Faculdade de Engenharia da Universidade do Porto

Market-based Higher Education Course Recommendation

Ana Isabel Neves Alves de Sousa

Dissertation Planning

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Professora Doutora Ana Cristina Costa Aguiar

Second Supervisor: Professora Doutora Luísa Lima

February 17, 2016

Market-based Higher Education Course Recommendation

Ana Isabel Neves Alves de Sousa

Mestrado Integrado em Engenharia Informática e Computação

February 17, 2016

Abstract

The choice of a higher education program is determinant in the future career. This choice needs to take into account several aspects, such as its future employability, if it provides the competences required for a desired job, as well as whether its scope of depth and breath fits the needs of the student. Having this is mind, it is worth to invest the time to make a better informed decision. Since alumni represent the most trusted source of information about the paths a certain degree confers access to, we propose a recommendation system based on alumni data to solve this problem.

Using the information about the alumni and job offers, we aim to find out how to use it in order to recommend the most appropriate higher education programs to achieve a certain job. The recommender system will have as input the user's desired career and, by making a match between the competences required for that career and the competences that a program provides, it will output a ranking of higher education programs to take. Since this goal fits into the recommender systems category, we present an overview on the state of the art of this research area, with special focus on collaborative filtering methods and evaluation of recommendation tasks.

One of the most important tasks in this area is to build a solid data set of information about alumni (namely education and skills, jobs and required job skills) for the recommender to work in. This was our first addressed task, which we will present in detail in the present report.

Finally, we present the next steps to take in order to build the recommendation system for higher education programs.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Goals . . . 1
- 1.3 Document Structure . . . 2

2 Recommender Systems **3**
- 2.1 Content-based Methods . . . 4
- 2.2 Collaborative Methods . . . 5
  - 2.2.1 Memory-based . . . 5
  - 2.2.2 Model-based . . . 7
  - 2.2.3 Challenges of Collaborative Filtering . . . 8
- 2.3 Hybrid Approaches . . . 10
- 2.4 Evaluation of Recommendation Tasks . . . 10
  - 2.4.1 Predictive accuracy metrics . . . 10
  - 2.4.2 Classification accuracy metrics . . . 11
  - 2.4.3 Rank accuracy metrics . . . 11
- 2.5 Conclusions . . . 11

3 System Overview **13**
- 3.1 Problem . . . 13
- 3.2 Social Networks . . . 14
  - 3.2.1 ITJobs . . . 15
  - 3.2.2 Jobvite . . . 15
  - 3.2.3 Landing.jobs . . . 16
  - 3.2.4 LinkedIn . . . 16
  - 3.2.5 Stack Overflow Careers . . . 16
  - 3.2.6 Upwork . . . 16
  - 3.2.7 Xing . . . 17
  - 3.2.8 Other sources . . . 17
- 3.3 Data Set Composition . . . 17
  - 3.3.1 Data Cleaning . . . 18
- 3.4 Conclusions . . . 18

4 Conclusions and Future work **21**
- 4.1 Work plan . . . 21

References **23**

A Work Plan **25**

List of Figures

3.1 Logic architecture. Full lines represent information that we have. Dashed lines represent relationships we need to establish. 14
3.2 Solution. The input is the user's desired job. That job has a set of competences required, than can be acquired in a set of programs. The output is the ranking of programs recommended. 15
3.3 Data model 18

A.1 Work plan 25

LIST OF FIGURES

List of Tables

2.1 An example of an user-item matrix 4

3.1 Data sources summary. Crosses represent data that is present in each API. 19

Chapter 1

Introduction

1.1 Motivation

Choosing a higher education course is not an easy decision. There is a wide variety of higher-education programs available, with important differences both in depth and in breadth of lectured topics. Their future employability is especially relevant considering the current lack of detailed employability statistics. When a student enrolls in a degree, he usually does not know the full employability potential of a program. As a consequence, having so many options can represent a real challenge. Knowing that this decision has a large impact in the future career makes even more important to invest time to make a better-informed decision. On one hand, sometimes it is straightforward to think about a dream job, but the higher-education program to take in order to achieve that goal is not so clear. On the other hand, the competences that a program claims to give their students are not always the ones that the alumni feel they have acquired there, and since they not only have attended the program but also felt the impact of their decision in their careers, it is reasonable to consider them as the most trusted source of relevant and real information about some of the possible paths that the degree confers access to.

