing order of their magnitude. We can discover several applications of interest by using SVD in a particular way: if we truncate Σ so that we only retain the k largest singular values, so as to yield Σ_k, and then reduce both U and V accordingly, the obtained result is a k-rank approximation $R_k = U_k \Sigma_k V_k'$ of matrix R. This decomposition can be interpreted as an expression of the feature (also referred to as 'topic') preference-relevance model. This way, the $|U| \times k$ rows of matrix U can be seen as the users' interest in each of the k inferred hidden features (topics). Correspondingly, the rows $|I| \times k$ of matrix V hold the relevance of each of topics to each item. Regarding matrix Σ, its *singular values* are the weights for the preferences representing the impact that a certain feature has on the users' preferences described on the original ratings matrix. With this, to obtain a user's preference on a certain item, we just have to compute the weighted sum of the user's interest in each of the topics times that item's relevance to the topic. In fact, this corresponds to the weighted dot product of the user i features vector U_i and the item j features relevance vector V_j.

Nevertheless, SVD is not known for dealing well with sparse matrices, where there are a lot of missing values [**?**]. Hence, as in usual recommendation problems there exists a lot of sparsity on the ratings matrices, some care must be taken in how to overcome this problem. Initially, some proposals tried to fill in the unknown values with normalized averages, but this is highly prone to overfitting[**?**]. A solution to this sparsity problem was found during the Netflix challenge, whose goal was to predict the ratings users would give to movies. To predict each rating, Simon Funk [7] proposed to use a *stochastic gradient descent* algorithm in order to compute the best rank-k matrix approximation using only the known ratings of the ratings matrix R. This process follows the same idea as the one used on training neural networks. With the error in a prediction of user i to movie j being (R_{ij} - $R_{k_{ij}}$), Funk's approach takes the derivative of the square of the error with respect to U_{ik} and then with respect to V_{jk}. Since R is constant, and $R_k \approx U \times V'$, the updates for the U and V then become (2) and (3), respectively:

$$\Delta U_{u,f} = \lambda((R_{u,i} - P_{u,i})V_{i,f} - \gamma U_{u,f}) \qquad (2)$$

$$\Delta V_{i,f} = \lambda((R_{u,i} - P_{u,i})U_{u,f} - \gamma V_{i,f}) \qquad (3)$$

In summary, the final solution of this learning problem is the combination of feature weights on both U and V such that the error in the approximation R_k is minimized. This solution is determined iteratively, as the gradient of the error function is computed at each iteration step. You should note that all features vectors have to be initialized with some values. Funk's basic approach is to fill in the values with the global rating average with some random noise introduced.

2.1 Courses Recommendation

One of the most challenging problems faced by university students is to correctly choose which academic path to take, based on the available information. Thus, students need counseling on making adequate choices to complete their academic degrees with success. In the last years, course recommendation systems have been suggested as the tool that should be able to provide the guidance students need.

AACORN [14] is a case-based reasoning system that recommends courses to graduate students. The case-based reasoning component first retrieves the most similar student histories for the given inquiry. So, it requires a partial history of the courses followed by a student before it can provide useful recommendations. In order to determine the similarities between course histories, the system uses the edit distance metric. AACORN adapts a solution by building a list of courses found in the retrieved histories but not found in the target student data. Finally, it ranks the courses in the following way: each time a course appears in one of the retrieved students' history it counts as one vote, and these votes are weighted according to the distance of the retrieved student history to the target student (the less distant, the more the weight). Hence, the courses are ranked according to their total vote weight. In 2011, Unelsrød [17] developed a recommendation system for course selection in higher education. The system has a component that uses a specialized user-based CF: it weights each user, based on their chosen major degree and whether or not the two users compared are friends. His expectations over the importance of the users friends were dashed, but there were improvements on the accuracy of the recommendations when focusing on users with common major degree. The system also has a content-based component that uses the courses features that a student has shown interest, in order to find the courses that suit best his preferences. Vialardi et al. [18] presented a decision tree based recommendation system with the goal of creating awareness of the difficulty and amount of workload entailed by a chosen set of courses. They based their research on the generation of two domain variables: the student's potential that is calculated for each course and represented as the average of the grades a student has obtained in the pre- requisites of that course; and courses' difficulty

represented by the average of the grades obtained by students in the course. Lastly, during enrollment students choose a set of courses and the system forecasts if they will pass or fail each of the courses using the C4.5 algorithm.

