TÉCNICO LISBOA

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

Fábio Oliveira Garrido Carballo

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Professor Cláudia Martins Antunes

Examination Committee

Chairperson:	Professor José Manuel da Costa Alves Marques
Supervisor:	Professor Cláudia Martins Antunes
Member of the Committee:	Professor Bruno Emanuel da Graça Martins

November 2014

To my parents, sister, girlfriend and friends.
Without them every challenge would be way harder to overcome.

Acknowledgments

I would like to delightedly thank my advisor Cláudia Antunes for her guidance, patience and ability to keep a sense of humour and calm when I had lost mine. Her counselling and rooted experience on the topic were a big help in completing the long term goals set in the beginning of this dissertation project. In fact, I am not sure that many graduate students are given this constant support on not losing track of the goals to be achieved. I can only say that this experience was very pleasant and, as while I had my downs, struggling with failed approaches, and times of manic desperation, I also had the opportunity to work on a topic that is of big interest to me, as well as to travel and connect with many impressive persons around the world. For this reason, Professor Cláudia, I thank you.

I would also like to thank to my dearest colleagues António Barreto, Cláudia Henriques, Daniel Serrano and Daniel Cardoso, who have been with me since the beginning of this journey. They gave me great advise, support and endless moments of fun.

To my parents, Anabela and José, I have to thank for all the understanding, and for pushing me to the right path when the struggle was big. I also would like to thank Inês, my sister, for knowing on how to leave me optimistic against challenges. Most importantly, I would like to thank to my girlfriend Joana. Her support, encouragement and love were undeniabily the crucial pieces to keep my inner strenght.

In conclusion, I recognize that this research would not have been possible without the financial assistance of Fundação para a Ciência e Tecnologia under the reserch project educare (PTDC/EIA-EIA/110058/2009)

Lisboa, July 2014

Fábio Carballo

Resumo

A decisão que os estudantes têm que tomar sobre quais as disciplinas de mestrado a frequentar é decisiva: esta escolha pode ter efeito sobre os seus objetivos académicos e pessoais, assim como restringir o seu futuro profissional. Uma má escolha pode levar à desmotivação do aluno, e consequentemente a que este abandone os estudos ou que não explore as suas capacidades. Contudo, usualmente não é oferecido qualquer tipo de apoio para enfrentar este problema. Sendo assim, é crucial compreender as particularidades dos estudantes de maneira a recomendar disciplinas que para além de interessantes, sejam adequadas às suas características.

O uso de sistemas de recomendação para sugerir objectos a utilizadores têm casos de sucesso reconhecidos, em áreas como o comércio electrónico e recomendação de filmes. Algumas das técnicas de recomendação com mais sucesso são a decomposição de valores singulares (SVD) e os filtros colaborativos (CF). Nesta dissertação, propomos a combinação destas duas técnicas, a par com classificadores ASAP, por forma a capturar os factores escondidos que explicam as notas obtidas nas várias disciplinas. Com esta informação pretendemos prever as notas de mestrado dos alunos, de maneira a poder recomendar as disciplinas de mestrado que são mais adequadas às capacidades de cada estudante. A utilização dos classificadores ASAP tornam possível antecipar a recomendação, fazendo uso de um número menor de dados do aluno alvo da recomendação. Os nossos resultados mostram que a abordagem proposta para prever as notas das disciplinas de mestrado tem potencial para servir de ponto de partida para produção das recomendações.

Palavras-chave: Sistemas de Recomendação, Singular Value Decomposition, Data Mining Educacional, Recomendação de Disciplinas, Classificação

Abstract

The decision that students have to make on which master's courses to enrol is really decisive: this choice can have a direct effect 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 capabilities. However, usually no support is offered to contest this problem. It is crucial that comprehension 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' master's courses marks, in order to be able to recommend the master's courses that are more adequate to their capabilities. 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

Contents

List of Tables

List of Figures

Glossary

ASAP Classifiers	As-Soon-As-Possible Classifiers
CF	Collaborative Filtering
GPA	Grade Point Average
IB	Item-Based
MAE	Mean Absolute Error
NMAE	Normalized Mean Absolute Error
SVD	Singular Value Decomposition
UB	User-Based

Chapter 1

Introduction

After bachelor, many students strive to select the master's courses that are most likely to meet their interests and capabilities. 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 capabilities. 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 capabilities and interests, which will keep them motivated and with a clear path to achieve their goals.

Educational Data Mining describes a research field that is concerned on developing methods that use several types of data originated from an educational context, so as to improve the learning process. Though some interesting results have been achieved on student modelling and performance prediction, when it comes to recommend courses we are not able to find a lot of diversity on the existing solutions [Romero and Ventura, 2010]. However, the good results that recommendation systems have on other areas are acclaimed [Linden et al., 2003] and, despite some few approaches, their (natural) application to this problem is lightly 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. Still, recommending courses will always be a tough task, as a course recommendation should constitute not only a point of interest to the student, but it should also be adequate to his skills and respecting any kind of constraints (for example, courses prerequisites).

To have a clear idea of how current works are distributed, we have defined an evaluation framework that categorizes each work on recommendation in education according four different dimensions. These dimensions are the *item of recommendation*, the *data mining technique* used, the *recommendation technique* used and the *users* for whom the recommendation is directed. With this evaluation framework, we can identify where more work is demanded and which are the least explored alternatives. Current solutions have a tendency to recommend courses due to its contents, not considering how that course can affect the students overall academic performance [Wang and Yuan, 2009][Ranka et al., 2010][Chu

et al., 2003]. Besides that, some solutions propose the use of overly subjective criteria when producing a recommendation [Farzan and Brusilovsky, 2006][Unelsrød, 2011], while others demand too much participation by the students [Wang et al., 2009][Sandvig and Burke, 2005][Bercovitz et al., 2009]. Therefore, we propose the creation of a system that, given a student's bachelor's results, automatically recommends the master's courses that are more adequate to the student's skills. To achieve it, we propose to combine two successful recommendation techniques: *singular value decomposition* and *user-based collaborative filtering*, so as to explore historical students' bachelor's and master's results towards the prediction of master's courses marks. We believe that we will be able to produce good master's courses recommendations if our predictions unveil themselves accurate. We also will show how to use ASAP classifiers [Antunes, 2010] so as to complete profiles of students whose bachelor information is not totally available. This allows to anticipate recommendation as well as to recommend courses to students' that for any reason has completed only a subset of the bachelor courses.

This dissertation is organized as follows: in chapter 2, we overview the existing literature and the main concepts linked to the *recommendation problem* and systems that address it. Then we introduce the application of data mining to the educational context in chapter 3. In this chapter, we also define an evaluation framework from where we can categorize recommendation systems in educational data mining. We then present the state of the art on recommendations in an education environment and on the existing methods to model students' skill and performance. We end this chapter with a critic look on the existing works approaching recommendations in education. In chapter 4, we give an insight to SVD and present our solution on how to explore it on top of an *user-based* approach to recommend courses to students. We also show how to use *ASAP classifiers* to model missing student data, as a preprocessing step to recommendation. Finally, chapter 5 presents a case study that contains academic students' data, where we will test our system, as well as an explanation on how we intend to evaluate our proposal and the corresponding results. These results show that our approach has good prediction accuracy when compared to baseline predictors - particularly, student's average and courses average - and to other *collaborative filtering* techniques.

Chapter 2

Recommendation Systems

Recommendation Systems have become an important multidisciplinary research field since 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[Linden et al., 2003], GroupLens[Resnick et al., 1994]), information access, entertainment 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.

2.1 Basic Concepts

2.1.1 The recommendation problem.

The *recommendation problem* [Adamavicius and Tuzhilin, 2005] can be defined as follows : Let *C* be the set of all users and let *S* be the set of all possible items that can be recommended. Let *u* be a utility function that measures usefulness of item *s* to user *c*, i.e., u : *C* $\times S \rightarrow$ *R*, where *R* is a totally ordered set. Then for each user $c \in C$, we want to choose such item $s' \in S$ that maximizes the user's utility. The utility of an item is usually represented by a *rating*.

The central problem of recommendation systems lies in that utility *u* is usually not defined on the whole $C \times S$ space, but only on some subset of it. This means *u* needs to be extrapolated from known to unknown ratings, either using heuristics or estimating functions.

When designing recommendation systems, some properties must be taken into consideration so as to evaluate which is the best solution to the problem in hands. One conspicuous property in most recommendation systems is that they must be able to retrieve feedback from the users actions, either *explicitly* or *implicitly*. Examples of explicit data include rating of items, ranking of items and asking the user questions. Examples of implicit data includes analyzing social relationships, items viewings and

particular user behaviours. This feedback retrieval is essential to construct an accurate user profile. However, if the system is too intrusive the users may not like it.

Another aspect to consider is for which major purpose the recommendations will act. We can categorize it into several user-tasks, that differ on the exact user perception of recommendations goal. The most popular are to *Find Good Items* , *Find All Good Items* and *Recommend Sequence*. While the two first tasks are based on pure recommendation, suggesting specific items to their users, they differ in one single aspect. In the first task we are happy if we are able to suggest any good item, as in the case of Amazon.com, where it is impractical to recommend all interesting items to a user. At the same time, any buying of a recommended item is profit, even if any possible interesting item is overlooked. In the second task we don't want to overlook any possible good item, such as in the legal industry, where leaving no precedent behind is important. The *Recommend Sequence* task is related to the recommendation of a set of items with an implicit "consuming" ordering. This can be seen on the recommendation of playlists of songs. Other users are not so focused on receiving good item proposals, but focused on divergent tasks such as improving their profile, influencing recommendations or helping other users in the system [Herlocker et al., 2004].

Even the way how recommendations are presented have a great effect on the quality of the system, since different ways of presenting recommendations affect the perception of the users over the recommended items differently. The most simple and used visualization technique is to present the top recommended item. However, in large domains it is more suitable to present a list of the top N-items. There are also some variations to this basic *top item(s)* approaches, offering similar items (single item or list) or taking into account any type of context. Some systems also benefit from presenting all predicted item ratings. Other systems bet on structured overview, achieving trade-offs between items. One example is the treemap structure, using different colours to represent different topics, and different sizes to represent different degrees of recommendations.

2.1.2 Social Implications

As other many paradigms, recommendation systems are not impune to social problems. First, as soon as one has established his own profile of interests it is easy to just consume recommendations while not giving any rating input to the system. The second problem lies on that if anyone can provide recommendations, then these systems are susceptible to "attacks" from content owners that are able to generate loads of positive recommendations for their own materials and negative recommendations for their competitors [Resnick and Varian, 1997].

Intuitively, recommendation systems will be more effective the more information about its users they have [Adamavicius and Tuzhilin, 2005]. However, this fact raises some deep personal privacy issues considering that people may not want their habits or opinions to be known. Some recommendation systems permit anonymous participation or participation under a pseudonym, but this is not a complete solution since some people may desire an intermediate level of privacy [Resnick and Varian, 1997].

2.1.3 Explaning recommendations

An interesting view over recommendation systems is on how to estimate if a system is good at explaining its own suggestions. Since a system is as good as the users think it is, the system must make good use of explanations to its users. The effectiveness of those explanations can be measured using two fundamentally different approaches: *promotion* and *satisfaction*. According the former, the best explanations are the ones that are most successful at convincing the user to adopt an item while for the latter approach, the ones that let the user assess the quality of an item best are the finest choice. Still regarding explanations, Bilgic and Mooney [2005] state that a good explanation should hold several desired properties in order to provide a better user experience. An explanation should be transparent, showing how the systems works, while it increases the user's trust and satisfaction when using the system. Other point that it should seize is to help users to make fast and good decisions, and in some cases to persuade them to try a certain item.

2.1.4 Evaluating Recommendation Systems

Evaluating recommendation systems is not a trivial task since the quality of the recommendations produced are inherently linked with the algorithms used (their performance can change with the dataset or domain) and with the main goal of the system. Thus, deciding what combination of measures to use is a significant challenge that must be taken for each different recommendation system and comparative evaluation [Herlocker et al., 2004]. Evaluations can be completed using both offline analysis, quickly and economically predicting certain values from a dataset, and with live user experiences. With offline analyses we must take into account the natural sparsity of ratings and that they will not be able to determine which is the particular reason for that users prefer a particular system. In live experiments, there is the ability to control the experiments when the system is made available to a community of users and to observe particular user behaviour interacting with the system and by what means the recommendations are influencing them.

Until today, there have been a major focus on studying the evaluation of a recommendation system's accuracy. Herlocker et al. [2004] propose three recommendation accuracy metrics classes: 1) *predictive accuracy metrics* measure how close the recommendation system's predicted ratings are to true user ratings; 2) *classification accuracy metrics* measure how many times a system has a correct judgement about whether an item is good for the user or not; 3) *rank accuracy metrics* measure the ability of an algorithm to produce an ordered list of items that matches the opinion of the user. However, there has been a recognizing that a recommendation system should also be judged in terms of its *usefulness*, and so evaluators should keep in mind aspects like the coverage of recommendations, the learning rate of the system, and which are the levels of novelty and confidence of the given suggestions.

2.2 Types of Recommendation Systems

Content-Based. In *content-based recommendation* the system learns to recommend items that are similar to those a given user has liked in the past. Thus, in content-based recommendation we can easily achieve user independence and quickly generate recommendations for a new item on the database, since the recommendations are only based on the set of item features [Balabanovic and Shoham, 1997].

However, due to its nature, if the system does not contain enough information to discriminate items the user likes from items the user does not like it will not be able to provide suitable suggestions. There is also an inherent over-specialization problem, since it will not produce novel recommendations, due to the tight link with items that the user has already rated. Finally, when a new user comes into the system it can be a problem, since enough ratings have to be collected before the system can create an adequate user profile.

Collaborative Filtering. In a pure *collaborative filtering* (CF) recommendation one identifies users whose tastes are similar to those of the given user and recommends items they have liked, never doing any analysis over the items at all [Balabanovic and Shoham, 1997]. For this reason it is said that it uses the common principle of *word of mouth*.

Pure collaborative recommendation solves most of the shortcomings given for pure content-based systems. 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 individual user. However, this approach is not ridden of problems: a user profile for a new user will only be valid after making some few 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 trough 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. Other problem is that it will create poor recommendations for a user whose tastes are not according to the majority of the population.

We can split CF into two major categories:

- ***Memory-based:*** Basicly, consists on comparing users against each other directly using correlation or other measures. Algorithms that fit into this category make predictions based on the entire collection of previously rated items by the users. In other words, the value of unknown ratings will be computed as an aggregate of the ratings of some other users for the same item. This category is the most popular and widely used. The most known types of approaches under this category are:
  - *User-Based:* User-based CF was first introduced by Resnick et al. [1994]. This kind of approach predicts an active user's interest in a particular item through the rating information from users with similar preferences. This is, to users with similar profiles, where the profile

is a row vector sorted in a user-item matrix M. In this matrix, the position M_{ij} stands for the rating value that user i gave to item j. This approach starts by measuring the similarities of any two row vectors. Then, for predicting the rating of a user for a particular item, we use the ratings information of the N most similar users to the active user. Once the neighborhood is defined we set the user's preference for a particular item as the weighted average (on the similarity) of the ratings that the neighborhood gave to that item. In sum, user-based CF finds other users whose past rating behavior is similar to that of the current user and use their ratings on other items to predict the active user's preference for an item he has not rated.

  - *Item-Based:* These approaches were first described by Sarwar et al. [2001] and use the similarity between items instead of users. After the similarity of items are calculated (similarity between columns in the user-item matrix), unknown ratings can be predicted by averaging the ratings of the k most similar items rated by the active user. This follows the idea that if two items tend to have the same users like and dislike them, then they are similar and users are expected to have similar preferences for similar items. In its overall structure, therefore, this method is similar to earlier content-based approaches to recommendation and personalization, but item similarity is deduced from user preference patterns rather than extracted from item data. The main benefit of this approach over the *user-based* one is its scalability. In systems with a sufficiently high user to item ratio, however, one user adding or changing ratings is unlikely to significantly change the similarity between two items, particularly when the items have many ratings. Therefore, it is reasonable to pre-compute similarities between items in an item–item similarity matrix.

- ***Model-based:*** Analyzes historical data to identify relations between different items so as to develop a model of users' ratings. In this process, a probabilistic approach is used to compute the expected value of a user prediction, given its ratings on other items, based on the found relations. Although developing models requires a great amount of time, this paradigm handles the sparsity better than the memory-based approach and provides quick recommendations [Adamavicius and Tuzhilin, 2005][Deshpande and Karypis, 2004][Sarwar et al., 2001].

Knowledge-based. This type of recommendation systems uses domain knowledge over all the recommendation space, both users and items, to generate a knowledge-based recommendation, reasoning about how products meet the user's requirements. In these type of recommendation systems it is needed knowledge on the objects being recommended and on the user. More important is the existence of functional knowledge, that makes it able to map between the user's needs and the object that might satisfy those needs. Successful knowledge-based recommendation systems are *case-based*, which exploit similarity metrics and *constraint-based*, which exploit rules that relate customers and products over the knowledge base [Burke, 2002][Burke, 2000].

With knowledge-based we do not have problems on the appearance of new users or new items since its recommendations are totally independent of users' individual tastes, producing always high quality suggestions. Due to these properties, these types of systems are highly complementary to other types

of recommendation systems. However if they are not able to learn, then they may be surpassed by other shallow methods. Other catching drawbacks are the need for knowledge acquisition, and the inability to discover user niches, since it can only make recommendations as wide-ranging as its knowledge base allows.

Dimensionality Reduction. On the *collaborative filtering* algorithms described above it is usual to look to the user-item ratings domain as a vector space. This way, we can see a user as an $|I|$-dimensional space, where I is the number of items, and items as an $|U|$-dimensional space, where U is the number of users. However, as these dimensions tend to get larger, also the redundancy enclosed on them gets larger. Due to this fact, there was a trend on trying to understand if it was possible to reduce these users and items dimensionality, such that the redundancy was avoided. Basicly, this consists in identifying a set of k topics, so that the user preferences on an item can be expressed as a combination of the user's interest in a topic and the extent to which each item is relevant to that same topic.