1.2 Goals

The main goal of this dissertation is to know how to use alumni and job posts information to recommend higher education programs appropriate to achieve a certain job. To achieve that, we propose design a recommendation algorithm and its associated system for higher-education programs. We will elaborate a proof of concept with Computer Science as the target discipline to reduce the scope of recommendations and facilitate the system validation. In order to do that, the sources of relevant information – social networks designed for business purposes – need to be identified and represented in the form of competences acquired, and various algorithms for the task need to be researched and experimented. Through these social networks, we expect to extract the required information about the alumni, such as competences acquired and education, as well as job offers and skills associated to them. Finally, due to the subjective nature of a recommendation, we propose to validate the proposed solution manually.

The recommendation system will have as input the user's desired career and as output a ranking of possible programs that he could take in order to be closer to that goal. This ranking will be computed taking into account competences acquired by a set of alumni, the capabilities that every program provides and the ones required for that specific career, doing a matching between the last two.

1.3 Document Structure

In addition to this introduction, this report contains three other chapters. In chapter 2, we describe the state of the art and present related work. In chapter 3, we provide a more detailed description of the system and its goals, as well as a overview on the major task achieved until this date which is the data set creation. Finally, in chapter 4 we present the conclusions and establish the work plan.

Chapter 2

Recommender Systems

Nowadays – even though we may not always realise it – recommendations are ubiquitous. For example, when someone wants to watch a movie, that person will probably ask one of his friends his opinion about it; when making a decision for purchasing a product, the most common method for deciding the best choice is to search for online reviews. Automated recommender systems are quite similar to these real-life recommendations, with the considerable advantage that we can have access to a much wider range of users who have the same preferences.

Recommender systems have become an important and interesting area both in academia and in industry, with a wide range of work being done in this rapidly expanding area [AT05]. Some of the most well-known recommender systems in industry are Amazon[1] that recommends books, CDs and other products, Netflix[2] that recommends movies and series, and MovieLens[3] that also provides movies recommendations.

A typical recommender system consists of:

- A set of users C;
- An active user c - the one the recommendation will be given to;
- A set S that comprises all items that can be recommended.

The goal is to maximise the usefulness/utility of an item s to an user c, which is represented by the utility function $u(c,s)$. This utility is often represented as a rating that expresses the opinion of a particular user c towards item s. Ratings can be either explicit – expressed as a value on a scale, or implicit – expressed as a purchase or a click. Ratings given by users can be converted into a *user-item matrix*. For example, in the user-item matrix in table 2.1, *Maria* is the active user and to who we want to compute a recommendation.

[1] Available at amazon.com
[2] Available at netflix.com
[3] Available at movielens.org

Table 2.1: An example of an user-item matrix

	X-Men	**Cinderela**	**Batman**	**Sherlock Holmes**
Andrea	Like	Dislike		
John		Like		Like
Lucas	Like		Like	Dislike
Maria	Like	Like		?

In our problem, the active user will be the one that is requesting the recommendation, while items will be higher-education programs and its features will be skills acquired by alumni during the programs.

The main categories for recommender systems nowadays are [AT05]:

- Content-based recommendations: Recommendations given to the user are based on items that he has liked in the past.
- Collaborative recommendations: Recommendations given to the user are based on items liked by people who have similar preferences and tastes.
- Hybrid approaches: Combination of content-based and collaborative recommendations.

The problem presented here fits into the collaborative category. However, a brief overview of the other two is also provided.