To the best of our knowledge, there is no proposal to use SVD along with *User-Based CF* in courses recommendation, and so we present one in the next section.

3. Masters Courses Recommendation using SVD and UB-CF

Our proposal is to apply a user-based approach to select the most similar historical students (neighbors) to a target student (considering their bachelor achievements) and then use those neighbors' bachelor and masters results to construct a new dimensional space using SVD. Afterwards, we use this new dimensional space to estimate initial marks for all masters courses. In the end, we perform a weighted average with the information of these estimated marks, so as to produce our final marks prediction, and then be able to recommend the master courses. One can see the flow of this process in figure 1. We will now describe the methodology taken to build the system and produce recommendations.

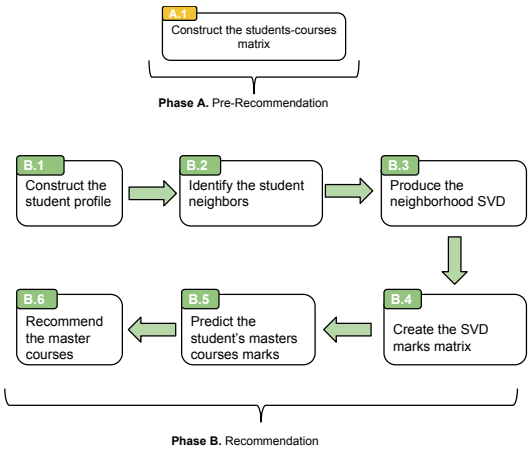

Figure 1: Main steps of our recommendation process

Phase A: Pre-Recommendation Before we can start to recommend courses to students, we have to parse students' results data into a structure that both *collaborative filtering* and *singular value decomposition* can explore. In our case, we will have to start from an historical record of triplets in the form of <Student, Course, Mark >.

As seen in section 2, the majority of the recommendation techniques makes use of a matrix R, that in general represents knowledge over the rating that a user gave to an item. Since we are trying to recommend courses that are adequate to the student's skills, we believe that representing a student as a vector of his marks is the starting point to capture his ability on different courses. Hence, our proposal is a novel mapping where we can look to the mark obtained by a student in a course as the usual rating in recommendation systems. More precisely, matrix R will have students represented on rows and courses on columns, and each entry R_{ij} will be filled with the grades obtained by the i^{th} student on the j^{th} course. In cases that there are missing values, that is, in the courses where the student did not enroll, we will represent the grade with the 0 value. This is a natural mapping, as we also want to recommend the items (courses) with predicted better ratings (marks). Additionally, we have the constraint of recommending only a subset of the items (the masters courses). From now on, we will illustrate some steps with a small example, so that we can better understand how the process unwinds. In figure 2, one can see that we have some historical students' results and how we structured this information in the students-courses matrix R. This matrix holds knowledge over 6 different students, 5 bachelor courses and 4 master courses. In this small example the marks scale for each course goes from 1 to 5.

	Bachelor courses					Masters courses			
	NA	AI	DS	SE	OOP	DP	DSS	MC	IM
Matt	2	3	4	4	4	5	5	0	0
John	2	4	3	3	3	0	2	3	0
Susan	4	3	3	3	3	0	4	4	5
Eric	2	3	4	3	3	3	5	4	0
Victoria	3	4	3	3	3	0	3	0	4
Michael	2	3	5	4	4	0	4	0	4

Caption:
NA - Numerical Analysis AI - Artificial Intelligence DS - Distributed Systems
SE - Software Engineering OOP - Object Oriented Programming DP - Distributed Platforms
DSS - Decision Support Systems MC - Mobile Computing IM - Information Management

Figure 2: Example of a marks matrix R

Phase B: Recommendation In this phase we already have our students-courses matrix built, and we are ready to receive a request of any student wanting to know which are the best master courses for him to attend. We will now describe which are the followed steps when a student asks for a recommendation:

1. Construct the student's profile. As a student requests a recommendation from our system, the first thing we must do is to encapsulate knowledge over the student on an appropriate representation. Since we already implicitly defined a student representation as a vector of the student's marks, as it was seen on the construction of the students-courses matrix, it seems natural to use the same representation to the student being recommended.