Some recommendation systems try to achieve this using either content featuring, tagging or clustering. However, the most novel used technique is *Latent Semantic Analysis* (LSA) [Deerwester et al., 1990], which provides a way to discover theses topics using only rating data. This technique bases itself on the belief that the topics are latent in the rating data. LSA has a lot of sucess in dealing with high-dimensionality representation, and is able to address several problems, such as sinonymy (different items reaching the same topic) and polisemy (an item with relevance for different topics). After the Netflix challenge [Koren et al., 2009], there was a huge trend to use the so-called *latent factor models*, aiming for revealling the hidden latent features that somehow explain the observed ratings. One of the most applied techniques with these models is the matrix factorization technique, *Singular Value Decomposition* , which enables the creation of a k-dimensional approximation of the original matrix using the hidden factors discovered.

Hybrid. Although the known methods presented above behave well in particular domains, sometimes there is a need to combine them into hybrid recommenders that will show performance improvements. According to Burke [2002], in hybrid recommendation systems the combinations of two or more recommendation techniques can be done through many different approaches :

`Weighted.` The score of a recommended item can be computed from the results of all the available recommendation techniques present in the system;

`Switching.` Some criterion can be used to switch between recommendation techniques;

`Mixed.` Presenting simultaneously recommendation from more than one technique;

`Feature Combination.` Collaborative information can be treated as an additional feature associated with each example and use content-based techniques over the augmented dataset;

`Cascade.` We can perform a staged process, where one recommendation technique is employed first to produce a coarse ranking of candidates and a second technique refines the recommendation from among the candidate set;

`Feature Augmentation.` One technique can be used to produce a rating for an item and that information is then incorporated into the processing of the net recommendation technique;

`Meta-level.` Here the model generated by one technique is used as the input for another.

Chapter 3

Recommendations In Education and Student Modeling

In this chapter, we will start by contextualizing the current paradigm of *educational data mining*. Then, following our goal of recommending items (courses) to users (students) in an educational context, we will structure and present the state-of-the art on recommendations in education through the use of an evaluation framework defined by us. This framework allows to categorize each work according the *item recommended*, *data mining techniques* used, *recommendation techniques* used and the *users* that are the target of the recommendation. Then, as we want to be able to profile students correctly, so as to be able to model accurately their skills, we will overview the state of art on student modeling and performance prediction. We will end by analyzing which are the current open issues on the state-of-art related to our research problem.

3.1 Educational Data Mining

In the last decade there has been a growing amount of interest in a research field related to the development of methods that use information gathered from several sources of an educational environment in order to gain better knowledge over the several components of the educational process. This field is known as *educational data mining* (EDM) and has a tight relation with known methods of data mining, machine learning, statistics and even pedagogical theories.

The success of *data mining* (DM) when applied to several types of business has its given proofs, due to its ability to extract, discover and present, in an interpretable way, interesting and novel patterns from large amounts of information. The application of these techniques regarding an educational context is recent, and owe it to the latest progresses on educational technologies: we are now able to retrieve large datasets from many sources of educational information. Similar to DM, but applied to educational data, EDM starts by pre-processing the raw data (gathered from many potential sources), so as to apply the most appropriate data mining techniques and then performs the post-processing, resulting in useful structured information that has the potential to better understand the learning process on education, the

settings in which they are based on. Despite the similarities with average DM, in the educational context it is possible to exploit several levels of meaningful hierarchy in the data: for example, the data of the answer, student and teacher dimensions, with several different levels of aggregation such as student, teams and classroom.

With the expansion of internet, it is now feasible to spread education through web-based systems: nowadays we can use *e-learning* systems, that provide online schooling, *learning management systems*, that ease the communication between teachers and students, and there are also *Intelligent Tutoring Systems* and *Educational Hypermedia Systems* that aims to adapt to the particular needs of a student, providing the best resources to improve its knowledge.

According to Cristóbal and Romero in [Romero and Ventura, 2010], we can split EDM into eleven main categories as follows:

1. `Analysis and Evaluation.` For highlighting useful information to support decision making using statistical and visualization techniques.

2. `Providing Feedback for Supporting Instructors.` Providing feedback on new information to support course authors/teachers/administrators in decision making.

3. `Recommendations for students.` Recommendations to students according to their personalized profile.

4. `Predicting Student Performance.` Estimating the unknown value of a variable that describes the student profile.

5. `Student Modeling.` Developing cognitive models of students, including their skills and knowledge.

6. `Detecting Undesirable Student Behaviours.` Detecting those students who have some type of problem of unusual behaviour (cheating, low interest, academic failure)

7. `Grouping Students.` Creating group of students according to their personal features

8. `Social Network Analysis.` Studying relationships between individuals

9. `Developing Concept Maps.` Helping instructors/educators in the automatic process of developing concept maps

10. `Constructing courseware.` Constructing process of courseware and learning contents automatically

11. `Planning and Scheduling.` Planning future courses, helping with student course scheduling, planning resource allocation, helping in admission and counselling processes, etc.

Although there has been a rising interest on this research field, there is still work to be done, as it exists a lack of standards on data and models, tuning of algorithms in the context of education and an integration of data mining tools with the e-learning systems [Romero and Ventura, 2010].

Due to the sucessful applicalibility of both educational data mining and technology enhanced learning, there has been an increased interest in the application of recommendation systems on this particular domain. Hence, there is already a large number of recommendation systems under education settings, that distinguish from other recommenders (e.g product recommenders) for the different goals they try to achieve. Still, there is a need to identify particularities of education-focused recommendation systems, so as to provide a well defined design, development and evaluation.

3.2 Evaluation Framework

In this project we suggest the use of an evaluation framework to properly classify the behaviour of a recommendation system in educational data mining, according to a fixed set of well defined dimensions of analysis. With this said, we can see our framework as having four different scopes of dissection, that can be defined as follows:

Items recommended. In educational data mining, there is a large span of work on recommendation systems with many different goals, that differ mostly about the type of item being recommended. Thus, it is a crucial dimension to scrutinize when talking about recommendation in an educational context. The items recommended can be categorized into :

1. *Learning materials (LM):* Any type of courseware content, lesson or resource that best suits the active student's needs.
2. *Feedback*: Information about some behavior in a specific context that that can be taken into account to influences the same phenomenon in the future.
3. *Groups*: The recommendation is over how to form the best working teams.
4. *Courses to take:* The object of recommendation can be a set of courses to enrol on.

Users. Since a recommendation system on its own its worth nothing it is pretty reasonable that our second dimension of analysis classifies the recommendation system according to its users and stakeholders. With our educational context in mind we can easily partition this dimension in three different types of stakeholders:

1. *Students:* They are the main interested when it comes to receive personalized activities, learning paths, courses and learning resources (books, links, papers).
2. *Educators:* Their interest is in recommendations on how to make them able to provide better learning practices, to detect undesirable student behaviour and even on how to predict student performance.

3. *Course developers/Learning Organizations:* The significance of recommendations is on which are the best course contents and how they could improve student learning. They can also be interested in knowing which courses are more valuable for a class of learners, for which applicants they should be accepted and which ways would bring more value to their courses.

Recommendation technique. Despite the continuous study on developing better and novel techniques to produce more accurate recommendations, there are well defined categories of recommendation techniques as seen in section 2.2. Thus, a recommendation system can be categorized, according to its technique, as follows:

1. *User-based CF*
2. *Item-based CF*
3. *Content-Based*
4. *Knowledge-Based*
5. *Dimensionality Reduction*

A category to hybrid recommender systems is not needed since we can see them as a combination of characteristics of other recommendation techniques, and in that case they can be categorized as using more than one technique.

Data mining technique. Since EDM also lies on data mining techniques, the vast majority of systems will use one (or more) of the known different approaches. Hence, we can classify each recommendation system according to the data mining approaches they use to extract meaningful information that sustains the basis of its recommendations. We can split it into two major categories:

1. *Unsupervised Techniques*. This category is defined by the set of techniques that do not use any external knowledge to guide the knowledge discovery process, while searching for any regularity in the data. In other words, the correct results are not given in the training process. The most popular examples on this class of techniques are *clustering* and *association rules*.

2. *Supervised Techniques.* Contrarily to the techniques described above, this set of techniques can use knowledge about the data in order to create an accurate model that is able to correctly classify instances of the domain in analysis. This knowledge is nothing more than a label or class to each known instance. This way, after the training, the created model will be able to generalize a classification to each new instance. In this group, we can find several methods, such as *decision trees*, *bayesian networks* and *neural networks*.

3.3 Recommending in an Educational Context

The *recommendation problem* is general enough to be applied to any domain, and although its main uses are nested on entertainment and e-commerce, there is also some work done on recommending items in the education domain, either directly or indirectly, with the goal of improving the learning process. We will review the most important literature regarding these type of recommendations in terms of the *items recommended* dimension of analysis defined in section 3.2. In table 3.1 you can see each work is categorized according our evaluation framework. We did no display any work related to *dimensionality reduction* as we did not found any research that approached this topic.

3.3.1 Learning materials

A lot has been done in recommending material, resources and other types of learning contents to students in order to overcome the difficulty of locating suitable learning materials in the enormity of available resources.

Using only recommendation techniques we have three systems with a user based CF approach. First, Soonthornphisaj et al. [2006] developed a *smart e-learning recommender system* that evaluates each learner using a quiz and then recommends the most suitable learning materials to the active learner according the ratings given from the set of users most similar to him in terms of expertise. *PeRes* [Zhu et al., 2008] is a proposed SCORM (Sharable Content Object Reference Model) compliant multi-agent system, that along with several personalized services, offers a recommendation service that presents to the learner what he should learn next, using a balanced user-based CF algorithm that takes into account both the active learner's goals and interests. However, no details are given in how that balance is achieved. In Ge et al. [2006] system, a user starts by inputting keywords about the courseware he wants to learn. Then, the system retrieves a group of users that denote the same interests as the user being recommended (user-based CF). This way, the system is able to recommend the coursewares that are rated highly according to the members of that group and that contain information about the keywords searched by the user. Hsu [2008] combined user-based CF with an unsupervised technique to recommend lessons to students. The author used clustering to create groups of students that exhibit the same study behaviour. With the clusters defined, association rules mining is applied to analyze the association of lessons in each cluster. The rules retrieved from the association rules mining are then used to recommend lessons to a student, according to the cluster to where the student belongs.

Looking into item-based CF, we have an e-learning framework, presented by Ghauth and Abdullah [2010], based on the idea of recommending learning materials with a similar content to those of the item the student is viewing, and sorting them according the rating that good students' gave them. If the item has been rated by good students, its rating is automatically extracted. If not, the rating is estimated by taking into account the item similarity to all the items stored in a item-item matrix and their respective rating. With the intent of integrating the collaborative and content-based techniques when recommending e-learning materials, a framework has been created by Lu [2004]. This framework starts by identifying the student's learning requirements with a multi-criteria analysis that takes into account

Table 3.1: Distribution of literature according our evaluation framework.

Categories	LM for students	LM for educators	Courses to students	Feedback for students	Feedback for educators	Groups for students	Groups for educators
Supervised Content based	Lu [2004]						
Unsupervised Content based	Lu [2004],Hsu [2008]						
Supervised User-based CF	Lu [2004]		Vialardi et al. [2009]				
Unsupervised User-based CF	Lu [2004],Hsu [2008]		Taha [2012]		García et al. [2009]		
Supervised Knowledge-based							
Unsupervised Knowledge-based	Krištofič [2005]				García et al. [2009]		
Supervised Techniques	Su et al. [2006],Baylari and Montazer [2009]	Su et al. [2006]	Sandvig and Burke [2005],Vialardi et al. [2010],Wang et al. [2009]		Chu and Hwang [2006],Romero et al. [2005],Vialardi et al. [2008]	Crespo and Antunes [2013]	Crespo and Antunes [2013]
Unsupervised Techniques	Su et al. [2006]	Su et al. [2006]	Bendakir and Aimeur [2006],Chu et al. [2003]		Romero et al. [2005],Vialardi et al. [2008]		
Content Based	Tsai et al. [2007],Ghauth and Abdullah [2010]		Unelsrød [2011],O'Mahony and Smyth [2007],Wang and Yuan [2009],Ranka et al. [2010]				
User-based CF	Ge et al. [2006],Soonthornphisaj et al. [2006],Zhu et al. [2008]	Zhu et al. [2008]	Unelsrød [2011],Surpatean et al. [2012],Farzan and Brusilovsky [2006]				
Item-based CF	Ghauth and Abdullah [2010]		O'Mahony and Smyth [2007]				
Knowledge-based	Shen and Shen [2004]		Sandvig and Burke [2005]				
Others			Ranka et al. [2010],Parameswaran et al. [2011],Bercowitz et al. [2009],Wang and Yuan [2009],Karampiperis and Sampson [2005]	Gutierrez et al. [2012]	M-Helene et al. [2006]	Rovira-Asenjo et al. [2013]	Rovira-Asenjo et al. [2013]

his learning style, learning material accesses and achievements. With the definition of the requirements, the neighbours of the student can be identified so as to recommend the most prefered learning materials among them. Using neighbours information, a set of fuzzy logic rules that relate students' requirements with learning material contents are generated. Hence, the production of recommendations consists in collecting the learning materials contained in rules that match the student's requirements.

With a content-based approach, Tsai et al. [2007] showed that a learner can retrieve desired learning materials by inputting a sentence with keywords that will be analyzed to discover and recommend learning objects with similar contents to the keywords combinations - the presumedly learner's intention. First, courses maps are constructed from keywords in webpages or learning objects, so as to create a concept ontology. Then, when a user expresses his intention (a set of features), the system gathers the courses with the most pertinent contents. Each Learning Object from these courses is evaluated according the preference the user has shown for features that the object holds. In the end, the recommended Learning Objects will be the ones that the user is more likely to prefer.

Following a knowledge-based approach, Shen and Shen [2004] recommended learning objects to learners, using competency gap analysis, by mapping the active learner target competency (based on the current and goal competencies of the learner) to concepts of the learning objects. This mapping is done through sequencing rules that use the concepts knowledge base. Still using a KB approach, but incorporating some unsupervised techniques, Krištofič [2005] presented a recommendation system incorporated in an adaptive hypermedia system, that recommended the lessons that students should study next by matching the concepts on the current user session with the concepts on the knowledge base. To achieve its goal it uses sequential and traversal patterns and association rule mining to represent automatic student knowledge.

Su et al. [2006] focused in using both supervised and unsupervised methods to recommend learning materials. They used a four phase model that tries to understand the grades of a student : 1) define a user model as learner profile, 2) apply sequential pattern mining to extract the maximal frequent learning patterns from the learning sequences within a learning portfolio and clustering (so as to create sets of profiles of good students), 3) construct decision trees to classify each new student to one of the identified clusters and 4) generate a personalized "activity" tree, with sequencing rules between the learning concepts, extracted from the selected cluster for a new student. Baylari and Montazer [2009] uses only one of the supervised techniques to recommend learning objects to students. They developed a multi-agent framework based on Item Response Theory and neural networks that is able to estimate learners' ability. The students will answer to some specific tests that estimate their ability and learning problems so that the system can recommend personalized materials that aim to solve the diagnosed learner's problems and to offer a continuous learning according to the learners' expertise.

3.3.2 Feedback.

Although little work was done in recommending feedback, there are some interesting approaches using mainly data mining techniques, both supervised and unsupervised. We will overview some other ap-

proaches that do not fit on any dimension of our evaluation framework, but that are still interesting and that deserve a mention.

The unique approach to recommend feedback using recommendation techniques is mainly directed to teachers and courseware authors and was proposed by García et al. [2009]. They ground their recommendations over modifications on courseware using both *collaborative filtering* (CF) and *knowledge-based* (KB) techniques: CF will find relationships on the mined association rules among teachers with similar profiles, while KB will be strengthened by significant and value-full experiences that can give rise to effective recommendations. Other tool to support teachers and course designers on how the course structure and contents can be enhanced is promoted by Vialardi et al. [2008]. It uses both supervised and unsupervised techniques - decision trees and association rules - to build a model on students' behaviour on a course. This model can be used to show teachers how adaptation in adaptive hypermedia systems is working for different student profiles, thus supporting decisions on the course design.

Romero et al. [2005] developed a mining tool that uses decision trees, association rules mining and grammar-based genetic programming to discover association rules on the usage data picked up during students' e-learning sessions. With this tool the course author can visualize the discovered rules, constrain the presented rules and decide on how it can improve its course.

More leaned into student-focused feedback, Chu and Hwang [2006] analyzed test results to suggest correctional behaviours to learners. Using the incorrect answer rate for each student and concept, the author approaches the identification of the poorly learned concepts through the fuzzification of those rates. This way, the system is then able to indicate the student which are the concepts that are dominated, and those that require more theoretical study or practice.

On other sight, and with support of the *Information Retrieval* area, Gutierrez et al. [2012] used ontologies to grade and detect errors automatically. The author was able to detect both logically and incorrect sentences from summaries, using both extraction rules and classification techniques. The detection of errors is seen as the raw basis for helping a student to understand which concepts need more study.

An alternative approach to the recommendation and data mining techniques was proposed by M-Helene et al. [2006]. A special compiler used by students to solve programming tasks provided logged data, with information over their interactions. This data was mined using statistical information and the edit distance to the correct solutions so as to identify the most common errors and provide contextual feedback.

3.3.3 Groups

As in many domains of life, group activities and teamwork are important pieces of the education domain. Thanks to the expansion and increasing interest in the area of *Computer Supported Collaborative Learning* it was noticed an appearance of works with the goal of studying how best to recommend groups.

Recently, Rovira-Asenjo et al. [2013] showed that, although difficult, it is possible to predict future conflicts between team-members so as to identify less problematic groups of people. To solve this

trusting issue, they confronted social network analysis with group-based models, to found that it is easier to predict conflict in small teams using the probabilistic treatment given by the latter. Other progresses were made by Crespo and Antunes [2013] that proposed *educare team adviser*, a novel recommender system that suggests new teams in the context of a specific course. In particular, a student can ask for the best group from all available students or from the set of previous teammates. The system is composed by three main components : 1) a *profiler* that profiles each user and student in the list of available students enrolled in the course using social network analysis, 2) a *teamwork classifier* that uses a classification algorithm to predict a given team performance and 3) a *recommendation module* responsible for answering the query and suggesting a list of potential groups for the student.

3.3.4 Courses to take

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. Course recommendation systems have been suggested as the tool that should be able to provide the guidance that students need.

Wang et al. [2009] proposed to discover the most adaptive learning sequences for a teaching concept using students' profiles - a special case of the courses sequence recommendation. Employing a decision tree algorithm, grounded on a set of variables that define the student profile, they found nine learning sequences that optimize students outcome and fifteen optimal learning groups that help the teacher to categorize each student. With a similar goal, Karampiperis and Sampson [2005] designed an AEH (Adaptive Hypermedia System) that first generates all possible learning sequences that match the learning goal and then selects the best one according the user preferences and learning characteristics. This way, it escapes to the usual approach of generating the learning path by populating a concept sequence with available learning resources based on pre-defined adaptation rules.