2.1 Content-based Methods

In content-based methods, recommendations are made by identifying which items are similar to the ones preferred by the active user in the past. More formally, it can be defined as the estimation of the utility $u(c,s)$ of an item s for user c based on the utilities $u(c,s_i)$ that the user c has given to items $s_i \in S$ that are similar to the current item s [AT05].

A similarity measure is a distance measure and is used as weight. The more similar two users c and c' are, the more weight c' ratings will have on making recommendations for user c [AT05]. The same applies for items.

These methods make recommendations by analysing the content of textual information and finding patterns within it. To make predictions, they rely on the user and items features that are extracted from textual information [SK09].

Some of the limitations of these methods are: limited content analysis (the features associated with the objects recommended have to be in a form that can be automatically parsed by a computer such as text, or assigned manually), overspecialisation (user c will only be recommended items similar to those he liked in the past) and the new user problem (a user c has to rate a significant number of items before the system understands his preferences). More details can be found in [AT05].

2.2 Collaborative Methods

Collaborative recommender systems, also know as collaborative filtering systems, make recommendations to a new user based on the known preferences of a similar group of users. The fundamental assumption of such systems is that if two users c_i and c_j rate n items similarly, it is reasonable to assume that they will act similarly towards other items [SK09]. More formally, the utility $u(c,s)$ of item s for the active user c is estimated based on the utilities $u(c_j,s)$ assigned to item s by those users $c_j \in C$ that have similar preferences to the active user c [AT05].

Collaborative methods have the ability to filter any type of content (such as text, artwork or music), since the recommendation process is based on data from other users, whereas content-based methods do not have that ability since they rely solely on the history of the user for each type of content. The main advantage of collaborative methods over content-based methods is perhaps the fact that they do not depend on error-prone machine analysis of content [HKR00].

In the problem we address, the active user chooses its preferred job which has a set of skills associated. That set of skills can be achieved by education programs. Thereby, the recommender system finds users that have similar preferences to the active user, which are expressed in the form of skills. After that, only the most common education programs among users with same skill preferences are recommended.

According to [SK09], algorithms that perform collaborative filtering fit into one of three main classes: memory-based, model-based or hybrid, which will be described next.

2.2.1 Memory-based

Memory-based algorithms make their predictions based on the entire collection of already rated items by the users. Every user is considered to be part of a group of people with similar interests and preferences, that are often referred to the *nearest-neighbours* of the active user.

A prevalent memory-based collaborative filtering algorithm is neighbourhood-based algorithm, which comprises the following steps [SK09]:

1. Calculate the similarity (or weight) between two users or items;

2. If it is a top-N recommendation task, identify the k most similar users or items;

3. Produce a prediction for the active user.

The algorithm steps are described in the next section.

The high complexity computation of the nearest-neighbours can be reduced by performing clustering before and then compute nearest-neighbours search only among the users that belong to the nearest cluster, or use the cluster centroids [DK04].

In this way, before computing the nearest-neighbours, we group job positions into broader categories according to their skills.

2.2.1.1 Similarity Computation

Similarity computation is a critical step in memory-based collaborative filtering algorithms and can be calculated between items or users. In the case of item-based algorithms, only users who have rated both items i and j are considered. After that, similarity computation is applied to determine the similarity $w_{i,j}$ [SK09]. For user-based algorithms, the similarity $w_{u,v}$ is calculated between users u and v who have both rated the same items [SK09].

There are several methods to compute the similarity between users or items. The application of the correct measures results in a more accurate data analysis. There is not a similarity measure that works well for all implementations, so various have to be experimented in order to identify the one that fits best to the specific problem and context.

[SSSHT10] presents seventy six binary similarity and distance measures that were collected and analysed. Hierarchical clustering was performed to estimate the similarity between measures.

Two of the most used similarity measures are described next; other measures can be found in [SSSHT10].

Correlation-Based Similarity – The similarity is measured by computing the *Pearson correlation* or other correlation-based similarities. Pearson correlation measures the extent to which two variables linearly correlate with each other [RIS+94].

Vector Cosine-Based Similarity – The similarity between two items is calculated by treating each item as a vector of ratings and then computing the cosine angle formed by the vectors [SM86].