This way, we can use the same kind of student profile all along the recommendation process, and enable easy similarity computation between different students in the following steps. We will also keep the same rules, as above, on how to fill in the values of the student's vector. Therefore, each position will hold the mark achieved by the student on the corresponding course, or 0 if the student have not enrolled on that same course.

2. Identify the student's neighbors In this step the goal is to identify the set of k neighbors of the student being recommended, considering only bachelor marks. This idea follows our belief that the more two students have similar marks on bachelor, the more will their masters marks be similar. The goal of this identification, is that later on the process, we can use knowledge over these neighbors' masters marks to produce the recommendations to the target student. For finding the k nearest neighbors to the student being recommended, we have to measure the similarity between this student and all those historical students. To do this, we will consider the bachelor courses marks, i.e, the values of the first n positions of the vector of each student, with n being the number of bachelor courses. The distance metric used to calculate the similarity value between two students will be the *Pearson Correlation* (see equation 4). This metric is commonly used in recommendation problems, as it has already shown to achieve good results when used to define the user's neighboorhod. In our approach, this metric can be seen in (4) :

$$w_{a,u} = \frac{\sum_{i \in I}(g_{a,i} - \bar{g}_a)(g_{u,i} - \bar{g}_u)}{\sqrt{\sum_{i \in I}(g_{a,i} - \bar{g}_a)^2}\sqrt{\sum_{i \in I}(g_{u,i} - \bar{g}_u)^2}} \quad (4)$$

where I is the set of bachelor courses attended by both students, $g_{u,i}$ is the mark that student u obtained on bachelor course i, and $\bar{g}_u$ is the average bachelor mark obtained by student u. This metric only uses the courses that both students have attended. Its values will range between 1 and -1, and higher values represent higher similarity between two students. Although *Pearson correlation* would suffer from computing high similarity between students with few attended courses in common, in our specific case that will not happen. This is due to the fact that all students being compared will have enrolled on the exact same set of bachelor courses.

In our example, with the number of neighbors to select set to 3, we would have to first compute the similarity of each historical student on matrix R to a hypothetical target student, considering their bachelor marks, and then select the three most similar neighbors. In fact, one can look up on a possible result on figure 3 and verify that the most similar neighbors are Michael, Matt and Eric, respectively.

	Bachelor courses					
	NA	AI	DS	SE	OOP	Similarity
Matt	2	3	4	4	4	0,534
John	2	4	3	3	3	-0,422
Susan	4	3	3	3	3	-0,133
Eric	2	3	4	2	3	0,422
Victoria	3	4	3	3	3	-0,08
Michael	2	3	5	4	4	0,628

Figure 3: Neighbors identification after similarity calculation

3. Create the neighborhood students and courses features space It is in this step where SVD is used on top of the user-based approach. To do it, we start by applying Funk's SVD using the bachelor and master marks for all the found neighbors. We go on to construct the U and V Funk' matrices that correspond the students' features space and the courses features space. The first matrix U is a $k \times f$, where k is the number of neighbors and f is the number of hidden features that we want to consider. The second one, matrix V, holds the courses features weights and is a $(n + m) \times f$, where n is the number of bachelor courses, m is the number of master courses and f is once again the number of features that we want to consider. Note that f has always the same value for the two matrices. These two matrices are then initialized with the global mark average on matrix R and then the *gradient descent* process learns the values (marks) within a predefined number of iterations, trying to minimize the error between the predicted and original marks.

In this step, when producing these dimensional spaces in our example, we set the number of hidden features to two (we choose this value to ease the task of visualizing the data). In figure 4, one can see the resulting students' features space and courses features space. These resulting matrices hold the relation that both students and courses have with each of the hidden discovered features, and the product between the two constitutes a 2-rank approximation of the original matrix R.