AACORN [Sandvig and Burke, 2005] 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. This way, 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 query 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 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.

Farzan and Brusilovsky [2006] created *CourseAgent*, an adaptive community-based recommender system for course recommendation that takes into account the student's carreer goals and both courses workload and relevance. It uses both implict and explicit feedback given by the users to distill the collective wisdom of the community. The system tries to provide recommendations in the form of in-

context visual adaptive annotations instead of generating a sorted list of recommended courses: with each course recommendation it presents the expected workload, difficulty and relevance to the students' career goals. The recommendation output is grouped by areas of study. The authors try to overcome the problem of lack of ratings by making the achievement of a personal goal dependent upon their contribution to the community. This way, if the students want to see the effect of a course taken to its career goals they have to rate that specific course in terms of workload, relevance and difficulty.

In 2011, Unelsrød [2011] developed a recommender system for course selection in higher education. The system has a component that uses a specialized memory-based CF: weights each user based on their chosen degree major 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 in the accuracy of the recommendations when focusing on users with common degree major. 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.

Surpatean et al. [2012] proposed a user-based collaborative system for recommending masters programs. The system uses the academic profiles of alumni students labeled by the masters program they have chosen. Then, given the academic profile of a student, the recommendation system returns the set of masters programs of the alumni students whose academic profiles are among k-closest in the training data. They used both ECTS (ects of the courses) and binary representation (if the student enrolled or not on the courses) on the students' academic profile and the former representation was the one that has shown better results. The authors also used both *cosine similarity* and *Tversky index* as similariy metrics when retrieving the k-closest academic profiles, but no major difference between them was recorded. Also with a collaborative edge when recommending courses to students, Vialardi et al. [2009] developed a system based on data mining techniques. In particular, the recommendation system uses the data of an historical database of students and results obtained by term, with the goal of discovering patterns from the rules produced by decision trees. Those patterns indicate if a student is likely to pass or fail a certain course and can be used to suggest recommendations over a student's enrollments.

Taha [2012] has a singular approach to collaborative filtering using the clustering technique. On the offline phase, divides students in biclusters that take into account their skills and performance. With this, as new students appears looking for counselling on their enrollments, the system only has to identify to which cluster each student belongs and then recommend the courses that were rated high among the members of the bicluster.

O'Mahony and Smyth [2007] designed a system that integrates two different recommendation systems. The first recommendation system uses collaborative filtering technique to suggest elective courses, presenting the courses that reveal the past choices of like-minded students. This recommendation system has a variation to the widely-used-item based CF: it constructs student profiles - abstracted to the level of the subject code - using two matrices that record the core and elective courses that are selected by students. Then for each elective subject, computes the total similarity between all core subjects selected by the student. Individual elective courses are then output - grouped and ranked by subject code. The second recommender system is content-based, which recommends courses based on keyword

similarity.

Using BM25 ranking function, Wang and Yuan [2009] employed content-based filtering, to measure the relevance of the matching courses to a query. The authors suggested an improvement to BM25 so as to be able to handle situations where different terms from multiple topics should have different weights. When a term is match, its weight according the student's interest level is considered. To feed recommendations, *Information Retrieval* techniques use the course title and description to build an inverted index for courses while students' interests are extracted given each course selection type: mandatory or free-choice (from a predefined list of courses or from all courses). In the same line of thought, *Pundit* [Ranka et al., 2010] is a content-based recommender system that employs a data retrieval and mining approach to recommend courses by matching students' profiles and course contents. The authors built student profiles using past data on courses and academic background. They also collected meaningful information about courses contents to constuct the course "inverted index". Finally, they suggested an improvement by defining an evaluation function is defined to determine the goodness of recommendations.

Vialardi et al. [2010] presented a decision tree based recommender 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 of a student has obtained in the prerequistes of that course - and courses difficulty - represented by the average of the grades obtained by students. 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.

In the set of approaches using only unsupervised techniques, we have Bendakir and Aimeur [2006] who created *RARE*, a course recommender system based on association rules. It starts by extracting association rules from previous course selection data that relates academic courses followed by former students. These rules are later used to infer recommendations. To improve the experience, *RARE* lets students to rate the recommendations, which may result in rules improvement by adding or removing courses from rules. Chu et al. [2003] course recommendation system also takes into consideration students' preferences. To find out which were the students prefered courses categories it uses Prediction Methodology, that is based on both association rule mining and graph theory.

With the intent of providing flexible recommendations, Bercovitz et al. [2009] created *CourseRank*, a course recommendation system that uses traditional recommendation operators (like Jaccart and Pearson similarity) and query operations to express filters declaratively over structured data. Using *CourseRank*, students can search for courses of interest, get personalized recommendations and check if all requirements are met to take a chosen course. The requirements of each course are captured through a simple formal model developed by the authors. Parameswaran et al. [2011] approached the problem of recommending a set of items (in this case, courses) that satisfies several constraints. The authors approached the problem in the context of *CourseRank* recommending courses that not only help to satisfy constraints(e.g., take two out of a set of five math courses), but that also experience high popularity among similar students. To achieve this, the authors developed two models to express

requirements: a core model whose requirements can be checked by flow algorithms and an extended model that uses Integer Linear Programming to validate the requirements.

3.4 Student Modeling and Performance Prediction

The other outlook we give is to the literature on student skill modeling, as well as into student performance prediction. We focus on these aspects with the goal of understanding students' behavior and which factors have major influence on their academic performance. If we are able to to correctly model students knowledge then we may be capable of providing course recommendations that comply with the students' skills. Besides that, students will take more advantage of recommendations where they are predicted to have better performance.

Thai-Nghe et al. [2010] used matrix factorization to predict student performance. This factorization is based on a proposed mapping from the usual recommendation context to an educational context, where a student is seen as a user, while the correct first attempt indicator was mapped into to a rating. Other approach was taken by Baker et al. [2011]. The authors introduced a model to predicts students' performance on a post-test by measuring how a student skill will be helpful in future learning. Based on a combination of theory and prior work detecting related behaviors, they have came up with a set of features (pauses after hints, help avoidance, off task behavior) taken from e-learning log data to be used as input to the skill prediction cross-validation correlation algorithm. An alternative approach was proposed by Wang et al. [2010], on how to represent student performance measures. Using a naïve approach, they replaced traditional binary measures (right or wrong) with continuous partial credit, which gives a score or penalty according the number of tries to get a correct answer, and achieved slightly better results. Given a set of different difficulty English classes, Martinez [2001] tried to predict in which of these classes the student would have better performance. Discriminant function analysis was used to produce functions that help to define known groups. The functions first seek to distinguish the first group from the others, then the second group from the rest, and so on. Hence, Martinez uses the produced functions to predict which is the English class that suits better a specific student.

With the goal of predicting students' future responses, Bergner et al. [2012] proposed an alternative to the Item Response Theory. The authors used an item-based collaborative filtering technique so as to analyze student response matrices and find parameters for students and items that when combined show predictive power over student performance, on an item by item basis. Thus, instead of assigning an item response model *apriori*, they opt to use collaborative filtering to train a class of log-linear models on the data and select the one that performs the best in terms of accuracy.

Following a knowledge-based approach, Hien and Haddawy [2007] proposed a methodology to predict applicants' CGPA (cumulative grade point average) on university. A case-based retrieval mechanism was derived from the Bayesian network predictive model in such a way that the similarity measure used by the case-based system is consistent with the predictive model. The case-based mechanism is used to retrieve the past student that has the most similar background with the applicant. Once that student is found, it is "predicted" that the applicant will have the same performance. The particularity of the case-

based component is that it expresses the similarity between two different values of an attribute as the inverse of their dissimilarity, which is estimated by the difference between two expected CGPAs given each value of the attribute.

Kovacic [2012] proposed the usage of data mining techniques to predict if a student would pass or fail the course using only enrollment data. The feature selection showed that the most important factors to distinguish sucessful and unsuccessful students were ethnicity, course program and which trimester the students were in. Among the classification techniques, the most sucessful was the Classification and Regression Tree (CART). However, Kovacic stated that only enrollment data is not sufficient to separate the successful from the unsuccessful students. With a similar approach, Barker et al. [2004] tried to distinguish the students that graduate from the ones that do not graduate. Feeding both neural networks and support vector machines with students' demographic and performance variables, Barker approaches training and testing in three different ways : training with data from one year and test with the following year; training and testing only in the same year; and training and testing with all years. Although the results differed little, it appears that training with data from all years results in a more generalized and accurate classification. Kotsiantis and Pintelas [2005] also used demographic and performance variables as the training set for students' grades prediction in an Hellenic Open University. Behind the idea of estimating the quality of attributes according to how well their values distinguish instances that are near to each other, the authors found a small set of predictor variables. A comparison of several regression models exposed a superior accuracy by the MD5rules algorithm. Other interesting remarks are that among the demographic variables 'sex' and 'number of children' were the most relevant while the grade of the first written test was not relevant since students always try harder on their first evaluation.

Targeting the prediction of GPA, Zimmermann et al. [2011] explored the statistical relationship between a mandatory BSc and the performance on a freedom of choice MSc. A random-forest algorithm was used to estimate decision trees for regression on random partitions of the data while predictions were evaluated using an out-of-bag scheme. The classification was done on a small set of predictor variables, that were able to explain 55% of the cumulative grade point average variance. The authors concluded that the third undergraduate year was the most informative for future performance. It was also shown that models based on grades from final attempts (in the case of repeated exams) yelded a significantly higher prediction accuracy.

Romero et al. [2008] compared several data mining methods to classify students according Moodle usage data (number of assignments, total assignment time, mark, etc.) into one of four categories: excelent, good, pass or fail. The authors also applied discretization and rebalance preprocessing techniques on the original numerical data in order to verify if better classifier models were obtained. Their conclusions were that some algorithms, such as Kernel, KNN, AprioriC, Corcoran, AdaBoost and LogiBoost are not affected by rebalancing while the two decision tree methods (CART and C4.5) give worse results with rebalanced data but most of the algorithms (all the rest, 17 out of 25) obtain better results with the rebalanced data. The authors also remarked that the classifiers don't distinguish well the excelent students from the good or pass students. KABAKCHIEVA [2001] took an analogous approach using some popular classifiers (C4.5, NaiveBayes, BayesNet, kNN, OneR and JRip) to classify students into

five classes : 'Excelent', 'Very Good', 'Good' , 'Average' and 'Bad'. Kabakchieva also concluded that the decision trees provided the best results and that the predictions are worst for the Excellent class, and bad for the Average class with the kNN being totally unable to predict any of them. Shangping and Ping [2008] also approached students' final grade prediction based on features taken from logged data of an e-Learning system. Taking advantage of genetic algorithms, the authors created the ARMNGA algorithm, that avoids generating impossible candidates and therefore is more efficient in terms of execution time. This way, they launched themselves into the discovery of association rules on which their predictions would be based on.

Trying to foresee poor performance, Agapito and Ortigosa [2009] have proposed a data mining based two-step approach to find patterns that can be seen as symptoms for future low performance. They start by generating production rules using the C4.5 decision tree on the logged information describing student's e-Learning system usage. Then the rules are filtered so as to select the most relevant ones. Although three rules have been found as indicators to future low performance, it was clear that decision trees and production rules are strongly related to the distribution of the data, and therefore small variations could end in different conclusions.

Superby et al. [2006] approached the problem of finding which were the most correlated variables when predicting the risk of failure associated with a student (low, medium or high). First, the authors state that a student's success is influenced by personal history, behavior and perceptions on the academic context. This way, they sought to find which were the most significant variables associated with each factor. The grade of the last year of secondary was the most decisive on the student's personal history . On other hand, the number of hours a student's states to attend classes its the most significant regarding behavior. Although the perceptions of the student are highly subjective, the authors stated that the confidence of the student on his/her abilities were highly important. Despite not achieving remarkable prediction rates, the authors claim that discrimant analysis, neural networks and random forests seem to be able to lead to interesting results.

Bresfelean et al. [2008] used clustering to find relevant features that are able to draw up a student's profile and then feed a classification algorithm for exam failure and success prediction. After an initial preprocessing, the features were selected using Gain Ratio feature evaluator through the Ranker method. Then, for the clustering process the FarthestFirst (FF) method was used. FF algorithm works as a clustering approximation modelled after simple k-means. It starts by randomly selecting an instance to be the cluster centroid. The instance that is farthest away from its centroid is selected as another cluster centroid. This process is repeated until the number of clusters is greater than a specified threshold. The authors chose to produce only two clusters : the students who passed all exams and the ones who failed one or more exames. The highest ranked attributes on the clusters were the students desire to continue or not their education with post university studies, as well as the fulfillment of their prior expectations. Classification was used to predict the students' success on passing the academic exams based on their behavioral profile.

Antunes [2010] proposed the use of *as soon as possible classifiers* to predict students' dropout. These classifiers try to predict the class of an instance only considering the first n of m attributes. On a

first phase, a classifier is built on the entire dataset. Then, for each unobservable attribute a_i, creates a subset of instances with information from a_1 until a_i, where a_i is the class. This is done to find a set of *class association rules* that will be used to predict missing values. Hence, when classifying an instance with missing attributes, the values of this one will be calculated using the class association rules produced for each unobservable attribute. This method can be applied to student failure prediction by using the information on the first work assignments.

Adaptive learning requires the identification of different students' learning styles so as to provide the best learning materials and strategies. Chang et al. [2009] approached this problem with a mechanism that improves kNN classification by combining it with genetic algorithms (GA). Their algorithm tries to fight three major kNN weaknesses: i) if a large number of learning behavioral features need to be considered, it results in a heavy computation complexity; ii) when there is a large quantity of samples, the process of computing distance is time-consuming; (iii) after the execution of k-NN classification, establishing order in class ranks, when classes have the same number of samples is difficult to be determined. To solve those weaknesses they employ GA, along with pre-constrast and post-comparison algorithms, to extract the learning behavioral features and then to reduce computation complexity.

Ogor [2007] enables real-time student performance prediction upon graduation using assessment-based performance monitoring and data mining techniques. Using C4.5 decision tree algorithm, rules are derived with the goal of enabling the classification of students into classes. This research concluded that the highest relative variable is the Credit Performance per grade point average. To further understand the performance groups a subjective selection of six clusters was set for this study and two distinct clusters were identified (one cluster identified the foreign students that performed better than their counterparts and the other cluster identified the local students).

When analyzing e-learning data, the existence of statistical outliers (students with unusual behavior) result in the creation of classifiers with less accuracy. García-Saiz and Zorrilla [2012] proposed a method to eliminate those outliers as a previous step to the classifier construction. The authors start by carrying out a correlation study and then remove dispensable attributes. A two-class classifier is built and determine which are the incorrectly classified instances of both classes. With this information, a prototype of each class is calculated as well as the euclidean distance of each instance to its prototype. Then, the average distance in each class is obtained and the one which has a larger value is chosen - call it K. Next, the incorrectly classified instances of the chosen class are selected and a k-means is applied to separate these instances in two groups. The centroid in both clusters is calculated and the instances belonging to the cluster whose centroid has a larger euclidean distance to the instances of the K class are removed from the training set. This action eliminates the instances that have a more irregular distribution with respect to the distribution of the instances set of K class.

3.5 Open Issues

Given the presented literature, we aim for presenting a critic look over the existing solutions to recommendations in education. Therefore, we will criticize works according the *items recommended* dimen-

sion of the evaluation framework we defined in section 3.2.

Among recommendations of learning materials, some of the works try to recommend materials that the students are likely to prefer [Tsai et al., 2007][Ge et al., 2006]. For instance, Ge et al. [2006] recommended items to a target student according the rating the item had among the students that are most similar to to the target student. In this approach the similarity of users is only based on demographic or in the major degree chosen. However, students with the same demographic profile or with the same major degree can have several different interests. Therefore, a more detailed profiling process is needed. Hsu [2008] also tried to capture the students' preferences, dividing them into clusters and retrieving the most frequent learning sequences among each cluster. Once again, learning needs of students are not being taken into account. Soonthornphisaj et al. [2006] took into account students' capabilities by recommending the items with best ratings among the students with the same level of expertise of the target student. Still, this approach has a contra, since it requires that students perform quizzes to determine its level of expertise. A better scenario would be one where their expertise could be captured automaticaly, as it happens in the proposal of Krištofič [2005]. The author proposes to automatically build students' requirements as they interact with an adaptive hypermedia system. Yet, in this case the recommendations given will also be extremely related to the concepts students are viewing. Hence, it will not bring novelty to students' knowledge and will only work if their interactions are related to their needs. Lu [2004] also modelled students requirements automatically, but according to their learning history so as to find the most similar neighbors and recommend their most frequent learning patterns. This proposal as well as the one done by Shen and Shen [2004] takes an interesting step on recommending items that will help students to reach their learning goals. Both proposals look into students' current skills, and try to recommend learning materials that will fill some competencies gaps in order to reach some learning goal. However, the latter presents a recurring problem, that requires a lot of work and discussion - the definition of a tree of learning concepts. Another weakness shown in some approaches [Su et al., 2006][Gutierrez et al., 2012], is that the recommendations are done according the best students' ratings. This is problematic, as learning strategies inside classrooms differ according students' skills and characteristics, so the same care must exist when recommending learning materials. Some other general problems are the lack of experimentation with several types of learning materials, as this could influence students' learning, and with different types of levels of education, as all approaches are related to higher education.