Since similarity computation is a critical step, there has been active research on improving these measures. [JSB11] presents a probabilistic definition of item similarity. A co-occurrence between two items i and j is defined as the number of users that have liked both items. Item similarity is defined as the ratio between the actual number of co-occurrences and the number of co-occurrences that would happen if user choices were random. It then applies the user's usage history to the item similarity matrix. It presents experiences with real-world usage data form different data sets to access the quality of the algorithm. Its quality was measured against several well known algorithms and it tied for first place. In [CAL] the accuracy of the Dice similarity measure is improved by processing existing correlations between the characterising features.

2.2.1.2 Top-N Recommendations

The purpose of the top-N recommendation algorithm is to recommend a set of N top-ranked items that will be interesting for the active user. It tries to discover relations between different user or items and uses those relationships to compute the recommendations.

Top-N recommendations can also be achieved with model-based collaborative filtering approaches such as association rule mining based models [SK09]. [DK04] proposes a item and model-based algorithm that first, computes the similarity between the items, and then combines these similarities in order to compute the similarity between a set of items and a candidate recommender item.

User-based Top-N Recommendation Algorithm – Typically, these algorithms comprise the following steps [SK09]:

1. Identify the k most similar users to the active user using a similarity measure;

2. Identify the set of items $N \subset S$ liked by those users and their frequency;

3. Recommend the top-N most frequent items in N that the active user has not rated yet.

User-based top-N recommendation algorithms have some limitations regarding scalability and real-time performance.

Item-based Top-N Recommendation Algorithm – Item-based algorithms address the scalability problem of the user-based ones. Usually they comprise the following steps [SK09]:

1. Compute the k most similar items for each item in the set of items S using a similarity measure;

2. Identify the set of items N constituted by the union of the k most similar items and removing from N the set of items U that the user already liked;

3. Calculate the similarities between items in N and U;

4. Sort S by decreasing order of similarity to obtain the list of Top-N recommended items.

We are interested in providing the active user with top-n recommendations rather than a single recommendation, allowing him to choose the option that is most suitable for him to select and achieve his desired job.

2.2.1.3 Prediction and Recommendation Computation

In this step the predictions or recommendations are obtained. In the neighbourhood-based collaborative filtering algorithm, a subset of the nearest neighbours of the active user is chosen based on the similarities with the user. After that, a weighted aggregate of the neighbours ratings is generated in order to make predictions for the active user [cKBR99].

2.2.2 Model-based

Model-based approaches have been developed to overcome some of the shortcomings of memory-based approaches and to achieve better performance. These approaches use the collection of already rated items to learn a model, which can be a data mining or machine learning algorithm, and is used to predict item's rating. Among the most well-known techniques, we can highlight Bayesian belief net models, clustering models and latent semantic models [SK09].

2.2.2.1 Bayesian Belief Net Collaborative Filtering Algorithms

Each item is modelled as a node having states corresponding to the ratings given by users to that item. Then, a network is build on these nodes in such way that each node has a set of parent nodes that are the best predictors for the child's rating [DK04].

2.2.2.2 Clustering Collaborative Filtering Algorithms

The goal of clustering is to discover the natural grouping of a set of items [Jai10]. Items in the same cluster have a high level of similarity among each other and a low level of similarity with other clusters items. Since data tends to be sparse because users do not rate every item in S, much more accurate predictions can be made by grouping users into clusters and also grouping items into clusters [UF98].

Clustering methods can be classified into two groups: partitional or hierarchical. Partitional methods produce only one partition of the items, while hierarchical ones produce a nested series of partitions [JMF99].

Usually clustering is just an intermediate step and its results are used for analysis or classification tasks [SK09]. Clustering Collaborative Filtering can be applied in different ways. For example, [UF98] presents a formal statistical model of collaborative filtering and then compares it against different algorithms for estimating the model parameters. Users and items are each divided into clusters and there are link probabilities between these users and items in different clusters. [CH01] uses clustering, followed by a memory-based collaborative filtering algorithm to make predictions within each created cluster.