4. Create the SVD marks matrix In this step, we will use the recently created students' and courses features spaces to create a new matrix M that holds a mark for each neighbor student-course pair. This new matrix M corresponds to the approximation of the original matrix created by the appliance of the gradient-descent SVD on the previous step. In this matrix, each of its entries will hold a relation between a neighbor student and a course, more precisely on how they are related to each one of the discovered hidden features. In fact, this relation is materialized into a mark value, that corresponds to the dot product between a student's and a course's features weights. This way, to pro-

Students' Features Space (U)		
Course	Feature 1	Feature 2
Matt	1,465	1,444
Eric	1,286	1,272
Michael	1,416	1,398

Courses Features Space (V)		
Course	Feature 1	Feature 2
NA	0,801	0,807
AI	1,068	1,065
DS	1,427	1,411
SE	1,252	1,242
OOP	1,252	1,243
DP	1,317	1,309
DSS	1,513	1,496
MC	1,262	1,259
IM	1,243	1,2377

Figure 4: Resultant matrices of the application of SVD in the neighbors' space

duce the computation of a M_{ij} value on this matrix it is used (5):

$$M_{i,j} = U_i \cdot V_j \tag{5}$$

This approximation has the advantage of having no sparsity, which is good to produce predictions with higher quality. Since it also uses the hidden features that have more weight to the explanation of the marks distribution, this approximation will also reduce unwanted noise and capture the most weighting factors. This way, we will have a complete matrix with all the estimated marks for the selected neighbors. We can see a specific scenario of the construction of this SVD marks matrix following our example. For instance, if we want to know which is Matt's mark on the DP course we have to compute the dot-product between S_{Matt} and V_{DP} (see figure 4). Hence, the mark value would be $1.465 \times 1.317 + 1.444 \times 1.309 \approx 3,82$. One can see the totality of the SVD marks matrix computed for our example in figure 5.

	Masters courses			
	DP	DSS	MC	IM
Matt	3,82	4,379	3,669	3,602
Eric	3,357	3,849	3,225	3,169
Michael	3,696	4,237	3,55	3,489

Figure 5: Matrix with marks obtained by SVD prediction

5. Predict the student's master courses marks Now, we have a new matrix M, with less noise, no sparsity and that encloses the relations between the marks obtained. This seems a good scenario, as this matrix holds a lot of knowledge over marks that we can explore to predict the master marks for our student. Our suggestion is to predict a student u mark on course i using a weighted average of the marks on course i present on matrix M, using the *Pearson Correlation* as the weight:

$$p_{u,i} = \bar{M}_u + \frac{\sum_{u' \in K} PearsonCorrelation(u,u')(M_{u',i} - \bar{M}_{u'})}{\sum_{u' \in K} |PearsonCorrelation(u,u')|} \tag{6}$$

Subtracting the students' average mark $M_{u'}$ compensates for differences in students' marks obtained, as some students will tend to have higher marks than others. In sum, in this step, we apply the expression above for each one of the existing master courses.

6. Recommend master courses Finally, and since we have predicted all the master courses marks for the student being recommended, we can approach on which will be the chosen set of courses that we will recommend. On this step, we argue that if our previous steps are able to produce accurate predictions, than the recommendation can be really simple. For instance, if N is the number of master courses to recommend, we can simply opt for a *Top N* approach, and recommend the N master courses with the best predicted marks.

4. Completing Students Profiles Prior to Recommendations

Now that we presented our proposal in how to recommend courses with value to students we want to step onto another problem: recommending courses when the data over the student being recommended is not fully observable. We will approach this problem using ASAP classifiers [2].

These classifiers take a different look to what is usual in classification. In particular, if I is a set of instances, A a set of attributes and C a set of possible classes, an instance x_i from I is described by an ordered list of M attributes from A, and is represented as $x_i = x_{i1}x_{i2}...x_{im}c_i$, where c_i is the instance class. Then, given I and number of attributes n, such that $n < m$, classifying an instance x_i *as soon as possible* consists on finding the value of c_i, only considering the first n attributes, the *observable* ones.