Regarding recommendation in which courses to take, one weakness is that some works present is the lack of capability to present recommendations at the time of the request. Some approaches miss an automated process when modelling students [Wang et al., 2009], while in others the recommendation process takes a long time to produce results [Unelsrød, 2011]. This aspect is considerable negative as students tend to leave these choices to the last minute. To solve the former problem some approaches sit on top of education platforms having quick access to student information. However, given that they only have access to limited information, these approaches tend to create simple student profiles, only recording which courses the students took. Therefore, student profiles only model the ”potential” interests of a student, taken from the contents of courses the student enrolled in. This leads to another

question: why do the existing recommendation systems give so much importance to courses contents? In fact, this is not a problem - they indeed have their factor of importance. The big problem is in the lack of attention given to which will the be the student performance in a certain course, given its capabilities and the course difficulty. In fact, one or other approach try to give recommendations based on the prediction of student performance [Vialardi et al., 2010][Vialardi et al., 2009]. However, they only give recommendations based on the likelihood of a student to be approved or to fail on a course. As some students are not just worried in passing courses, a better scenario would be of giving recommendations based on a estimation of the student final mark. This way, they could decide which courses to follow given the prediction of the mark they would obtain. Besides this, all these approaches are not prepared to deal with cases in where not all the necessary data to build a student profile is available. Other aspect these approaches have in common is that their authors are looking to this problem as a *classification problem*. Though, this problem is more linked to recommendation than classification. Hence, it is needed to reformulate the view taken when approaching this problem. For instance, a greater care of how to present recommendations should be taken. No work had particular consideration for how many recommendations it should present and how they should be presented. However, this is of particular importance. For example, if we know that there is a certain course that has great value for a student, then we should present the recommendation in such a way that if the student does not want to follow all courses, that then it will at least follow that specific course. From the couple of approaches that used recommendation techniques, one took into account courses contents [Unelsrød, 2011] while other used students' ratings over courses difficulty and workload [Farzan and Brusilovsky, 2006]. Both approaches do not take into consideration students' performance while the former approach uses ratings on overly subjective metrics. Furthermore, as both approaches are based on *collaborative filtering* techniques they will only get better with the more ratings the students give. Thus, it is probable that a student only receives good recommendations in the end of the graduation However, we realize that the transposition of the usual recommendations solutions to course recommendation is hard to do. In this context, the object of recommendation has different properties: the user is not constantly consuming recommendations and the impact of a bad recommendation is much heavier than on other domains - for instance on movies recommendations or commerce recommendations.

In the recommendation of feedback we can see that most of the works are directed to teachers, so giving feedback to students is an area that still can be explored. Systems that monitorize the students' actions when solving a problem and that give counseling to students are little explored. Only Chu and Hwang [2006] tried to suggest corrective actions to students according the results on a test. The feedback given by their system indicated which concepts the students dominated, as well the ones that required more theoretical study or more practice. However, the definition of trees of concepts still requires a lot of work as well the generation of adequate test sheets that help on asserting students' skills. A system that could be refined into giving feedback to students is the one proposed by M-Helene et al. [2006]. The authors produced a system that looks into students actions solving a problem and tries to predict their intentions and why they are making errors, while providing that information to teachers. It would be interesting to somehow present that information to students as they were solving problems,

in order to help them in the problem resolution. All the other systems looked into giving feedback to teachers, mostly by presenting association rules that identified a problem, and sometimes a suggested corrective action [Vialardi et al., 2008][García et al., 2009][Romero et al., 2005]. However, these systems were very linked to particular study areas, still existing the need to experiment with different academic environments and other types of teachers. Another weakness is that in some systems the corrective measures must be defined by experts and teachers [García et al., 2009]. Hence, this is subjected to personal views of experts and teachers and it would be better if this could be done automatically by the system. Other systems such as the one proposed by Romero et al. [2005], only act as a tool to mine students information, but the decision of which rules are interesting is left to the teachers. A possible improvement would be for the system predicting which rules were most interesting. As in most works, it would be desirable the use of more advanced data mining techniques, that should be able to detect better potential problems from a little set of symptoms.

In group recommendation there are just a couple of works, so we should note that there are still several doors to open. As Crespo and Antunes [2013] tried to predict teamwork results and Rovira-Asenjo et al. [2013] tried to predict future conflict, other works could be done in predicting which teams are more likely to not have conflicts, and which groups are more adequate according the type of work and the topics approached in each work. Efforts on the estimation of which group members are improving our lowering the team productivity would be of great value. Besides that, it would be interesting to recommend which positions (with different work activities) different members should have in a group. Research on alternative ways to represent teams is needed, as there only few representations have been experimented.

Chapter 4

Masters Courses Recommendation

Taking into account the open issues mentioned in section 3.5, we propose a system that is able to recommend the set of master courses that maximizes the likelihood of students having a good academic performance, given their bachelor results. Our proposal tries to explore historical students' marks using *singular value decomposition* on top of a *collaborative filtering* approach, so as to capture the hidden latent features that are able to explain the marks obtained by students. We chose to explore SVD since we believe that it can be used to discover features that students' results disclose, and that are powerful enough to be used in the prediction of masters courses marks. We also aim to contest the problem of having incomplete students' profiles at the time of recommendation using *ASAP classifiers*.

In this chapter, we will put into context the SVD technique, and how it is commonly used in recommendation. Then, we will present our proposals to recommend master courses only using students' results, and in how to deal with incomplete students' profiles prior the production of any recommendation. We will finish by presenting our view on how these proposals should be evaluated.

4.1 Singular Value Decomposition in Recommending Items

In this section we will start by giving a theoretical context on the appliance of SVD in recommendation problems. We will then describe each step of our methodology, in parallel with a description of a small example that, hopefully, allows for a clear understanding of the whole process.

4.1.1 Singular Value Decomposition

Mathematical Definition

Singular Value Decomposition is a matrix factorization technique that has its roots in linear algebra. This technique factors a matrix R into three matrices as follows:

$$R = U\Sigma V' \tag{4.1}$$

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

diagonal $r \times r$ matrix, where r represents the rank of the matrix R (see figure 4.1), and whose values σ_i are the so-called *singular values* of this decomposition. These values are stored in decreasing order of their magnitude. Each of the entries σ_i in Σ represents a hidden feature and the corresponding value stands for the weight that the feature has to the variance of the values on matrix R. In fact, if we sum all the values stored on this matrix we can get the total variance on matrix R.

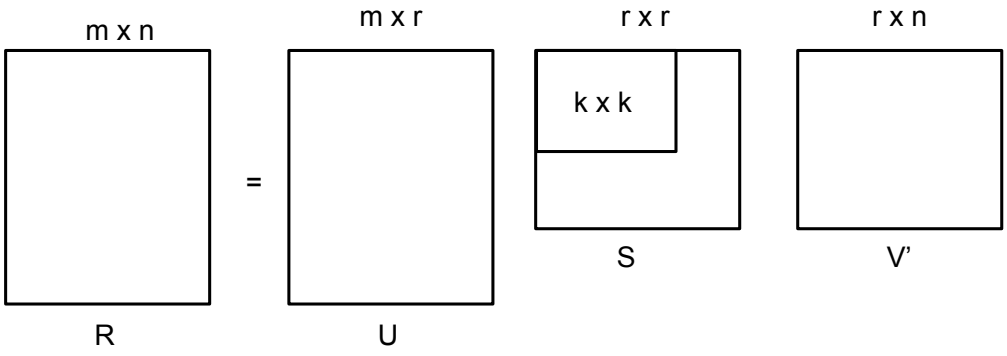


Figure 4.1: SVD Decomposition of Matrix R

Hence, in the pure form of the SVD, U is $m \times \hat{k}$, Σ is $k \times \hat{k}$, and V is $n \times \hat{k}$, where R is $m \times n$ and has rank $\hat{k}$. However, this simple decomposition form does not bring any benefit to the production of recommendations. Still, 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. (see figure 4.2). In fact, this technique is widely used for being a simple way to find a good approximation of a matrix R. Furthermore, it is possible to obtain the best k-rank approximation, R_k, to the matrix R, in such a way that the *Frobenius norm* of $R - R_k$ is minimized. The *Frobenius norm* ($||R - R_k||F$) is simply defined as the sum of squares of elements in $R - R_k$ [Deerwester et al., 1990].

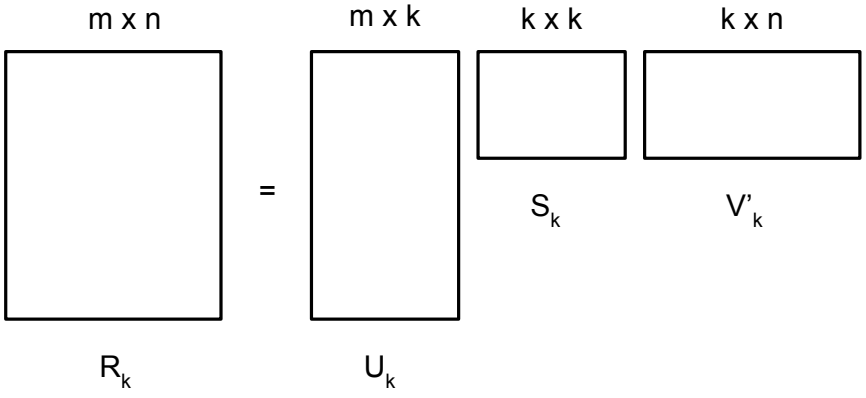


Figure 4.2: k-rank approximation of matrix R

Application of SVD to Recommendation

When using this technique on recommendation systems, the $m \times n$ matrix R corresponds to the users-items matrix, the *ratings matrix*, where m is the number of users and n is the number of items. The value of the cell R_{ij} corresponds to the rating that the i^{th} user gave to the j^{th} item. Recommendation systems can benefit a lot by prooceding to the truncation that we explained above, which results in an approximation of the original ratings matrix. In this context, the truncation of the matrix encase some catching benefits: on one hand, as we are just keeping k features (with $k < r$, and where r is the original rank of the ratings matrix), then we are choosing to represent both items and users in a k-dimensional space, thus decreasing the original dimensionality of the vectors spaces. This is good both in terms of memory usage, as we need to use less space, whether in terms of complexity of the algorithms, as the algorithms can benefit of having to learn from less data. A second advantage of dropping the *singular values* with less influence to the variance of the matrix is that we are consequently eliminating small perturbances that manifest themselves as noise behind the ratings distribution. This noise is mostly created by external factors that somehow affect the sheer preference of the user. Therefore, the truncation enables the production of better quality recommendations [Sarwar et al., 2000].

But how should we interpret the SVD expression in a recommendation context? As we said, the decomposition results in $R \approx U\Sigma V'$, which 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 relevance to the topic. In fact, this corresponds to the weighted dot product of user i features vector U_i and the item j features relevance vector V_j:

$$p_{i,j} = \sum_f U_{i,f}\sigma_f {V'}_{f,j} \tag{4.2}$$

Following this idea, it is possible to calculate a user's i preference for all items by multiplying his features vector with the weighted item matrix:

$$p_u = U_i \Sigma V' \tag{4.3}$$

Sometimes it is beneficial to normalize ratings by subtracting baseline predictors (for example, the user or item averages) prior to computing the model. When doing this, however, it is necessary to use the subtracted baseline when computing the predictions:

$$p_{i,j} = b_{i,j} + \sum_f U_{i,f}\sigma_f {V'}_{f,j} \tag{4.4}$$

where $b_{i,j}$ is the normalization factor subtracted from the ratings before model computation. Since the normalization is encoded into the decomposed matrix, then it must be reversed at the prediction stage.

In order to use SVD, it is necessary to first compute the matrix factorization. There are a variety of algorithms for computing singular value decompositions, including Lanczos' algorithm, the generalized Hebbian algorithm, and expectation maximization [Ekstrand et al., 2011]. Nevertheless, SVD is not known for dealing well with sparse matrices, since it can only be well defined when the matrix is complete (when it has no missing values). As in usual recommendation problems there exists a lot of sparsity on the rating matrix, some care must be taken in computing the *matrix factorization*. Initially, some proposals tried to perform the *imputation* of the matrix, filling in the unknown values. Some approaches filled the missing values with either the user or the item average [Sarwar et al., 2000]. However, these approaches are higly prone to overfitting.

Gradient Descent approach

In order to escape from matrix *imputation*, some approaches were developed so as to compute an estimate of SVD by only using the known ratings of the matrix. One of the proposals consisted in using the *least squares method* to learn a regression for each user. Though, the biggest breakthrough happened during the Netflix challenge when Simon Funk proposed to use a *stochastic gradient descent* method in order to compute the best k-rank approximation, only using the known ratings of the matrix [Funk, 2006]. According to his approach, which is still considered as under the *matrix factorization* category, the user-item matrix R can be separated into two distinct matrices that are able to give an approximation of the original matrix R when multiplied, thus:

$$R_k \approx UV' \tag{4.5}$$

where R_k is a $m \times n$ matrix, U is $m \times k$ matrix, and V' is a $k \times n$ matrix and k is the number of features to consider. R consist of integer values, whereas U and V consist of real values. The idea behind Funk's approach, is that we should find matrices U and V that minimize the error caused by the differences between the known original ratings and the ratings resulting from matrix multiplication. This allows us to get ratings for every user on every item, which eliminates any sparsity on the original ratings matrix. We will refer to ratings in the matrix with R_{ui}. The error is represented by

$$SE = \sum_{u \in \text{Users}, i \in \text{Items}} (R_{ui} - P_{ui})^2 \tag{4.6}$$

where $P_{ui} = \sum_{k=1}^{K} (U_{uk} V'_{ki})$, and K is the number of features found. To obtain the minimum error, we then are able to follow the same idea as the one used on training *neural networks*. To find the optimal values on the matrix, the *gradient descent* algorithm tries to minimize the error on a predefined number of iterations. In every iteration, the algorithm trains each topic f in turn using all the known user-item ratings and the following update rules (λ is the *learning rate* and typically 0.001):

$$\Delta U_{u,f} = \lambda(R_{u,i} - P_{u,i})V'_{f,i} \tag{4.7}$$

$$\Delta V'_{f,i} = \lambda(R_{u,i} - P_{u,i})U_{u,f} \tag{4.8}$$

This *learning rate* acts as a percentage that influences the speed and quality of values learning. In this problem it sets how much to change the singular vectors away from errors. In fact this parameter determines how fast or slow we will move towards the optimal weights to minimize the error. If λ is very large we may skip the optimal solution. If it is too small we will need too many iterations to converge to the best values.

The *gradient descent* algorithm to compute the SVD also enables regularization to prevent overfitting of the resulting model. Hence, the resulting model will not be a true SVD of the rating matrix, according to the mathematical definition, as the component matrices are no longer orthogonal. This is essentially equivalent to penalizing the magnitude of the features, which tries to cut down on overfitting, ultimately allowing us to use more features. Though, the resulting model tends to be more accurate at predicting unseen preferences than the unregularized SVD. This regularization is achieved with the addition of a term to the update rules in Equations (4.7) and (4.8), that is γ - the *regularization factor* - typically between 0.1 and 0.2. The equations that take into account the regularization are presented below:

$$\Delta U_{u,f} = \lambda((R_{u,i} - P_{u,i})V'_{f,i} - \gamma U_{u,f}) \tag{4.9}$$

$$\Delta V'_{f,i} = \lambda((R_{u,i} - P_{u,i})U_{u,f} - \gamma V'_{f,i}) \tag{4.10}$$

In summary, the final solution to 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. One issue that we should take into consideration is that all feature 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.

4.1.2 Recommendation Methodoloy

Our idea is to apply a *user-based CF* approach to select the most similar historical students (neighbors) to a target student (considering their bachelor's achievements) and then use those neighbors' bachelor's and master's 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 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. 4.3. We will now describe the methodology used to build the system and produce recommendations.

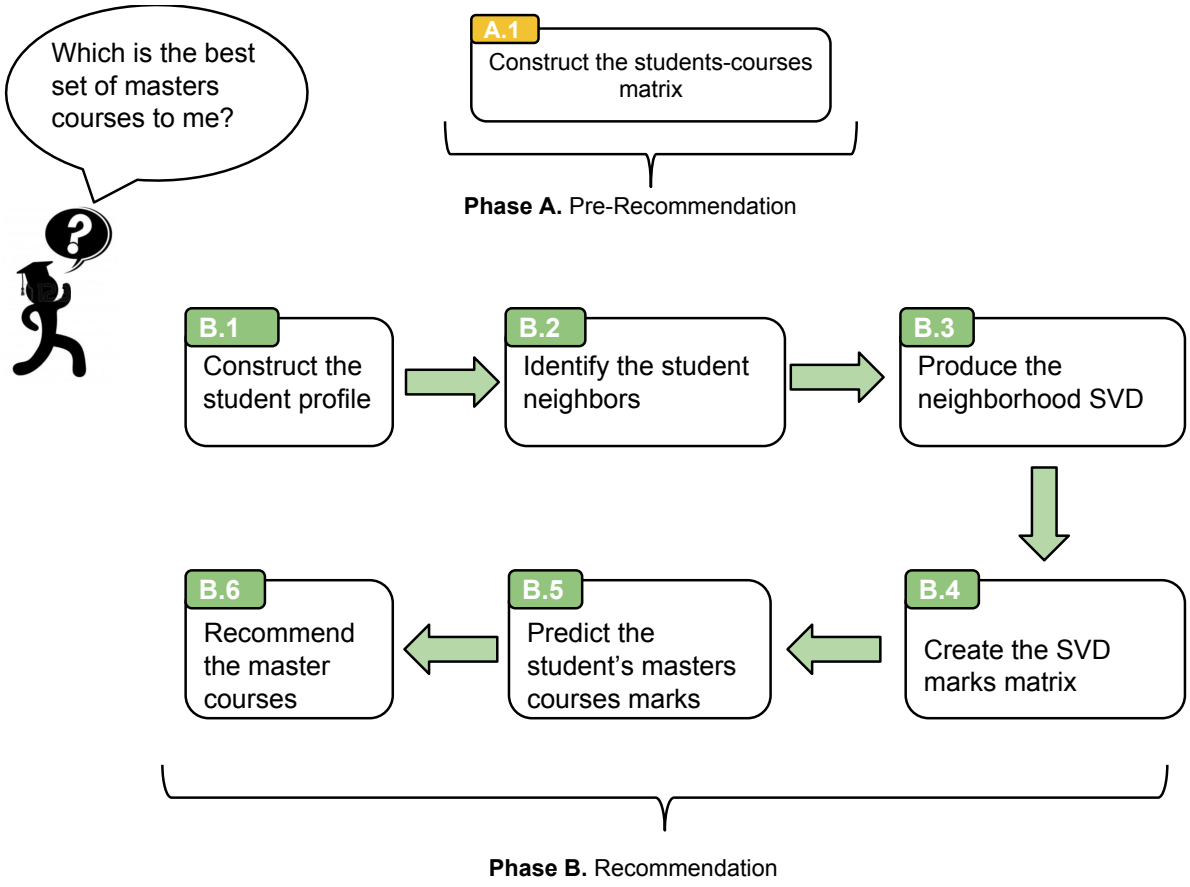


Figure 4.3: Main steps of our recommendation process

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*$>$. We will now describe how to structure this data:

1. Construct the students-courses matrix As seen in section 2.2, 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. In usual representations, users are placed in rows and items in columns, and each cell R_{ij} in the matrix corresponds to the rating that the i^{th} user attributes to the j^{th} item.