2.2.2.3 Latent Semantic Collaborative Filtering Models

Latent Semantic Collaborative Filtering uses a statistical modelling technique that introduces latent class variables in order to discover user communities and interest profiles. Latent variables are variables inferred from variables that are measured. The main advantage of this technique in comparison with memory-based methods is higher accuracy and scalability [SK09].

2.2.3 Challenges of Collaborative Filtering

Like content-based systems, collaborative filter systems also have their own limitations and challenges. Some of the issues include the new user, new item, gray sheep (users that do not have a consistent opinion towards linking items), shilling attacks (users that give many good ratings to their own materials and bad ratings to others) and scalability (when the number of users and items grows tremendously, computational resources will go beyond acceptable levels) problems. These ones will not need to be addressed in the problem described here due to the static nature of the data. For more details about these problems and how they can be overcome see [AT05] and [SK09].

On the other hand, the issues that can emerge in the recommender system for the problem described here are explained next.

2.2.3.1 Data Sparsity

The size of the previously given ratings in comparison with the size of the ratings that need to be predicted is very small. An effective recommender system needs to make good recommendations even when a small number of users or ratings is available.

Several ways to overcome this problem have been researched. For example, [Paz99] uses demographic user data such as gender, age and education, when calculating user similarity. Another approach is to use Singular Value Decomposition [SKKR00], which is a dimensionality reduction technique for sparse rating matrices.

Some model-based collaborative filtering algorithms address the sparsity problem by providing more accurate predictions in such situations. Some of the techniques that address this problem are: association retrieval technique [HCZ04]; Maximum margin matrix factorisations [RS05]; and multiple imputation-based collaborative filtering approaches [NKH04].

2.2.3.2 Synonymy

According to [SK09], "synonymy refers to the tendency of a number of the same or very similar items to have different names or entries". This represents a challenge because most recommender systems are unable to identify this relation between different items.

One of the methods developed to solve this problem was Latent Semantic Indexing, which is a Singular Value Decomposition technique that takes a matrix of term-document association data and constructs a semantic space with terms and documents. The distance terms represents its closeness. However, this technique only solves part of the problem and the other part - words having more than one meaning, remains unsolved [SK09].

2.2.3.3 Explainability

Explaining the recommendations is viewed as an important aspect from the users perspective. Users are more likely to accept a recommendation that has an explanation than one that does not provide one, because the explanation provides more transparency and exposes the reasoning and data behind certain recommendation [HKR00]. Another advantage of providing explanations is that they help identify the likelihood of errors in the recommendation, such as model/process and data errors.

[HKR00] describes experiments that were performed with the intent of discovering, among other issues, what models and techniques are effective for supporting explanation in a collaborative filtering system, and the conclusion is that histograms of the neighbour's ratings, past performance and similarity to other items in the active user's profile are the most compelling methods to explain the reasoning behind a certain recommendation.

2.3 Hybrid Approaches

A hybrid recommender system can incorporate the advantages of both content-based and collaborative approaches, and, at the same time, prevent some of their disadvantages.

There are four key ways to do this [AT05]:

1. Implement collaborative and content-based methods independently and combine their predictions;
2. Add some content-based characteristics into a collaborative system;
3. Add some collaborative characteristics into a content-based system;
4. Build a model that incorporates both approaches, content-based and collaborative.

In [BS97] it is possible to find an distributed implementation of such a system that recommends items based on the active user preferences and the common preferences. It also explains in detail how disadvantages from both approaches are mitigated through their combination.

2.4 Evaluation of Recommendation Tasks

The quality of a recommender system can be assessed through an evaluation using metrics. The type of metrics used is critical and depends on the type of collaborative filtering application.

[GS09] highlights the importance of the evaluation metric on the evaluation of different algorithms to decide which one to use according to the domain and task of interest. Since each metric might favour different algorithms, they define the most appropriate metrics for specific tasks: recommending good items, optimising utility and predicting ratings. Finally, they empirically show how different metrics can rank two algorithms differently, showing the importance of the metric choice.