Hence, our goal is to use these classifiers to incrementally estimate each missing bachelor mark. This is, if s is a student instance with m attributes that correspond to the bachelor marks, where only the first n are known, then for each bachelor course i, where $i > n$, we will estimate the s_i using s_1, s_2, ... s_n. This process goes on incrementally, with

each mark estimation s_i being used on the prediction of the mark s_{i+1}. This process only ends when all s attributes are estimated. As this technique assumes that there is an order between the attributes, it is required that we define an order on the bachelor courses. Our idea is to order courses by year, and for each year order them by semester, as temporal order is the natural order seen in the academic process. However, we recognize that inside each semesters we can explore several orders that may hold different results, as the marks on course X may be correlated to the marks on course Y, but the inverse situation does not hold. Hence, the ideal scenario would be one where the order between the courses did not matter in the process. One should also note that this approach is prone to error propagation due the progressive bachelor marks estimation. This is related to the fact that, for instance, if we want to estimate the bachelor marks x_i and x_{i+1}, we will first estimate x_i with the $x...x_{i-1}$ observable marks, and then we will estimate x_{i+1} with the $x...x_{i-1}$ observable marks and with the estimated x_i mark, that may hold an associated error (as usually the predictions are not perfect).

5. Results

To be able to evaluate and corroborate our proposal, we have collected and analysed real students' results to use as a case study. The used dataset was dran from a set of students enrolled in the academic program of Information Systems and Computer Engineering at Instituto Superior Técnico, Universidade de Lisboa, in Portugal. This program is composed by a three years bachelor and a two years masters. In this dataset we are able to reach students' results on 22 mandatory courses, the bachelor courses, and around 60 optional masters courses. In this dataset, the grade scale for all courses goes from 0 to 20, where 10 is the minimum mark that a student must obtain to be approved on any course. Our dataset contains around 25000 results collected from 550 students. In our view, our recommendation problem can be decomposed into two other problems: the prediction of the marks and the production of recommendations. We will give more focus on evaluatiing masters marks prediction accuracy. The reason is that, we believe that, if we achieve a low error on the predictions then we are able to recommend courses with more certainty. Though, we will also explore other aspects to evaluate our approach.

5.1 Masters Marks Prediction Analysis To analyse how good is our approach in predicting students' masters marks, we have to use one of the statistical accuracy metrics. These metrics aim to compare the prediction value against the actual real-values for the customer-product pairs, i.e. student-course pairs in our specific problem. Some of the most used metrics are the *Mean Absolute Error* (MAE) and *Root Mean Squared Error* (RMSE) between the predicted and the real values. As research experience shows that both metrics typically track each other, we opted to use only one, in this case the MAE, because it is the most commonly used and easiest to interpret.

To begin our experiment we chose two baselines that are straight-forward and constitute good points from where we can establish comparisons to our prediction results:

- *Course Average*: This first baseline sets the predicted grade of a student i on masters course j as the average of the marks obtained by the training students on course j.
- *Student's Bachelor Average*: This second baseline takes a similar approach but sets the predicted grade for a student i on course j as his grade point average on bachelor.

To start our experiment, we began by building the students-courses matrix with the training data (70% of all students). We set the number of features to 20 and the number of training steps of the gradient descent to 25 (these were some values that showed to achieve good results). We then ran the recommendation process and predicted the master's courses marks for the test set using different number of neighbors. We would like to add that in our approach, and for each chosen number of neighbors, we ran the experiment five times and made an average of the achieved MAE, due to the random noise present in the training of the *gradient descent* algorithm. Then, we applied both the *user-based* approach and the two mentioned baselines, to predict the test students marks on every masters course.

From analysis of the results presented in figure 6, we can see that the *students bachelors average* approach is the one that presents the worst results, achieving a MAE around 2.2. This may be connected with the fact that the marks obtained on the masters tend to be higher than the ones obtained in the bachelor. Then, we can see that around the 1.9 of MAE, we have the *course average* baseline approach.