As a first step to map our problem to the appliance of SVD on top of a *user-based* approach, we must parse our historical students' marks records into a matrix R that holds our knowledge over students' capabilities on each course taken. 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 the 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 the value of each entry R_{ij} can be calculated as follows:

$$R_{ij} = \begin{cases} \text{student's } i \text{ mark on course } j, & \text{if the student enrolled in course } j \\ 0, & \text{if the student did not enroll in course } j \end{cases} \tag{4.11}$$

Easily, R_{ij} will be filled with the mark obtained by the i^{th} student on the j^{th} course. In the cases that there are missing values, that is, in the courses where the student did not enroll, R_{ij} will hold 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 each step with a small example, so that we can better understand how the process unwinds itself. In figure 4.4, 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's courses and 4 master's courses. In this small example the marks scale for each course goes from 1 to 5. For instance, in the presented matrix, we can see that Matt had 3 on the AI course during bachelor and then had 5 on the DSS course on his masters. On the other hand, it

Historical Students' Results

< Matt, NA, 2 >
< Matt, AI, 3 >
< Matt DS, 4 >
< Matt, SE, 4 >
< Matt, OOP, 4 >
< Matt, DSS, 5 >
< Matt, DP, 5 >

< John, NA, 2 >
< John, AI, 4 >
< John DS, 3 >
< John, SE, 3 >
< John, OOP, 3 >
< John, DSS, 2 >
< John, MC, 3 >

< Susan, NA, 4 >
< Susan, AI, 3 >
< Susan DS, 3 >
< Susan, SE, 3 >
< Susan, OOP, 3 >
< Susan, DSS, 4 >
< Susan, MC, 4 >
< Susan, IM, 5 >

< Eric, NA, 2 >
< Eric, AI, 3 >
< Eric DS, 4 >
< Eric, SE, 3 >
< Eric, OOP, 3 >
< Eric, DP, 3 >
< Eric, DSS, 5 >
< Eric, MC, 4 >

< Victoria, NA, 2 >
< Victoria, AI, 4 >
< Victoria DS, 3 >
< Victoria, SE, 3 >
< Victoria, OOP, 3 >
< Victoria, DSS, 3 >
< Victoria, IM, 4 >

< Michael, NA, 2 >
< Michael, AI, 3 >
< Michael DS, 5 >
< Michael, SE, 4 >
< Michael, OOP, 4 >
< Michael, DSS, 4 >
< Michael, IM, 4 >

	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 4.4: Example of students-courses marks matrix R

can also be seen that Matt has not enrolled in both MC and IM courses, as his marks on both courses are 0.

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 of 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 mark, 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 respective course, or 0 if the student has not been enrolled on that same course.

Now, we think that it is the ideal chance to present Damian, an example of a student that requests a recommendation from our system. He is a student that has just finished his bachelor, so we have full data about his bachelor marks to create his profile according the rules explained above. One can see his profile representation in figure 4.5.

	Bachelor courses				
	NA	AI	DS	SE	OOP
Damian	3	2	4	4	3

Figure 4.5: Example of a profile representation of a student being recommended.

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 data. This idea follows our belief that the more two students have similar marks on bachelor, the more will their master's marks be similar. Following this idea, we can then later 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's courses marks, i.e, the values of the first n positions of the vector of each student, with n being the number of bachelor's courses. The distance metric used to calculate the similarity value between two students will be the *Pearson Correlation* (see equation 4.12). 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 as follows:

$$w_{a,u} = \frac{\sum_{i \in I}(g_{a,i} - \bar{g}_a)(g_{u,i} - \bar{g}_u)}{\sqrt{\sum_{i \in I}(g_{a,i} - \bar{g}_a)^2}\sqrt{\sum_{i \in I}(g_{u,i} - \bar{g}_u)^2}} \tag{4.12}$$

where I is the set of bachelor's 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 mark obtained by student u on his bachelor. 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 Damian, considering their bachelor marks, and then select the three most similar neighbors. In fact, we can look up to these results on figure 4.6 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 4.6: 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 (see section 4.1.1) using the bachelor and master marks for of 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 - the gradient descent 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, that holds the courses features weights - and that corresponds to the gradient descent matrix V - 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 4.7 one can see the resulting students' features space and courses features space. These resultant 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.

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 4.7: Resultant matrices of the application of SVD in the neighbors' space

The relation that each student and each course hold with each feature, represented through their feature vectors in S and C respectively, can be well understood through figure 4.8. This figure shows a 2-dimensional feature space, and how are the students, bachelor's and masters' courses distributed along this dimensional space. With little observation, we should take some interesting conclusions about the power of these students' and courses features vectors. For instance, one should note that Matt and Michael are very close in this dimensional space. In fact, by inspection of figure 4.4, one can see that Matt and Michael have a similar mark average - 3.4 and 3.6 respectively. Concerning courses it is possible to note an apparent relation between the SE bachelor's course results and the IM master's course results, as they are placed around the same area. One could say that it is related just with the course average mark, but that it is not true as both the SE and IM courses have different average marks, 3.33 and 4.33 respectively, while AS has also an average mark of 3.33 and is really distant from SE. This way, one can see, that with a small number of features we were able to distribute both students and courses according to their contribution on each of the discovered hidden features.

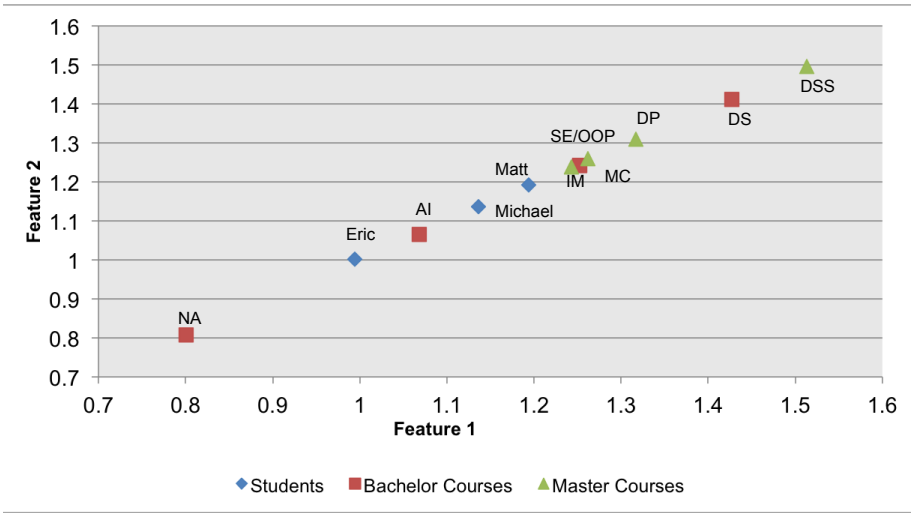


Figure 4.8: Neighbor students' and courses distribution on the 2-dimensional feature space.

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 produce the computation of a M_{ij} value on this matrix is as follows:

$$M_{i,j} = U_i \cdot V_j \tag{4.13}$$

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 neighbors selected on step B.2 .

We can see a specific scenario of the construction of this SVD matrix following our example. For instance, if we want to know which is Matt's estimated mark on the DP course we have to compute the dot-product between S_{Matt} and V_{DP} (see figure 4.7). Hence, the calculated mark value would be $1.465 \times 1.317 + 1.444 \times 1.309 \approx 3,82$. One can see the totality of these inital prediction marks matrix calculated for our example in figure 4.9.

	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 4.9: Matrix with marks obtained by SVD prediction

5. Predict the student's masters 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:

$$p_{u,i} = \bar{M}_u + \frac{\sum_{u' \in K} PearsonCorrelation(u,u')(M_{u',i} - \bar{M}_{u'})}{\sum_{u' \in K} |PearsonCorrelation(u,u')|} \tag{4.14}$$

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. Note that, in this step we can decide to only use the x most similar neighbors to the calculation, where x is less than or equal the size of

the selected *neighborhood*. This can be an interesting approach to try to reduce the noise on this final stage.

To understand the logic of this prediction with more detail one should look to figure 4.10. In this figure we show the needed computation to predict Damian's mark on the DSS course. We have highlighted with different colours how each of the neighbors contributes to the prediction calculation. For instance, we use Matt's predicted mark on DSS, 4.379, and subtract 3.4 from it (which is his bachelor average) to take into account any possible bias in the marks obtained. We then multiply the computed value by his similarity weight. We do this for each neighbor, sum all the computed values, and in the end, we divide this sum by the sum of the similarities, so as to distribute the influence weight of each neighbor.

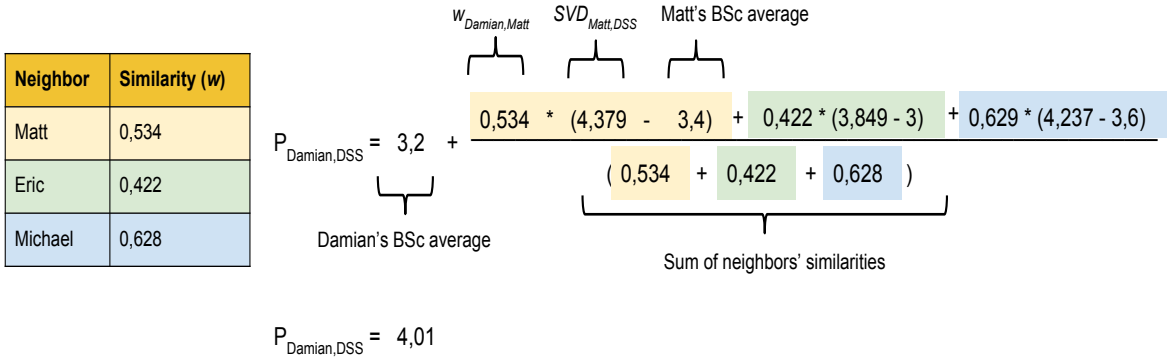


Figure 4.10: Example on DSS course mark prediction.

6. Recommend masters 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.

Regarding our example, we are now ready to present our proposal to Damian. One can see the predicted marks for each of the master courses on figure 4.11. According to our predictions, the courses that we would recommend to Damian would be DSS and DP as the two are ones with best predicted marks among the four possible - with marks of 4 and 3 respectively. In the end, we opt to present our recommendations and our estimated prediction as an integer value, as the marks are also integer numbers.

4.2 Recommending Courses with Unobserved Data

Now that we presented our proposal in how to recommend courses with value to students using *Singular Value Decomposition* and *User-Based Collaborative Filtering* we want to step onto another problem:

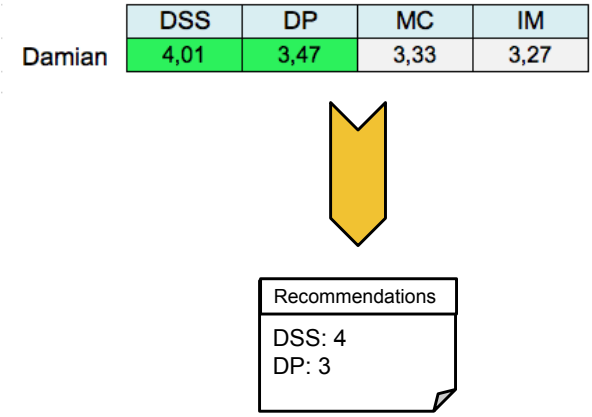


Figure 4.11: Example on final predictions and courses recommended

recommending courses when the data over the student being recommended is not fully observable. This problem is related to some problematic cases that appear when we think of recommending courses to students. One of them, is when the student asks for a recommendation despite not having yet finished all of his bachelor courses. In fact, this simple scenario hurts a lot of the current approaches as they usually require a complete student profile to produce recommendations. Other possible scenario is when a student coming from other university wants a recommendation. In this case, the student may only have a small subset of courses in common with any of the bachelor program of the university he is trying to get recommendations to. This might also happen when a student from a bachelor program in the university tries to be recommended with master courses of another program. For example, a student that has a bachelor on Mecanic Engineering tries to be recommended with courses of Computer Science Masters. In this case, the recommendation process could only use the bachelor courses that Mecanic Engineering and Computer Science have in common. In order to approach this problem, we propose a pre-processing step where we aim to predict the target student's missing bachelor marks using a technique called *as-soon-as-possible classifiers* [Antunes, 2010]. We will now briefly describe this technique and what it tries to accomplish, and then explain how we intend to use it to alleviate this problem.

4.2.1 As-Soon-As-Possible Classifiers

As-Soon-As-Possible classifiers [Antunes, 2010] take a different look to what is usual in classification. Escaping the tendency to study in how we can handle data to achieve an higher accuracy on classification - which is the main problem studied in this area - this technique approaches the problem of classifying instances that are partially observable, when classifiers may be trained with fully observable instances.

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. This way, this approach assumes that the order between attributes matters, and that the classifier for the instances with m attributes has to be able to classify instances described by a fewer number of attributes. Neverthless, nothing is claimed on how to train the classifiers. In fact, the use of historical records in the training set is compatible to the fact of the same ones being fully observable. This means that traditional classification methods can be used without extra processes.

Any classifier that works under this formulation can then be considered as an *asap* and it can be trained according two different strategies:

- **Pessimistic:** This first one trains the classifier only using the observable attributes. In fact, this approaches the problem using traditional classification, by reducing each training instance from its original m-dimensional space to an n-dimensional space, with $n < m$. Though, this approach does not use the totality of the data available at classification time, wasting the historical values of part of the attributes in the training set. This may result in less accurate classsifiers.

- **Optimistic:** This second one uses the entire set of attributes. This strategy needs to train classifiers from m-dimensional instances that can be applied to n-dimensional ones. In this case we can also take two approaches: one is to convert the learnt classifiers, a function from A_m to C, into a function from A_n to C, while the other is to convert the n-dimensional instances into m-dimensional ones. In this latter option, it tries to enrich non-totally observable instances, which can be achieved with any method capable of estimating unobservable attributes from observable ones.

Specifically, the *optimistic* approach reveals itself very interesting as the classification will be divided into two different phases: first, and for each unobservable attribute x_i, it will use all the observable data and estimated attributes x_j (with $j < 1$) to estimate its value; secondly, when all missing attributes values are already estimated it passes all the data, the observable and estimated values, to a regular classifier to produce the final classification. Hence, it may be seen as a progressive method to classify an incomplete instance.

So as to better understand the idea behind it, one should note that this technique was used to anticipate student's failure using the first student's results on a course. The approach was to use *ASAP classifiers* to estimate the marks of all of the upcoming students' evaluations on the course and then pass the known and estimated results to a classifier that stated if the student was going to pass or fail the course. The *ASAP classifiers* would then try to estimate the missing evaluation results using the results the student had until that point in time.

4.2.2 Completing Student's Profiles Before Recommendations

In this section we will describe how we plan to apply *ASAP classifiers* to our recommendation process, so as to enable higher quality predictions when the student's data is incomplete.

The way we will use this technique digresses from its original goal - of using it to to determine a final class for a partially-observable instance. This is, to estimate missing values to determine if whether a student would pass or fail on a given course. Instead, we will focus on using just the estimation phase.

This way, our goal is 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 observable. 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).

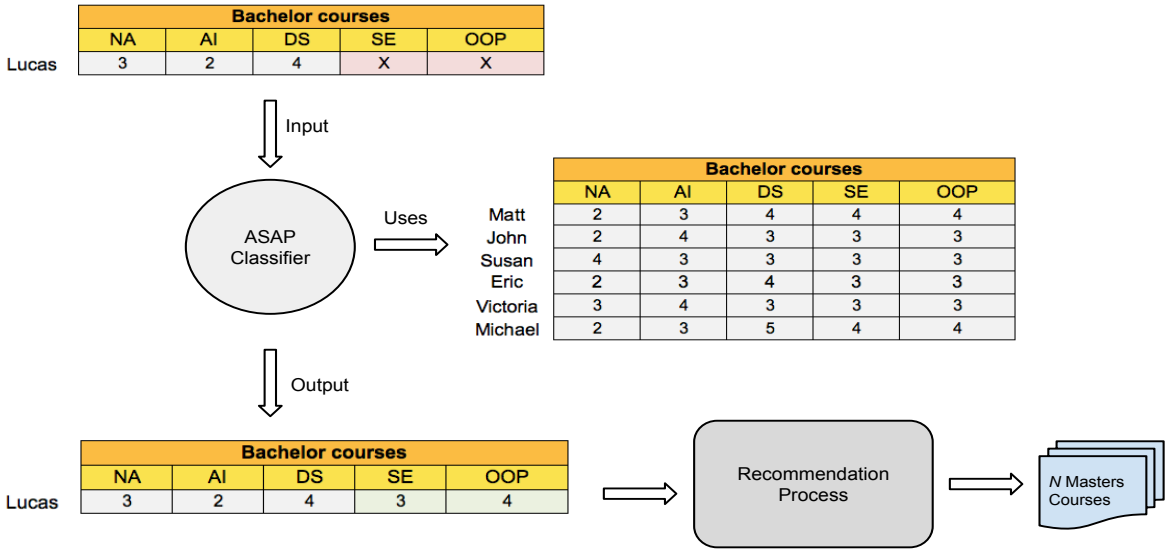


Figure 4.12: Application of ASAP classifiers as pre-processing step

For example, in figure 4.12 one can see the scenario in the same universe that we presented in the last section. Here, we have Lucas, an example of a student that has not yet finished the two last bachelor courses, SE and OOP, but thas has the chance to already attend some master courses. He requests a recommendation from our system. With our recommendation system presented in section 4.1, we could produce recommendations to Lucas using only the known bachelor marks. However, the identification of the neighbors would suffer from the lack of complete profiles, when comparing to other users. This is due to the fact that it is expected that the more information available the more accurate is the calculation of the similarity between Lucas and all the historical students. It is in the moment of the request, as we

identify that the student has missing values on his bachelor attributes that we put into use the *ASAP classifiers*. In this example, the *ASAP classifiers* would have been trained on the historical bachelor results, having a classifier for each one of the bachelor courses. Note that the classifier for each course value will use the bachelor marks posterior to that course. More precisely, in the first step the *ASAP classifier* would use Lucas' marks on NA, AI and DS to estimate SE's mark. As we can see on figure 4.12, his estimated mark on SE is 3. Then, this technique would use the recently estimated mark, along with the other three already used to estimate the mark on OOP. In this case, the predicted mark was 4. At this stage, the system would have a complete profile regarding Lucas' bachelor. Therefore, the new completed profile could be passed into our regular recommendation process, accurately identifying the neighbors, creating svd-dimensional space and producing the master courses predictions that sustain the selection of the courses to recommend.