[HKTR04] classifies evaluation metrics into the following categories:

- Predictive accuracy metrics;
- Classification accuracy metrics;
- Rank accuracy metrics.

2.4.1 Predictive accuracy metrics

Predictive accuracy metrics measure how close the recommendations are to the true ratings given by users [HKTR04].

Two important predictive accuracy metrics are:

Mean Absolute Error - computes the average of the absolute difference between the predictions and true ratings. The lower it is, the better the prediction was.

Root Mean Absolute Error - amplifies he contributions of the absolute errors between true ratings and the predictions made.

It is important to keep in mind that the most accurate recommendations might not be the most useful ones.

2.4.2 Classification accuracy metrics

Classification accuracy metrics measure how frequently a recommender makes correct decisions regarding whether an item is a good or bad recommendation [HKTR04].

Two important classification accuracy metrics are:

Precision - represents the probability that a selected item is relevant.

Recall - represents the probability that a relevant item will be selected.

Other classification metrics such as ROC Curves and Swets' A Measure can be found in [HKTR04].

The definition of *relevant* has been a cause of discussion, since from a recommender system point of view it is subjective and entirely dependent on the user [HKTR04].

2.4.3 Rank accuracy metrics

Rank accuracy metrics measure the ability of the recommender system to correctly order the items that match the order that the user would have chosen. These metrics are appropriate for domains where the user's preferences are non-binary and a ranked recommendation list will be presented to the user [HKTR04].

One important rank accuracy metric is the **Half-life Utility Metric**, which evaluates the utility of a ranked list, having in consideration that the probability of the user viewing an item decreases as its position on the list increases.

Other ranking metrics such as The NDPM Measure can be found in [HKTR04].

2.5 Conclusions

After the literature review, it is possible to identify where the higher education course recommendation system fits: it is a collaborative filtering recommender system, since the recommendations to the active user are based on the alumni preferences, expressed as their education. Within collaborative filtering algorithms, we pick a memory-based collaborative filtering algorithm, since we are interested in making predictions based on the entire collection of higher education programs. In the final system, the user will be given a list of possible higher education programs to take in order to achieve a certain position, so it will be a Top-N recommendation task. Finally, the Top-N recommendation will be a user-based one since we are interested in identifying the similar users to the active one, and then recommend the set of higher education programs that are most frequent among those users.

Chapter 3

System Overview

The main goal of the system to be designed is to make recommendations of higher education programs. The starting point will be the job position/area desired, which in the figure 3.1 is represented by the blue dashed line. To achieve that, it is necessary to infer two other relationships, that are represented by grey dashed lines in figure 3.1:

1. **Skills - Job Positions**: job postings have a job position and skills required for it, making possible to have job positions and skills for these jobs.
2. **Skills - Higher Education Program**: alumni have one or more higher education programs taken and a set of skills that they claim to have acquired, making possible to have higher education programs and skills they provide.

The initial idea was to have alumni feeding both the higher education and job positions using LinkedIn[1] data, however by the date of this report that is not possible to achieve. Since that is not a suitable option, data will be collected from the different sources and then combined. Through job postings will be possible to collect a set of job positions and skills associated to those jobs, and through alumni profiles it will be possible to collect a set of higher education programs and the skills they provide to their students. Section 3.2 provides more details on how this information will be obtained.

By doing this, it is possible to infer the real skills that certain higher education program gives to their students, instead of just skills it claims to give. On the other hand, the skills needed for a wide range of jobs in computer science are also achieved.

3.1 Problem

The problem described can be formulated as *How to use alumni and job posts information to recommend higher education programs appropriate to achieve a certain job?*.