Regarding our approach, we can establish a connection between the error achieved and the number of neighbors chosen. First, we can see that the error is larger when the number of neighbors used is less than 30. Hence, it is noticeable that our approach suffers when using small *neighborhoods*. This may be related with the unability of both the *user-based CF* and *gradient descent SVD* to gather knowledge

using a small amount of data. Our belief is that the algorithm is adjusting too much to the created model, since it has just a few historical students to analyze, which results in the overfitting of the model. Though, as we increase the number of used neighbors, we can see that the error decreases, stabilizing between 70 and 150 neighbors, where we reach our optimal error. Finally, the error starts to slowly increase as we reach the 300 neighbors (where it is using almost all the training set as the neighborhood). Nevertheless, our approach presents good results agains any of the baselines. In the optimal number of neighbors (150), our approach presents a decrease of 32% on the error regarding the *student's bachelor average*, and 17% when comparing to the *courses average* approach. At this time, we will put our approach to a bigger test by comparing it one of the most widely used recommendation technique: *user-based CF*. By looking into figure 6, we can be perceive that the *user-based* approach is worse than our approach: if we compare the optimal number of neighbors from both approaches, i.e the number of neighbors that hold the best results, we can see that ours presents a better accuracy. In particular, our best accuracy prediction has an error 4% smaller than the one achieved in the user-based CF. Hence, we show that the recognized power of singular value decomposition still holds in this domain, when applied on top of an user-based approach. It is interesting that it can find a way to discover the hidden factors that somehow explain the masters marks distribution. We may conclude this, since that with these comparisons we show that the predictions of the marks is more accurate in our SVD marks matrix, than in a regular marks matrix of a user-based approach. This means that we were able to gain a small advantage by creating our predictions from an SVD-dimensional space.

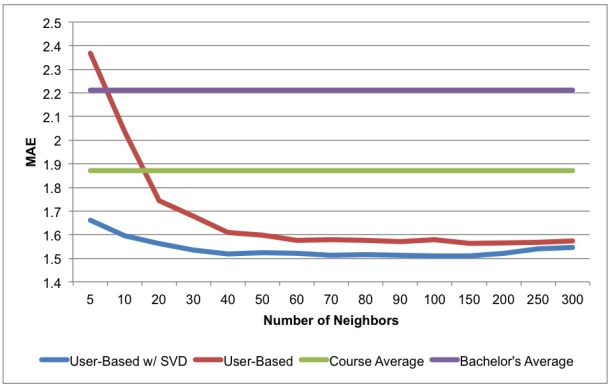

Figure 6: MAE on masters marks for our approach, the defined baselines and the user-based approach

We also compared our approach with a pure *gradient descent SVD*. This comparison was done in terms of the number of features, since it is the most decisive parameter that is common to both approaches. We should mention that we used the configuration of parameters that gave the optimal results on our approach. The results of this experiment, shown in figure 7, indicate that our approach is the better choice. The fact that our approach achieves an 6% lower MAE than the pure *gradient-descent SVD* approach points out that the user-based component on our approach has a great influence on the results.

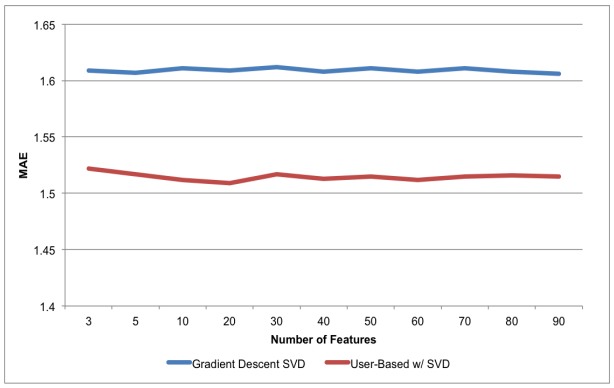

Figure 7: MAE comparison between our approach and a pure gradient descent SVD

5.2 Recommendation Analysis In this experiment we verify how our recommendation might affect students' masters average. As we are doing an offline evaluation of our system, using historical data, we will do this by calculating the average grade of the courses recommended that the students 'followed'. Hence, we will only take into account the courses grades where the course was one of the recommended by us for the testing student. To have a comparison to our approach we used two baselines:

- *Best Marks*: Recommends the N masters courses where students have higher marks.
- *Most Popular*: Recommends the N most frequented courses in the training data.