Chapter 5

Case Study

In this chapter we apply and evaluate our proposal to recommend master's courses in the context of a case study. In our view, our recommendation problem can be decomposed into two subproblems: the marks prediction and the production of recommendations. We will give more focus on evaluating master's marks prediction accuracy. This way, if we achieve a low error on the predictions we can recommend courses with certainty. However, despite the focus that we will give on evaluating prediction accuracy, we will also explore other aspects to evaluate our approaches. To be able to evaluate and corroborate our proposal we have colllected and analysed real students' results that we have used as a case study.

The used dataset was drawn 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 both a 3 years bachelor and a 2 years masters. In this dataset we are able to reach students' results on 22 mandatory courses, the bachelor's courses, and around optional 60 master's courses, making it a good source from where we can test our recommendation system. In this dataset, the mark scale for all courses goes from 1 to 20, where ten is the minimum mark that a student must achieve to be approved on any course.

As our dataset includes academic results of a span of twenty years, it holds changes on the bachelor's and master's programs that happened along this time. To take this into account, we did some pre-processing in courses normalization, merging courses that we considered equivalent (i.e, different courses that approach the same contents) into only one course. Another aspect that we had to take into account was that in this dataset we had no information on the mark achieved by a student when he failed the course. Our solution was to give a mark value of 9 to each course result in this condition. After this inital pre-processing step our dataset contained approximately 25000 evaluation results from 550 students. Each student record contains the marks for all of the 22 bachelor's courses and for, at least, 15 master's courses.

To perform our experiments we extended the Java framework *Recommender101* [Jannach et al., 2013]. This framework enables us to carry out offline experiments for recommendation systems. It provides several metrics and a set of recommendation techniques to use as baselines for our results.

5.1 Evaluation

Current research on recommendation systems uses several types of measures for evaluating the success of recommendations. However, the correct way to evaluate each system depends heavily on the system goals and domain. As we mentioned in section 4.1, our recommendation process can be decomposed into two different problems: the marks prediction and the decision over which courses to recommend given the computed predictions. Our belief is that the overall quality of the recommendations, independently of the method used to produce them, depends to an extent on the quality of our predictions. This is, the lower the error on the master's courses marks predictions the better will be the quality of the produced recommendations. Hence, this will be the main focus of our experiments.

To achieve it, we had to use one of the commonly applied statistical prediction 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 the *Root Mean Squared Error* (RMSE) between the predicted and the real values. As the research experience shows that both metrics typically track each other [Sarwar et al., 2000] we chose to use only one, in this case the MAE, because it is the most commonly used and the easiest to interpret as it is presented in the same scale as of the ratings. In our domain, the MAE corresponds to the average of the absolute difference between the course mark prediction and the real mark obtained for all the master's courses in the test set. This is, if we have n master's courses the MAE is computed as follows:

$$\frac{1}{n} \sum_{u,i} |p_{u,i} - g_{u,i}| \tag{5.1}$$

where $p_{u,i}$ is the predicted mark of student u on the master course i and $g_{u,i}$ is the actual mark obtained by student u on course i.

Another aspect that must be measured is the system ability to provide distinct recommendations, as we do not want the system to recommend only a small subset of the possible master's courses. We can verify this by measuring the percentage of times that each course is recommended to students, and realizing if there are any courses that are recommended much more times than others, and if some of them are never recommended.

Another evaluation measure to be taken into account is the students' recommendation acceptance: how many courses of our recommendations do the students follow. It is important that our system has a good acceptance by the students, since a recommendation system without users has no utility. The acceptance evaluation measure is defined as follows:

$$\text{acceptance} = \frac{\text{nr. of courses recommended that the student attended}}{\text{nr. of courses that the student attended}} \tag{5.2}$$

Finally, it would be interesting to measure the average mark that students obtain when following our recommendations. This can be computed with a simple average between the sum of marks achieved in followed recommendations and the total number of followed recommendations. However, we know that the results of this metric may be biased as we cannot control the total number of courses that students

follow from the set of courses that we recommend.

5.2 Master's Marks Prediction Analysis

We will start by taking a look on which is the accuracy that our master's mark predictions yield. To analyse it, we will use the MAE (Mean Absolute Error) to predict the error associated to our predictions. As we referred in section 5.1, this metric allows us to easily interpret the result according to our domain knowledge and verify if they are reasonable, as its value will be in the same scale as the marks. We also want to refer that we divided the dataset into a training and testing set, in a $70\% - 30\%$ proportion, respectively.

5.2.1 Effect of the size of the neighborhood

To start our evaluations we set up some experiments that show how our approach behaves with different parameter configurations. We first attempted to verify what was the effect of the chosen number of neighbors on the *user-based collaborative filtering* step of our approach. For this experiment we set the number of features to 20 and the number of training steps of the gradient descent to 25. We then ran the recommendation process and predicted the master's courses marks for the test set using different number of neighbors. In this experiment we used the same number of neighbors to form the *neighborhood* and to predict the marks. We should remember by now that we could predict the marks using only a subset of the *neighborhood* as explained in the step B.5 of section 4.1.2. We would like to add that, for each chosen number of neighbors, we ran the experiment five times and recorded the average of the achieved MAE, due to the random noise present in the training of the *gradient descent* algorithm.

Looking into to the results of this experiment in figure 5.1, it is possible to 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 analyse, 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 100 neighbors, where we reach our optimal error. Finally, the error starts to slowly increase as we reach the 300 neighbors (where is using almost all the training set as the neighborhood). Despite the fact that the size of our dataset limits us to not having the chance of using more than 300 neighbors, we believe that this tendency to increase the error with the number of neighbors would uphold, as it is the usual behavior seen in *CF* approaches [Sarwar et al., 2001][Herlocker et al., 2000]. This is due to the fact that when we use to many neighbors to generate the predictions, the amount of data is so much that it may be bringing unwanted noise from the more distant neighbors, losing overall quality in the predictions.

With these facts summed up, we may state that it is possible to generate accurate predictions using

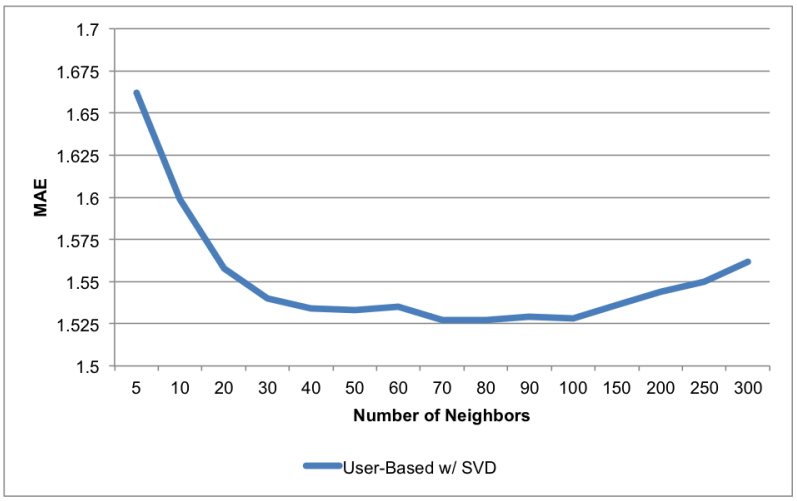


Figure 5.1: MAE on master's marks for different number of neighbors in our approach

20% of the neighbors (around 90). Indeed, we think that our recommendations can be very secure since in average we know that the student's mark will be at most one value below or prediction on this case study. As our error is around 1.5, any rounding done will act as a floor, rounding to 1. This is, if we predict a student's mark to be 16, we can affirm with confidence that his mark will be between 15-17. Furthermore, these results are interesting as they show that these system can be applied in an academic environment with a small amount of historical students, i.e in recently created programs or universities.

5.2.2 Effect of the number of features used

Other aspect that we were interested in was in the effect that the number of features had to our predictions. Usually, for each domain, there is an optimal number of features that captures the the set of features that are better on explaining the ratings deviation. In this experiment we varied the number of features, from 1 to 80, and recorded the obtained MAE. Once again, we performed five experiments for each chosen number of features and calculated the average of the achieved MAE.

By inspecting figure 5.2, we can notice that independently of the chosen number of features there is no great difference on the accuracy. This indicates that the chosen number of features does not have a great impact on the accuracy, which was not expected at first. However, it might be explained by the fact that our prediction calculation is strongly based on the defined neighborhood, where the SVD component acts a small refinement. However, we are able to see that with only one feature the accuracy is slightly lower, which makes sense since usually one feature is not enought to be able to explain the deviation on the ratings. It is also verifiable that as we reach the optimal number of features (20 features), the tendency is to increase slowly with the number of features. This is natural as with the increase on the number of features it also comes an increase on the unwanted noise in the SVD-based marks matrix.

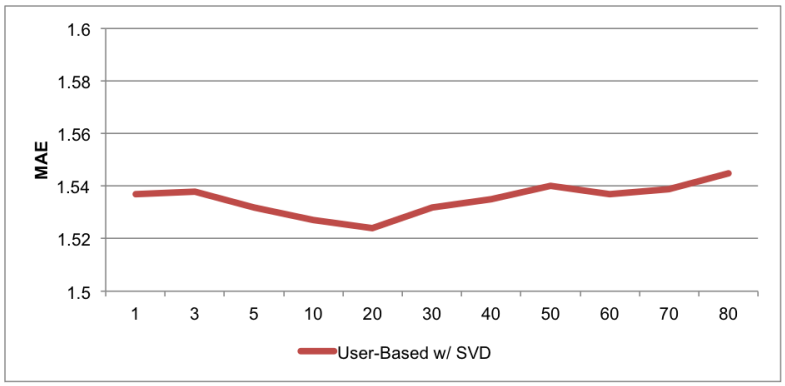


Figure 5.2: MAE variation with the number of hidden features

5.2.3 Effect of the number of training steps used on SVD

Despite the results presented just above, where we did not see a huge effect of the number of features used, we attempted to see if other SVD parameters had any particular effect on the results. We turned our attention to the number of training steps in the *gradient descent* algorithm. After recording the several MAE achieved using various number of steps (from 5 to 500), the results did not bring any particular conclusion.

In figure 5.3 we can see that independently the number of steps used, the MAE only suffers small changes. This goes inline with what we have concluded just above: the SVD step is used as just a refinement of the results, where the *user-based* component holds most of the obtained knowledge.

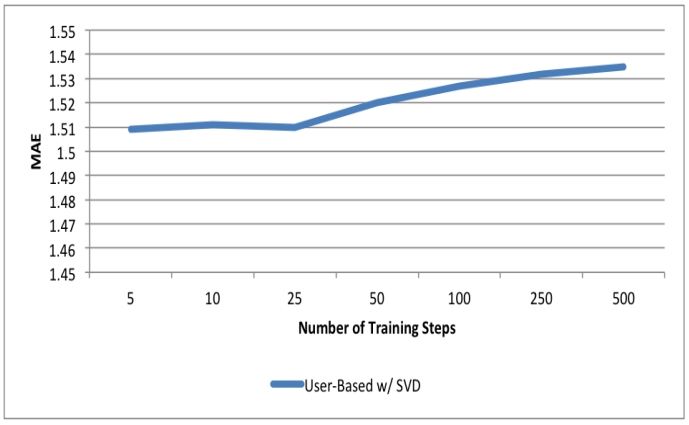


Figure 5.3: MAE variation with the number of training steps

5.2.4 Effect of the SVD technique used

Now that we have seen some of the effects of the configurable parameters of the *gradient descent SVD*, we want to see if there is some difference on the results if we used the original algebric SVD technique. To achieve this, we performed the experiment using the algebric SVD with three different initializations. We opted to initialize the student-course matrix, due to the recognized SVD bad performance with missing values [Sarwar et al., 2000]. In the first initialization, we filled the each student's missing marks with their corresponding bachelor's average. The second approach took another look by filling the courses

missing marks with the corresponding training students average on that course. The third approach, the most simple, filled all missing marks with the global average of the training set.

The results that can be seen in figure 5.4 help us to perceive that the different initializations have a deep effect on the achieved results. In fact, we were expecting this, as it is what is related on the existing literature [Sarwar et al., 2000]. Nevertheless, the best approach that used the algebric SVD (using course average on initialization) is around 12% worst than our approach with the *gradient descent SVD*.

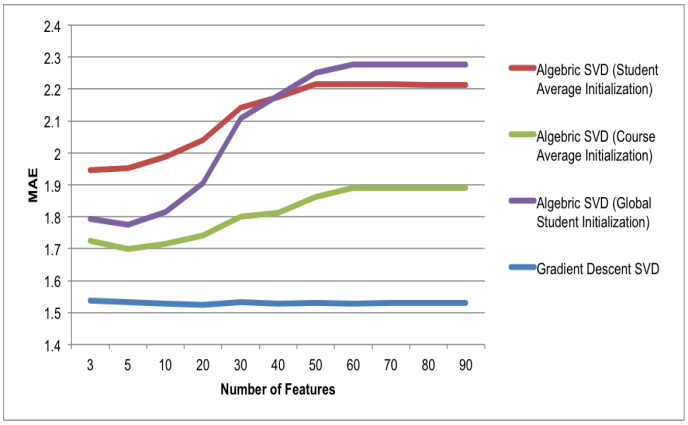


Figure 5.4: MAE variation using different SVD approaches on our proposal in terms of the number of features

5.2.5 Effect of the number of neighbors used in marks prediction

On this experiment, we wanted to verify if filtering the used neighbors to predict the final master's courses marks had some effect on the achieved MAE. Hence, we set up an experiment where we set the size of the *neighborhood* as 90 and the number of features to 20 (the optimal combination seen in section 5.2.2). We then varied the number of neighbors used in the marks predictions from 1 to 90, experimenting the whole possible range. Once again, we recorded the average of five experiments for each variation on the experiment.

The results of this experiment can be seen in figure 5.5. By inspecting this figure, one can see that, using 1 neighbor the results are slightly worst, since we are exploring few information. We reach the optimal minimum error by using 5 neighbors to predict the master's courses marks. From that point on, the noticeable trend is that the error gets larger until our maximum number of neighbors used in the prediction - where we use all the students in the *neighborhood* to predict the marks. We can then conclude that our filter on the neighbor used to predict the marks has a positive effect on the MAE achieved, despite the small improvement (it corresponds to a improvement of a 2% decrease compared to when we do not use any filter).

After this experiment, we decided to do another one where we set the number of neighbors used to predict the marks to 5, the number of features to 20, while changing the size of the original *neighborhood*. Our goal was to obtain a general idea of how the results of the approach using the neighbors post-selection compared to the approach where we do not use any neighbors post-selection (this is, when we use all the *neighborhood* to predict the marks).

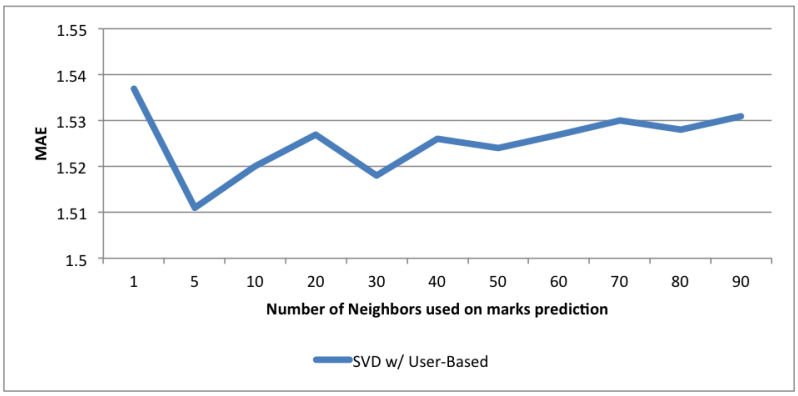


Figure 5.5: MAE variation with the number of neighbors used in marks prediction

What we could conclude was that the major trends presented on section 5.2.1 are inline with the results obtained on this experiment, as it can be seen in figure 5.6. The results show once again that the approach is worse when we use a small *neighborhood*. It is also seen that the error decreases and stabilizes when the number of neighbors is bigger than 70. The major difference to the results seen in section 5.2.1 is that this approach needs to use more neighbors (around 150 neighbors, agains 90 from the other approach) to reach its optimal error. In other words, this approach needs a neighborhood with half of the size of the training set to reach the lowest MAE (a value of almost 1.5). Finally, when the number of neighbors is bigger than 150 the error increases again. This results in a chart that presents the usual behavior on *user-based CF* approaches.

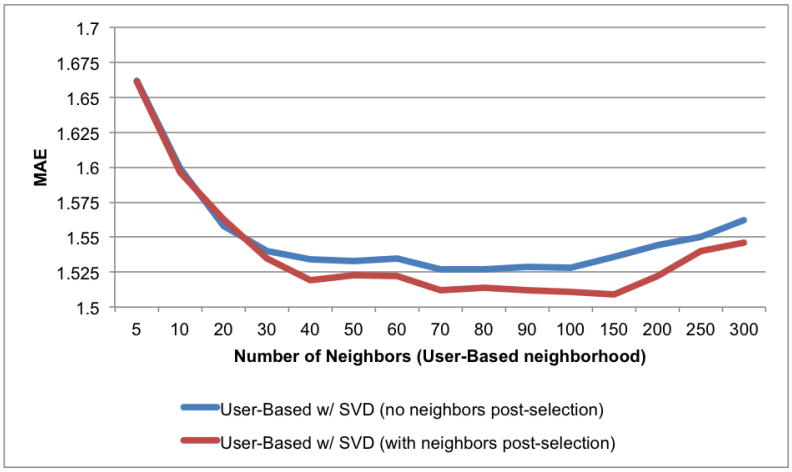


Figure 5.6: MAE comparison on using or not a neighborhood filter when predicting marks with different neighborhood sizes

5.2.6 Comparison with other approaches

In this section we will compare our approach with others so as to perceive the benefits that may come with our decisions over the system architecture. Above all, we expect to be able to see that our predictions reveal themselves more accurate than any baseline or regular approach used in this kind of problems.