[1] https://www.linkedin.com/

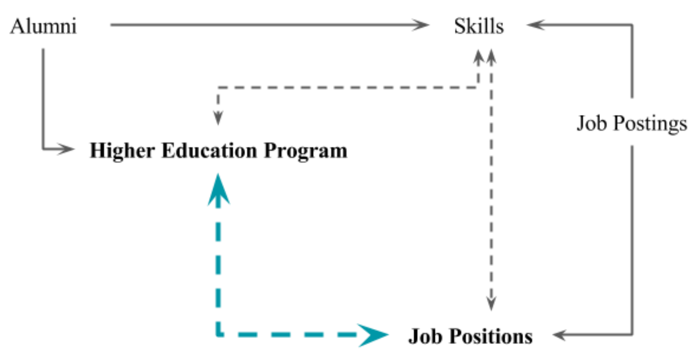

Figure 3.1: Logic architecture. Full lines represent information that we have. Dashed lines represent relationships we need to establish.

Figure 3.2 presents a general overview of the solution to this problem. It has the following steps:

1. The user chooses a computer science related career;
2. The desired career has a set of competences needed - job postings will allow to identify which they are;
3. The competences needed can be obtained by taking one or more higher education programs - alumni information will allow to do this match;
4. We will then recommend a set of higher education programs that can provide those skills to the user is recommended.

3.2 Social Networks

In order to build a data set with jobs, education and the corresponding skills, it was necessary to search for professional social networks that provide this data. A social network can be defined as "a network of social interactions and personal relationships" and "a dedicated website or other application which enables users to communicate with each other by posting information, comments, messages and images" [oxf16].

Social networking can be done for social, business purposes or both. Our focus was on the social networks for business purposes. This type of networks allows their users to establish and document networks of people they know and trust professionally.

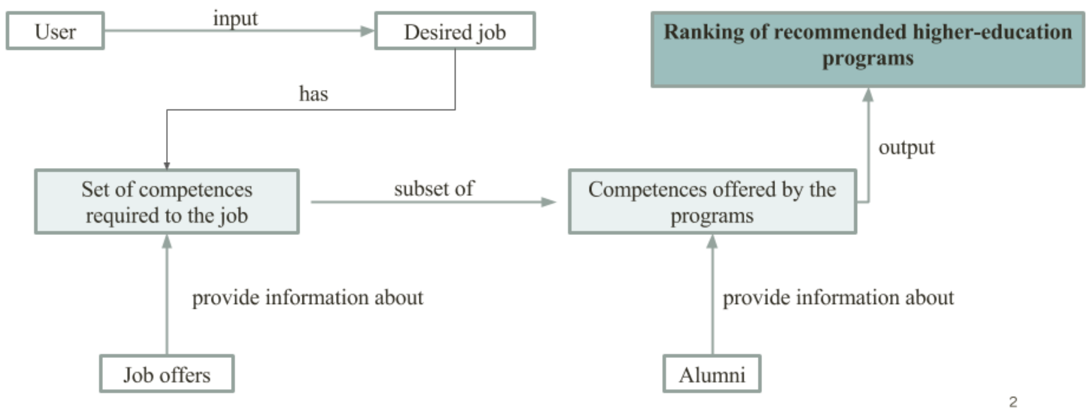
User
input
Desired job
has
Ranking of recommended higher-education programs
output
Set of competences required to the job
subset of
Competences offered by the programs
provide information about
provide information about
Job offers
Alumni
2

Figure 3.2: Solution. The input is the user's desired job. That job has a set of competences required, than can be acquired in a set of programs. The output is the ranking of programs recommended.

After an extensive search for social networks for business purposes, the following were selected for further exploration: *ITJobs*[2], *Jobvite*[3], *Landing.jobs*[4], *LinkedIn*, *Stack Overflow Careers*[5], *Upwork*[6] and *Xing*[7].

In order to extract the data from the social networks, it was necessary to explore the Application Program Interface (API) that each of them provides to assess which jobs, education and the corresponding skills would be possible to extract. In the following, we describe the explored APIs in detail.

3.2.1 ITJobs

ITJobs is an online platform were companies can post jobs and candidates can apply and get hired for those jobs.