From observation of table 1, we can see that the average mark that students achieve with the recommendations of our approach is high, only being surpassed by our superior baseline *Best Marks*. These results support our idea on the importance of a small grade prediction error in order to produce good quality recommendations.

Best Marks	Most Popular	UB w/ SVD
16,454	14,59	16,245

Table 1: Average mark on followed recommendations

5.3 Student Modeling Analysis In this section, we will evaluate our approach on how to solve the problem of estimating the missing values on the students' bachelor information, so as to be able to produce the best recommendations. Once again, we will use MAE as our metric of evaluation. As we aim to use *ASAP classifiers* to estimate missing values on students bachelor profiles prior to recommendation, our worst-case scenario is the one where we have an incomplete student profile input to our recommendation process. With this said, it is natural to consider this scenario as one of our baselines. This way, we set up three different scenarios to have some initial insights. One is what we consider the regular scenario, where we have the complete target student data (from all three bachelor years). The other two are the ones where we are missing the data since the student's second and third bachelor years, respectively. We defined other inferior baseline that fills missing bachelor's marks with the average grade obtained by students on the bachelor's course where the value is missing. Our "superior" baseline, which constitutes our goal, is our proposed approach using the complete students' profiles. The results in terms of prediction accuracy for the baselines and our regular approach can be seen in figures 8 and 9. How it was expected, it is possible to see that the master's marks prediction accuracy increases with the amount of data used in the students' profiles. We can also see that filling the missing values with the respective course average does not bring any particular benefit in student profiling as the accuracy is inline with our inferior baseline. With this, we can conclude that initializing the unknown marks with some naïve knowledge is no good to contest the lack on information. In fact, it may end in constituting a source of noise, which results in not improving the accuracy.

Other conclusion reached is that ASAP estimation is slightly more accurate than any baseline. However, the results cannot achieve our superior baseline that is the case where we have full data over students bachelor marks. In the two presented scenarios, the ASAP estimation is the one that presents best results, but it still holds a considerable distance to our perfect scenario. This may be due to the fact that these classifiers also have an error associated to their estimations, carrying the error of its estimations to our prediction process.

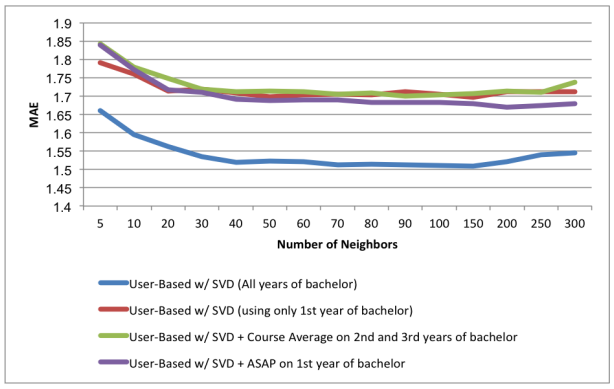

Figure 8: MAE variation of several approaches starting from knowledge on the first year of students bachelor

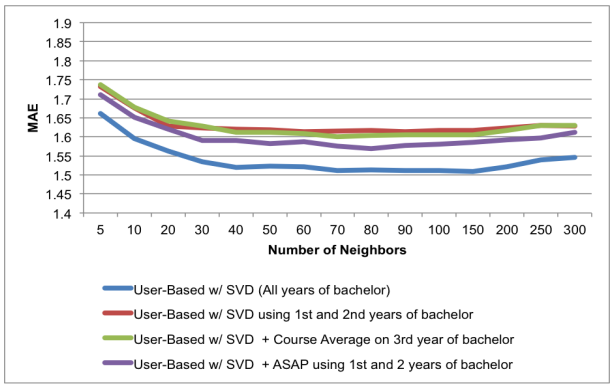

Figure 9: MAE variation of several approaches starting from knowledge on the first and second years of students bachelor

6. Conclusions

The urge to solve the problem of which masters courses are most suitable to each student led us to explore a set of recommendation techniques in order to predict all the possible masters courses marks. In particular, what we proposed was to use *singular value decomposition* on top of a *user-based collaborative filtering* approach. Hence, we have combined successful recommendations techniques with a novel mapping that looks to couses grades as the rating of the item. We can state that our approach gives a fresh look to course recommendation, since the historical approaches tried to recommendations based on the potential student interest on each course. We believe that our results were interesting, achieving a low MAE on the presented case study, and that open paths for future research. We also have shown an approach to try to estimate student data prior to the recommendation process, when the knowledge over the student is incomplete.