Inferior Baselines. First, we will define and present two inferior baselines that are straight-forward and constitute a good starting point from where we can establish comparisons to our prediction results. The defined inferior baselines are:

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

Our first comparison will be to these two mentioned baselines. The results can be seen in figure 5.7. Notice that we displayed the results with several number of neighbors but that this only applies to our approach. Any of the baselines do not depend on neighbors and always presents the same MAE. We opted to present this chart like this so as to perceive how our approach compares to the baselines independently of the size of the *neighborhood*.

From analysis of the presented chart we can see that the *student's bachelor's average* approach is the one that presents the worst results, achieving a MAE around 2.2, on a mark scale from 1 to 20. This may be connected with the fact that the marks obtained on the master's 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. In our view this baseline presented a surprisingly good master's mark estimation considering that it is a very naïve approach.

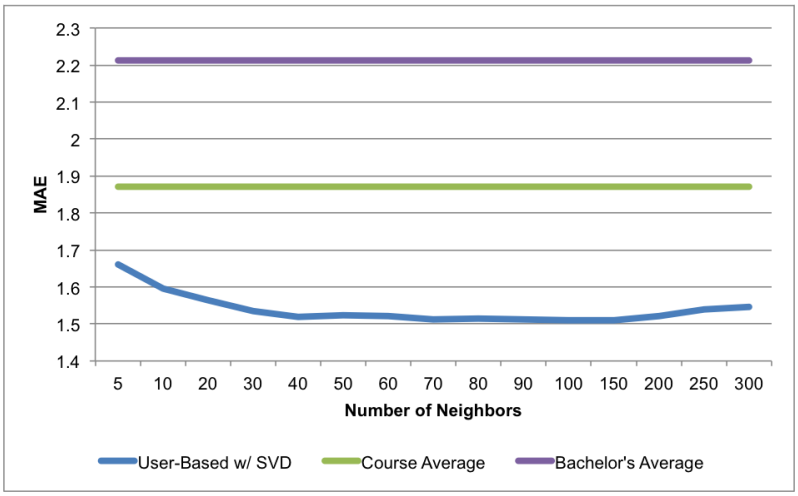


Figure 5.7: MAE comparison between our approach and inferior baselines

Finally, we have our approach which is labeled on the chart as *'User-Based w/ SVD'*. For this approach we set the configuration that gave us the optimal error on section 5.2.5. What we could conclude was that our approach presented good results against 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's average*, and 17% when comparing to the *course average*.

Item-Based CF. As we already made a comparison with the baselines and took a look into how the *user-based* component affects the accuracy of the predictions, we will now put our approach onto a big-

ger test. We will compare our approach with one of the most widely used recommendation techniques: a simple *item-based CF*. In this experiment, we varied the number of neighbors to consider when predicting the course ratings. This is, the number of ”most similar courses” to consider, where the similarity measure was based on the achieved mark. On this experiment the range of similar items considered went from 1 to 22 (the total number of bachelor's courses). We only considered bachelor's courses as these are they are the ones where we have information to predict the target student's marks.

In figure 5.8, we can see a chart where the red line represents the *item-based* approach, while the blue line stands for our proposed approach. Notice that the red line only uses a small number of neighbors since our range is limited to the number of bachelor courses (as explained above). Nevertheless, we can see that in this context the results achieved by *item-based* approach are poor, independently of the chosen number of neighbors. In fact, the optimal result achieved by this approach is around 0.55 mark points worse than the lower MAE obtained with our proposal.

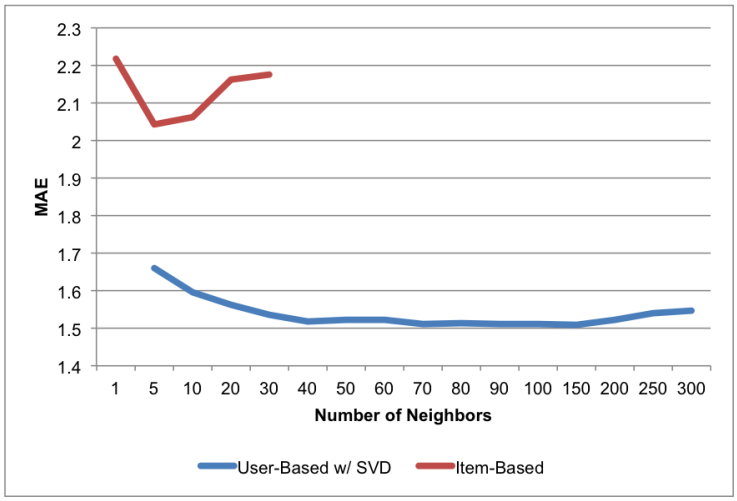


Figure 5.8: MAE comparison between our approach and a regular item-based CF

Gradient Descent SVD Since we are using a combination of *user-based CF* with *gradient descent SVD*, it is natural that we will compare our approach with each one of the base approaches. We will start by doing that comparison with the *gradient descent SVD*. We will do the comparison in terms of the number of features, since it is one of the most decisive parameters that are common in both approaches. For our approach, we used the configuration that presented the best results in section 5.2.5.

We can observe in figure 5.9 that our approach reveals itself as a better choice, although the good results achieved by the original *gradient-descent SVD*. 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.

User-Based CF. By comparing to this approach, we want to see if the creation of the SVD marks matrix brings any benefit to the prediction of the final master's marks, and if we can justify our option of using SVD on top of the *user-based* approach. To achieve it we used our *user-based* component

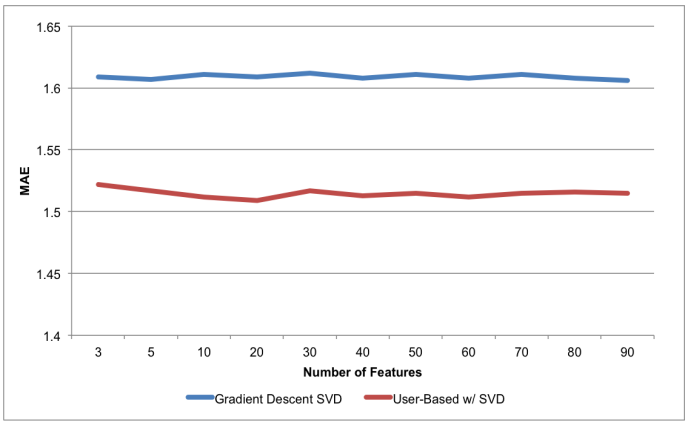


Figure 5.9: MAE comparison between our approach and a pure gradient descent SVD

and predicted the master's courses marks right after the *neighborhood* selection. We did this for several *neighborhood* sizes, ranging from 5 to 300.

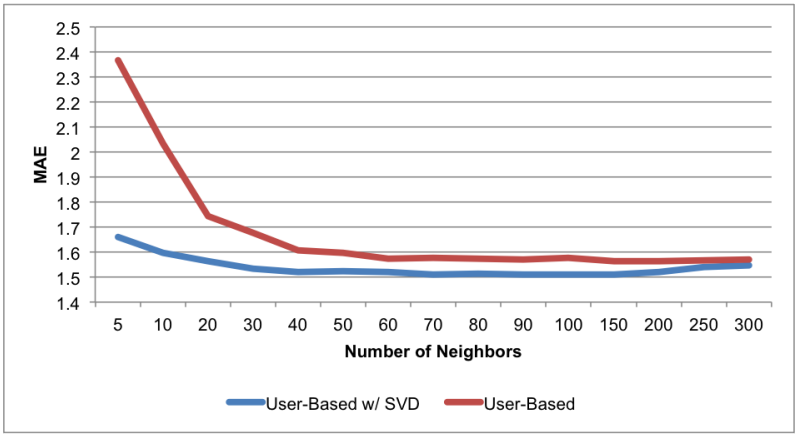


Figure 5.10: MAE comparison between our approach and a regular user-based CF

If we look into figure 5.10, one can see the MAE results of our approach and of the *user-based CF*. What it can be perceived is 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. With this said, 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 master's 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.

5.3 Recommendations Analysis

In this section we will try to analyze some aspects of the produced recommendations, despite our larger focus in evaluating the prediction accuracy. We will inspect how diverse are our recommendations, to see if there is any tendency to recommend the same courses or to not recommend some courses at all. Then, we will inspect how students follow our recommendations, i.e what is their acceptance (see section 5.1). However, as we have already stated, this evaluation is not truly reliable since we are doing an offline analysis. This way, we may just analyze if we are recommending courses that are usually part of the students' choices. Finally, we will verify how our recommendation affect the students' master's academic achievements.

5.3.1 Recommendations Marks

In this section we will see on how our recommendations might affect students' academic achievements, in particular, their master's average. As we are doing an offline evaluation of our system, using historical data, we may predict the mark for all master's courses but then we are not able to verify if all the predictions were correct. Hence, for each student, and in the best case scenario we will have 15 master's courses that we may use in the students' overall master's average. This factor might influence the results that we are trying to present as the different approaches we will test may have different acceptance rates, which indirectly affects the global average on the followed recommendations. To have a comparison to our approach we defined other two baselines, more appropriate to what we are trying to evaluate here:

- *Best Marks:* This baseline looks into the students-courses matrix and recommends the N master's courses where students in average obtain higher marks.
- *Most Popular Courses:* This approach inspects the students-courses matrix and recommends the N most popular courses, this is, the courses that were the most frequented ones in the historical trainining data.

To complete the experiment, we calculated the average mark of the courses recommended that the students in the testing set enrolled on. This was done for our approach, a pure user-based CF and both baseline approaches. From observation of table 5.1 , 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 mark prediction error in order to produce good quality recommendations. Though, to be completely accurate the number of followed recommendations for each of the approaches should be the same, which is not.

5.3.2 Recommendations Coverage and Acceptance

In this section we want to evaluate how diverse are our recommendations. It is important that our system has the ability to produce personalized recommendations. We believe that if our system indeed

Technique	Average Mark
User-Based w/ SVD	16,245
User-Based	16,05
Best Marks	16,454
Most Frequented	14,59

Table 5.1: Average mark on followed recommendations

recognizes the skills of each student, then it will be able to produce specific recommendations to each student.

In order to verify this, we run our recommendation process and recommended 15 master's courses for each student. We registered which were the courses recommended and how many times each of those courses were recommended. From among the 60 master's courses susceptible of being recommend, 33 were actually recommended from our system. This is, around half of the courses were suggested by our system. Although, the coverage was not as diverse as we would ideally want, we are able to state that our system does not fall in the error of just recommending the ”easiest” courses. We state this, since we simulated a recommendation where we would recommend the 15 courses with best marks, which allowed us to verify that only 5 of these courses were part of our 33 recommended courses.

Another aspect that we verified, was that 7 of those 33 courses were always recommended to students. Hence, our system suggests that there is a set of courses that easily fit every students capabilities. One could say that these courses may be some of the ”easiest” courses, as we recommend the courses with predicted higher mark. However, we verified that from these 7 ”always recommended” courses, only 2 of them were on our recommendation of the courses with historical higher mark average.

We also observed that 12 from the 15 most frequented courses by students were present on the 33 courses recommended. Hence, we believe that our system is giving a big weight to the data contained on those courses, since they are the ones that appear often in the students results, maybe creating a trend to recommend them.

As we mentioned, we are evaluating our proposal performing an offline analysis, that in terms of acceptance restricts the results, since we are not actually seeing the ”true” choices of the students. However, we were interested in seeing which was the acceptance of our proposal comparing with other approaches and some baselines.

In this experiment, we opted to compare our approach with the *user-based CF* (which was the second best approach, just after the one we proposed), and with the two baselines defined in section 5.3.1. For our proposal and the *user-based CF*, we used the configuration that gave the best results in terms of MAE, as shown in section 5.3.1.

What we could see, as presented in table 5.2, was that on our approach almost half of the recommendations were followed by students, which follows our idea that students are not choosing the best courses according their skills. If we look into the *user-based* approach, we see that the acceptance is slightly lower than our approach but pretty similar. This makes sense, as our approach is highly dependent of the *user-based* component. This also may implicate that our positive difference in the

acceptance is due to our *SVD* component. It was also interesting to see that the baseline *Best Marks* had a low acceptance. This means that students are not choosing the ”easiest” courses. However, as the high acceptance of the baseline *'Most Frequented'* points out, there is a set of ”core” courses that all the students end on enroll on.

Technique	Acceptance
User-Based w/ SVD	0,401
User-Based	0,362
Best Marks	0,268
Most Frequented	0,912

Table 5.2: Acceptance on recommendations

5.4 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. We will see how our approach to estimate the student's missing values affects the accuracy of our marks prediction.

We will start by contextualizing this problem and explain how we will evaluate it. As we know, we aim to use *ASAP classifiers* to estimate missing values on students bachelor profiles prior to recommendation. Hence, we are trying to contest the worst-case scenario where we have to input to our recommendation process an incomplete student's profile. With this said, we think that it is natural to consider this scenario as one of our baselines. We did an experiment to compare this baseline to the regular scenario where we have full knowledge of every students' bachelor's profile. In this experiment, our regular scenario was the one where we had the bachelor's marks for all three years. On the other side, we set our baseline scenarios as one where all the testing students' profiles were incomplete, having only information about the first or the two first years of bachelor. This last scenario can easily approximate from a real-life case, where a students wants to enroll on master's courses despite not having finished some of the last years' courses.

The results in terms of prediction accuracy for both baselines and our regular approach (using the three years to represent the testing students profiles) can be seen in figure 5.11. 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. This is, the error is larger when we only use the marks of the bachelor's first year to represent the profile of the student being recommended. Then, the prediction error decreases when representing the student with the first two years' marks. The error finally reaches its minimum when we always have complete students' profiles. However, these results hide some suprises: we have a better prediction accuracy using only the first year on the students' profiles than with any of the baselines defined in section 5.2. These statements combined suggest that our approach is be good in capturing the factors that explain the marks even when the neighbors identification is not done in the

most desirable conditions.

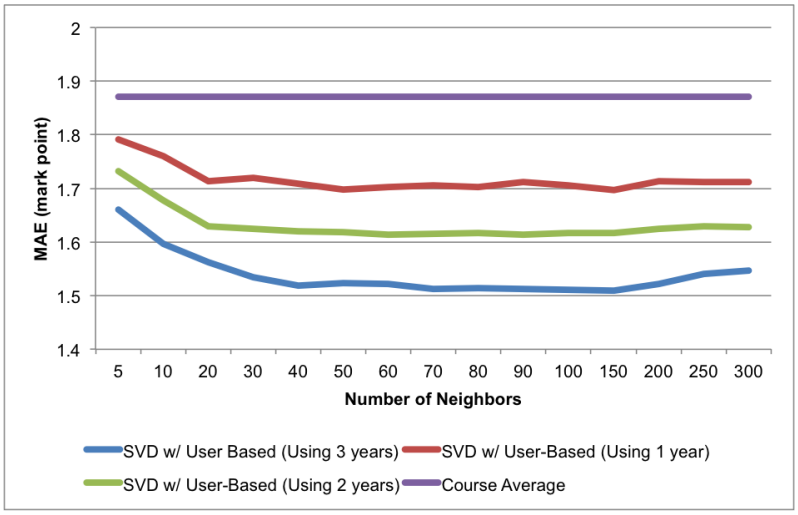
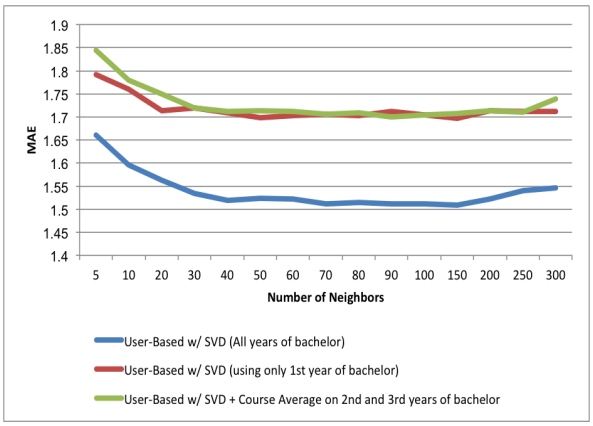


Figure 5.11: MAE comparison between different amount knowledge levels on students' profiles

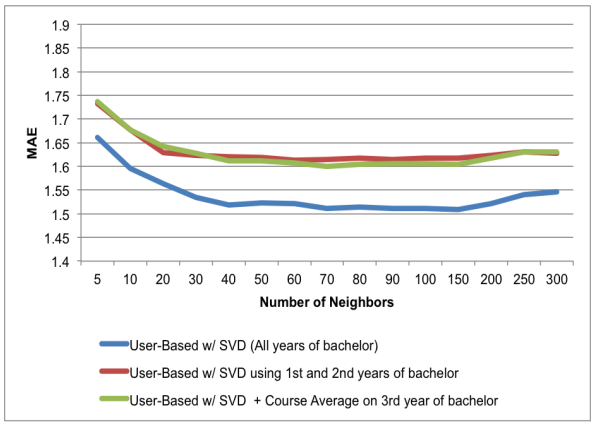
We defined another inferior baseline that fills missing bachelor's marks with the average mark obtained by students on the bachelor's course where the value is missing. We are now ready to use *ASAP classifiers* and verify how the estimated values work out when passed to our recommendation process. Our "superior" baseline, which constitutes our goal, is our proposed approach using the complete students' profiles. If we were able to reach this baseline it would mean that our ASAP estimations would correspond to the real marks, or that at least were able to reflect the "real" behavior when identifying the neighborhood and predicting the marks. We drilled down this into two experiments using ASAP classifiers. One where we only had data over the first year of students' bachelor and another where the data was over the first two years of bachelor. In each of the experiments, we started by training the *ASAP classifiers* with the training students, so as to learn a classifier for each bachelor course. We then passed the incomplete students' profiles into the *ASAP classifiers*, which estimated the marks of the missing bachelor's courses. With the students' profiles completed, we just had to pass them into our regular recommendation process, verify our predictions and record the MAE results. To record these results, we used the configuration that gave the best results in the prediction accuracy evaluation.

By inspection of figure 5.12(a) we can 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. The same situation occurs when the scenario is the one where we use the first two years of bachelor to represent the students, as it can be seen in figure 5.12(b). 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.

If we look to figure 5.13(a), we can verify that *ASAP* estimation is slightly more accurate than any baseline. We can notice the same trend in the accuracy in figure 5.13(b), where the difference is that the known students' marks are the ones from the two first bachelor's years. However, the results cannot achieve our superior baseline that is the case where we have full data over students' bachelor marks. In both of the presented scenarios, the *ASAP* estimation is the one that presents best results, but it still



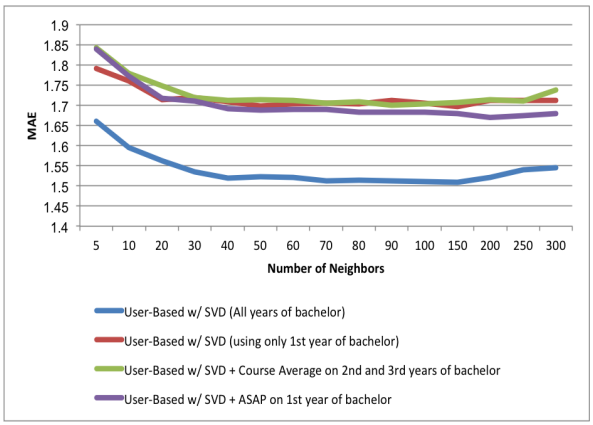
(a) With knowledge of 1 year of students' profile



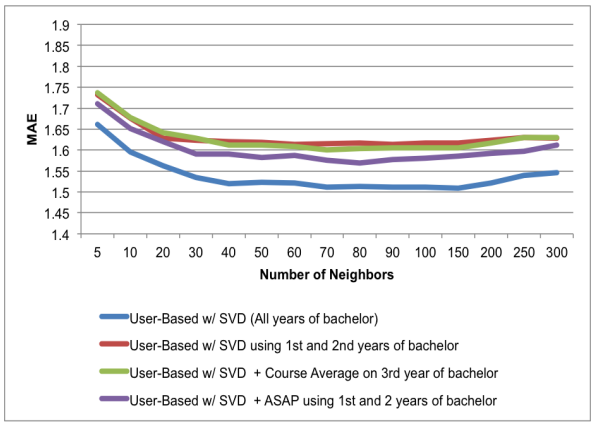
(b) With knowledge of 2 years of students' profile

Figure 5.12: Prediction error comparison on lower baselines when using complete and incomplete student profiles

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.



(a) With knowledge of 1 year of students' profile



(b) With knowledge of 2 years of students' profile

Figure 5.13: Prediction MAE comparison between using complete or incomplete student profiles

Chapter 6

Conclusions

The selection of which courses to attend on masters is one of the most decisive steps of an undergraduate student. However, students still struggle to chose the best set of courses for them, and in the majority of the cases they do not have the required support and counseling needed to make such an important decision. In this thesis, we show that we can use known recommendation techniques to recommend masters courses to students, that are not only interesting to them, but that are the most adequate to their skills. More precisely, we show that we can combine sucessful recommendation techniques, such as *Singular Value Decomposition* and *User-Based CF* to the educational paradigm, so as to explore historical student data and be able to explain and predict students' master's courses marks.

6.1 Contributions

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. We presented a proposal that relied on the combination and adaption of two of the most successful recommendation techniques to our educational context. In particular, what we proposed was to use *singular value decomposition* on top of a *user-based collaborative filtering* approach. By using *user-based CF* our goal was to try to explore the knowledge hold on the several historical courses results of a large number of students. We then combined it with *singular value decomposition* so as to search for the features that explain the ratings in the formed *neighborhood*. The found features are then used to predict the masters courses marks for each target student. Hence, we have used these techniques as a new way to look at this problem, 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. Though, we do still believe that more can be done to obtain better results, as we will explain in the section below. We have shown that our approach is able to recognize which courses are better for each student, as our recommendations are the ones where the students achieve better marks, almost achieving our superior baseline. We also have shown that our system gave some different recommendations to different students, which allows

us to believe that is being able to create personalized recommendations to each student, and not just recommending the courses that have the best marks.

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. We concluded that although our results show better results than some baselines, more work has to be done so as to avoid missing values while not introducing unwanted noise.

6.2 Future Work

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. This way, we can see a lot of open different paths to explore, and we are firmly rooting that in the end future researches can be joined in a suberb recommendation systems for course selection. In this section we will advance some ideas for future work and discuss some open-issues.

In a first look, we find interesting the oportunity to explore the development of different recommendation systems, that recommend courses according to each of the different factors that we have listed above. We could then perform an *ensembling* of each one of the recommenders to produce a final recommendation. Secondly, it could be very interesting to explore the idea of recommending courses under certain constraints. This way, students' could be recommended with the best set of courses that go along with user-defined contraints. It could be that only a subset of courses would be available in different terms, or even that some courses require the approval on other ones first. Another interesting view is on how to better preprocess the students-courses matrix before applying any recommendation technique (either SVD, User-Based, Item-Based). It would be fun to explore different domain-speficic matrices initialization, encapsulating several academic factors that somehow could help to better explain the variation on students achieved marks. Also, different research scenarios should be explored when data about the student being recommended is limited, i.e not the expected. We could experiment some recommendation and classification techniques to try to predict the missing student data.

A content-based component could be explored so as to not only recommend courses that potentially suit the students' skills but that also match with their interests, so as to refine our recommendations. Another interesting feature that could be added was the ability to learn with active feedback from the students. Also, live recommendations could be given, at each semester to help students to chose their courses at each phase of their masters. This feature would enhance the results, as it would use more information each semester, having the potential to produce more informed recommendations.

Ideally, this work should be experimented with real students, which would enable the analysis of the real acceptance of this system by the students. It would be really important to verify if it would help students to really solve the problem that we are approaching.

In educational contexts where the universe of courses is very large, we could try to diminish the number of courses, by merging similar courses. This could be done by using *Information Retrieval* techniques that could explore the courses programs, so as to find courses that approach the same contents.

Bibliography

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.

J. B. Agapito and A. Ortigosa. Detecting symptoms of low performance using production rules. In T. Barnes, M. C. Desmarais, C. Romero, and S. Ventura, editors, *EDM*, pages 31–40. www.educationaldatamining.org, 2009. ISBN 978-84-613-2308-1.

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

R. S. Baker, S. Gowda, and A. T. Corbett. Automatically detecting a student's preparation for future learning: Help use is key. In *Proceedings of the 4th International Conference on Educational Data Mining*, pages 179–188, 2011.

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

K. Barker, T. Trafalis, and T. Reed Rhoads. Learning from student data. *Proceedings of the 2004 Systems and Information Engineering Design Symposium*, 2004.

A. Baylari and G. A. Montazer. Design a personalized e-learning system based on item response theory and artificial neural network approach. *Expert Syst. Appl.*, 36(4):8013–8021, 2009.

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.

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. ISBN 978-1-60558-551-2. doi: 10.1145/1559845.1559994.

Y. Bergner, S. Droschler, G. Kortemeyer, S. Rayyan, D. Seaton, and D. Pritchard. Model-based collaborative filtering analysis of student response data: Machine-learning item response theory. In K. Yacef, O. R. Zaïane, A. Hershkovitz, M. Yudelson, and J. C. Stamper, editors, *EDM*, pages 95–102. www.educationaldatamining.org, 2012. ISBN 978-1-74210-276-4.

M. Bilgic and R. Mooney. Explaining recommendations: Satisfaction vs. promotion. *Beyond Personalization Workshop, IUI*, 2005.

V. Bresfelean, M. Bresfelean, N. Ghisoiu, and C.-A. Comes. Determining students academic failure profile founded on data mining methods. In *Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on*, pages 317–322, 2008.

R. Burke. Knowledge-based recommender systems. In *ENCYCLOPEDIA OF LIBRARY AND INFORMATION SYSTEMS*, page 2000. Marcel Dekker, 2000.

R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, pages 1–29, 2002.

Y.-C. Chang, W.-Y. Kao, C.-P. Chu, and C.-H. Chiu. A learning style classification mechanism for e-learning. *Computers & Education*, 53(2):273–285, 2009.

H. Chu and G. Hwang. A computerized approach to diagnosing student learning problems in health education. *Asian Journal of Health . . .*, 1(1):43–60, 2006.

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.

P. T. Crespo and C. Antunes. Predicting teamwork results from social network analysis. *Expert Systems*, 2013. ISSN 1468-0394. doi: 10.1111/exsy.12038.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6): 391–407, 1990.

M. Deshpande and G. Karypis. Item based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22:143–177, 2004.

M. D. Ekstrand, J. T. Riedl, and J. A. Konstan. Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173, Feb. 2011. ISSN 1551-3955. doi: 10.1561/1100000009. URL `http://dx.doi.org/10.1561/1100000009`.

R. Farzan and P. Brusilovsky. Social Navigation Support in a Course Recommendation System. In V. Wade, H. Ashman, and B. Smyth, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems: 4th International Conference, AH 2006*, volume 4018 of *Lecture Notes in Computer Science*, pages 91–100, Berlin, 2006. Springer Verlag. doi: 10.1007/11768012_11.

S. Funk. Netflix update: Try this at home, December 2006. URL `http://sifter.org/~simon/journal/20061211.html`.

E. García, C. Romero, S. Ventura, and C. de Castro. An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. *User Model. User-Adapt. Interact.*, 19(1-2):99–132, 2009.

D. García-Saiz and M. E. Zorrilla. A promising classification method for predicting distance students' performance. In K. Yacef, O. R. Zaïane, A. Hershkovitz, M. Yudelson, and J. C. Stamper, editors, *EDM*, pages 206–207. www.educationaldatamining.org, 2012. ISBN 978-1-74210-276-4.

L. Ge, W. Kong, and J. Luo. Courseware recommendation in e-learning system. In W. Liu, Q. Li, and R. W. H. Lau, editors, *ICWL*, volume 4181 of *Lecture Notes in Computer Science*, pages 10–24. Springer, 2006. ISBN 3-540-49027-2.

K. I. Ghauth and N. A. Abdullah. Learning materials recommendation using good learners' ratings and content-based filtering. *Educational Technology Research and Development*, 58(6):711–727, 2010. doi: 10.1007/s11423-010-9155-4.

F. Gutierrez, D. Dou, S. Fickas, and G. Griffiths. Providing grades and feedback for student summaries by ontology-based information extraction. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 1722–1726, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1156-4. doi: 10.1145/2396761.2398505.

J. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. pages 241–250, 2000.

J. L. Herlocker, J. A. Konstan, L. G. Terveen, John, and T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53, 2004.

N. T. N. Hien and P. Haddawy. A decision support system for evaluating international student applications. In *Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007. FIE '07. 37th Annual*, pages F2A–1–F2A–6, 2007. doi: 10.1109/FIE.2007.4417958.

M.-H. Hsu. A personalized English learning recommender system for ESL students. *Expert Systems with Applications*, 34(1):683–688, Jan. 2008. ISSN 09574174. doi: 10.1016/j.eswa.2006.10.004.

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.

D. KABAKCHIEVA. Analyzing university data for determining student profiles and prediction performance. In *Proceedings of the 4th International Conference on Educational Data Mining*, pages Pages 347–348, 2001.

P. Karampiperis and D. Sampson. Adaptive learning resources sequencing in educational hypermedia systems. *Educational Technology & Society*, 8(4):128–147, 2005.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263. URL `http://dx.doi.org/10.1109/MC.2009.263`.

S. B. Kotsiantis and P. E. Pintelas. Predicting students' marks in hellenic open university. In *ICALT*, pages 664–668. IEEE Computer Society, 2005. ISBN 0-7695-2338-2.

Z. J. Kovacic. Predicting student success by mining enrolment data. *Research in Higher Education Journal*, 15:1–20, 2012.

A. Krištofič. Recommender system for adaptive hypermedia applications. *IIT. SRC 2005: Student Research Conference*, pages 229–234, 2005.

G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan. 2003. ISSN 1089-7801.

J. Lu. Personalized e-learning material recommender system. In *In: Proc. of the Int. Conf. on Information Technology for Application*, pages 374–379, 2004.

M-Helene, B. Meyer, and K. L. Mannock. Understanding novice errors and error paths in Object-oriented programming through log analysis. In *Proceedings of the Workshop on Educational Data Mining at the 8th International Conference on Intelligent Tutoring Systems (ITS 2006). Jhongli, Taiwan.*, pages 13–20, June 2006.

D. Martinez. *Predicting Student Outcomes Using Discriminant Function Analysis*. Distributed by ERIC Clearinghouse [Washington, D.C.], 2001.

E. N. Ogor. Student academic performance monitoring and evaluation using data mining techniques. In *Proceedings of the Electronics, Robotics and Automotive Mechanics Conference*, CERMA '07, pages 354–359, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2974-7. doi: 10.1109/CERMA.2007.127.

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.

A. Parameswaran, P. Venetis, and H. Garcia-Molina. Recommendation systems with complex constraints: A courserank perspective. *Transactions on Information Systems (TOIS) – To Appear*, June 2011.

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. ISBN 978-0-615-37529-8.

P. Resnick and H. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported*

Cooperative Work, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM. ISBN 0-89791-689-1. doi: 10.1145/192844.192905. URL http://doi.acm.org/10.1145/192844.192905.

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.

C. Romero, S. Ventura, and P. D. Bra. Knowledge discovery with genetic programming for providing feedback to courseware authors. *Journal on User Modeling and User-Adapted Interaction*, (14(5)): 425–464, 2005.

C. Romero, S. Ventura, P. G. Espejo, and C. Hervás. Data mining algorithms to classify students. In *In Proc. of the 1st Int. Conf. on Educational Data Mining (EDM'08), p. 187191, 2008. 49 Data Mining 2009*, 2008.

N. Rovira-Asenjo, T. Gumí, M. Sales-Pardo, and R. Guimerà. Predicting future conflict between team-members with parameter-free models of social networks, 2013.

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

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. ISBN 1-58113-348-0. doi: 10.1145/371920.372071.

B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000.

D. Shangping and Z. Ping. A data mining algorithm in distance learning. In *Computer Supported Cooperative Work in Design, 2008. CSCWD 2008. 12th International Conference on*, pages 1014–1017, 2008. doi: 10.1109/CSCWD.2008.4537118.

L. Shen and R. Shen. Learning content recommendation service based-on simple sequencing specification. In W. Liu, Y. Shi, and Q. Li, editors, *ICWL*, volume 3143 of *Lecture Notes in Computer Science*, pages 363–370. Springer, 2004. ISBN 3-540-22542-0. URL http://dblp.uni-trier.de/db/conf/icwl/icwl2004.html#ShenS04.

N. Soonthornphisaj, E. Rojsattarat, and S. Yim-ngam. Smart e-learning using recommender system. In D.-S. Huang, K. Li, and G. W. Irwin, editors, *ICIC (2)*, pages 518–523, 2006. ISBN 3-540-37274-1.

J.-M. Su, S.-S. Tseng, W. Wang, J.-F. Weng, J.-T. D. Yang, and W.-N. Tsai. Learning portfolio analysis and mining for scorm compliant environment. *Educational Technology & Society*, 9(1):262–275, 2006. doi: http://www.ifets.info/journals/9_1/21.pdf.

J. Superby, J. Vandamme, and N. Meskens. Determination of factors influencing the achievement of the first-year university students using data mining methods. In *Workshop on Educational Data Mining*, pages 37–44, 2006.

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. ISBN 978-1-74210-276-4. URL http://dblp.uni-trier.de/db/conf/edm/edm2012.html#SurpateanSM12.

K. Taha. Automatic academic advisor. In *CollaborateCom*, pages 262–268. IEEE, 2012. ISBN 978-1-4673-2740-4. URL http://dblp.uni-trier.de/db/conf/colcom/colcom2012.html#Taha12.

N. Thai-Nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme. Recommender system for predicting student performance. *Procedia CS*, 1(2):2811–2819, 2010.

K. H. Tsai, T.-C. Hsieh, T. K. Chiu, M.-C. Lee, and T. I. Wang. Automated course composition and recommendation based on a learner intention. In J. M. Spector, D. G. Sampson, T. Okamoto, Kinshuk, S. A. Cerri, M. Ueno, and A. Kashihara, editors, *ICALT*, pages 274–278. IEEE Computer Society, 2007. ISBN 978-0-7695-2916-5. URL http://dblp.uni-trier.de/db/conf/icalt/icalt2007.html#TsaiHCLW07.

H. Unelsrød. Design and Evaluation of a Recommender System for Course Selection. Master's thesis, Norwegian University of Science and Technology, 2011. URL http://ntnu.diva-portal.org/smash/record.jsf?pid=diva2:444229.

C. Vialardi, J. Bravo, and A. Ortigosa. Improving AEH Courses through Log Analysis. *J. UCS*, 14(17): 2777–2798, 2008. URL http://dblp.uni-trier.de/rec/bibtex/journals/jucs/VialardiBO08.

C. Vialardi, J. Bravo, L. Shafti, and A. Ortigosa. Recommendation in higher education using data mining techniques. In *Proceedings of Second Educational Data Mining conference*, pages 190–199. Universidad de Córdoba, Córdoba, Spain, 2009.

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. URL http://scholar.google.com/scholar.bib?q=info:S4S3fCQfXWUJ:scholar.google.com/&output=citation&hl=de&as_sdt=0,5&ct=citation&cd=0.

X. Wang and F. Yuan. Course recommendation by improving bm25 to identity students' different levels of interests in courses. In *Proceedings of the 2009 International Conference on New Trends in Information and Service Science*, NISS '09, pages 1372–1377, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3687-3. doi: 10.1109/NISS.2009.104. URL http://dx.doi.org/10.1109/NISS.2009.104.

Y. Wang, N. T. Heffernan, and J. E. Beck. Representing student performance with partial credit. In R. S. J. de Baker, A. Merceron, and P. I. P. Jr., editors, *EDM*, pages 335–336. www.educationaldatamining.org, 2010. ISBN 978-0-615-37529-8. URL http://dblp.uni-trier.de/db/conf/edm/edm2010.html#WangHB10.

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. ISSN 0957-4174. doi: 10.1016/j.eswa.2008.09.008. URL http://dx.doi.org/10.1016/j.eswa.2008.09.008.

F. Zhu, H. Ip, A. Fok, and J. Cao. Peres: A personalized recommendation education system based on multi-agents and scorm. In H. Leung, F. Li, R. Lau, and Q. Li, editors, *Advances in Web Based Learning – ICWL 2007*, volume 4823 of *Lecture Notes in Computer Science*, pages 31–42. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-78138-7. doi: 10.1007/978-3-540-78139-4_4. URL http://dx.doi.org/10.1007/978-3-540-78139-4_4.

J. Zimmermann, K. H. Brodersen, J.-P. Pellet, E. August, and J. M. Buhmann. Predicting graduate-level performance from undergraduate achievements. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero, and J. C. Stamper, editors, *EDM*, pages 357–358, 2011. ISBN 978-90-386-2537-9.