Despite our contribution in mapping a successful combination of recommendation techniques to the educational paradigm, we feel that we have only gave the first step into a topic that can be deeply explored. The marks that students achieve on their masters are not only related with their past academic achievements. A student's performance is tied with motivation, interests, skills, working status, evaluation methods, their school teachers and staff, as well as any other kind of resources that

help them during their masters. Our suggestion is that future works try to relate these refered factors to the achieved marks by the students.

Acknowledgements
This work is supported by Fundação para a Ciência e Tecnologia under research project *educare* (PTDC/EIA-EIA/110058/2009).

References

[1] G. Adamavicius and A. Tuzhilin. Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data ...*, pages 1–43, 2005.

[2] C. Antunes. *Anticipating student's failure as soon as possible*. CRC Press, september 2010.

[3] M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40:66–72, 1997.

[4] N. Bendakir and E. Aimeur. Using association rules for course recommendation. *In Proceedings of the AAAI Workshop on Educational Data Mining*, pages 31–40, July 16–17 2006.

[5] B. Bercovitz, F. Kaliszan, G. Koutrika, H. Liou, Z. M. Zadeh, and H. Garcia-Molina. Courserank: a social system for course planning. In *SIGMOD '09: Proceedings of the 35th SIGMOD International Conference on Management of Data*, pages 1107–1110, New York, NY, USA, 2009. ACM.

[6] K.-K. Chu, M. Chang, and Y.-T. Hsia. Designing a course recommendation system on web based on the students' course selection records. In D. Lassner and C. McNaught, editors, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003*, pages 14–21, Honolulu, Hawaii, USA, 2003. AACE.

[7] S. Funk. Netflix update: Try this at home, December 2006.

[8] D. Jannach, L. Lerche, F. Gedikli, and G. Bonnin. What recommenders recommend - an analysis of accuracy, popularity, and sales diversity effects. In *Proc. 21st International Conference on User Modeling, Adaptation and Personalization (UMAP 2013)*, Rome, Italy, 2013.

[9] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[10] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan. 2003.

[11] M. P. O'Mahony and B. Smyth. A recommender system for on-line course enrolment: an initial study. In *Proceedings of the 2007 ACM conference on Recommender systems*, page 133, 2007.

[12] A. Ranka, F. Anwar, and H. S. Chae. Pundit: Intelligent recommender of courses. In R. S. J. de Baker, A. Merceron, and P. I. P. Jr., editors, *EDM*, pages 339–340. www.educationaldatamining.org, 2010.

[13] C. Romero and S. Ventura. Educational data mining: a review of the state of the art. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(6):601–618, 2010.

[14] J. Sandvig and R. Burke. Aacorn: A CBR recommender for academic advising. *Technical Report TR05-015, DePaul University*, 2005.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[16] A. Surpatean, E. N. Smirnov, and N. Manie. Similarity functions for collaborative master recommendations. In K. Yacef, O. R. Zaïane, A. Hershkovitz, M. Yudelson, and J. C. Stamper, editors, *EDM*, pages 230–231. www.educationaldatamining.org, 2012.

[17] H. Unelsrød. Design and Evaluation of a Recommender System for Course Selection. Master's thesis, Norwegian University of Science and Technology, 2011.

[18] C. Vialardi, J. Chue, A. Barrientos, D. Victoria, J. Estrella, A. Ortigosa, and J. Peche. A case study: Data mining applied to student enrollment. In *Proceedings of Third Educational Data Mining Conference, Pennsylvania, USA*, pages 333–335, 2010.

[19] Y.-h. Wang, M.-H. Tseng, and H.-C. Liao. Data mining for adaptive learning sequence in english language instruction. *Expert Syst. Appl.*, 36(4):7681–7686, May 2009.