Despite offering an endpoint for job search, each job does not have associated skills. Therefore, the use of ITJobs API was discarded.

3.2.2 Jobvite

Jobvite has the same purpose as ITJobs which was described in 3.2.1.

[2]https://www.itjobs.pt/
[3]http://www.jobvite.com/
[4]https://landing.jobs/
[5]http://careers.stackoverflow.com/
[6]https://www.upwork.com/
[7]https://www.xing.com/

It offers an endpoint for candidate search and one for job listing. The candidate search was discarded because it is not possible to know the candidate's education as well as their skills. The job listing was discarded because each job does not have skills associated. Therefore, the use of Jobvite API was discarded.

3.2.3 Landing.jobs

Landing.jobs defines itself as "a candidate-driven tech jobs marketplace" and has a more corporate employment focus. Just like ITJobs it allows the matchmaking between candidates and jobs.

Through the Landing.jobs API, is not possible to search or list candidates since it only allows to access the current user data and it does not include their education. However, it offers an endpoint for job listing where each job listed has a set of *tags* which are core competences needed for that job.

3.2.4 LinkedIn

On LinkedIn, people can post professional information about themselves such as, for example, education taken, skills and current and past job positions.

Despite having all these data about people, it does not provide an endpoint for searching people and only provides one to access the current user data. LinkedIn only provides its search API for people via its LinkedIn Partners Program; several applications were made and all of them were rejected.

After some research it was possible to conclude that it is possible to access candidate information if the user allows the access manually. Having this in consideration, the use of LinkedIn's API is on hold and in the future there is the possibility of building an application for individuals to manually accept the access to their data.

3.2.5 Stack Overflow Careers

Stack Overflow Careers presents itself as a solution for companies to search and find developers that are suitable for the job offers they have. Candidates are able to prove the knowledge trough the successful resolution of other people's questions.

It provides a paid candidate search, were it is possible to filter candidates and access their personal information such as education, current job position and skills acquired. Since the access to candidate search is paid, the use of Stack Overflow Careers for candidate search is on hold.

On the other hand, the access to job search is free and jobs can be searched using keywords and a location. Job data includes a title and a required skill set.

3.2.6 Upwork

Upwork defines itself as "an online workplace for the world – connecting clients with top freelance professionals" and is focused on freelancer workers.

It offers an endpoint for searching freelancers, and another for searching for jobs. Freelancers search was discarded due to not providing their education and employment history, although they are present in the web view of the profile. On the other hand, each job returned on the job search has a title, category and subcategory, and skills required or desired for that job. The category and subcategory represent competence or expertise areas.

3.2.7 Xing

Xing is "the largest business network in German-speaking countries".

It offers an endpoint to find people using various filters. Each person found has a set of core competences and an educational background.

It will possibly require data translation from German to English, since English is the prevailing language in the other data sources.

3.2.8 Other sources

ESCO[8] "is the multilingual classification of European Skills, Competences, Qualifications and Occupations". "It identifies and and categorises skills, competences, qualifications and occupations relevant for the EU labour market and education and training", and establishes relationships between the different concepts.

Browsed occupations in ESCO can be used as a job search starting point in the other data sources that provide job search. It can also be useful to match information from different sources through the relationships it establishes.

3.3 Data Set Composition

There are two different perspectives on the data represented in figure 3.3: jobs and education.

Each job has a title, a category, a subcategory and a set of skills associated. The category and subcategory can be viewed as an ontology. By using them it is possible to aggregate jobs on categories. Jobs with the same title are not aggregated in order to not miss the specificity of each job offer.

Each education has a degree associated as well as a set of skills that represent the competences that a certain higher education provides.

Until the date of this report, the database has 3017 jobs, 1892 skills and 13397 associations between the last two. This result was achieved by following these steps:

1. Collect jobs from Landing.jobs using the job listing;
2. Complement category and subcategory of already collected jobs using Upwork search - the information was only added if the down-case title of the job was equal between the two data sources;

[8]https://ec.europa.eu/esco/portal/home

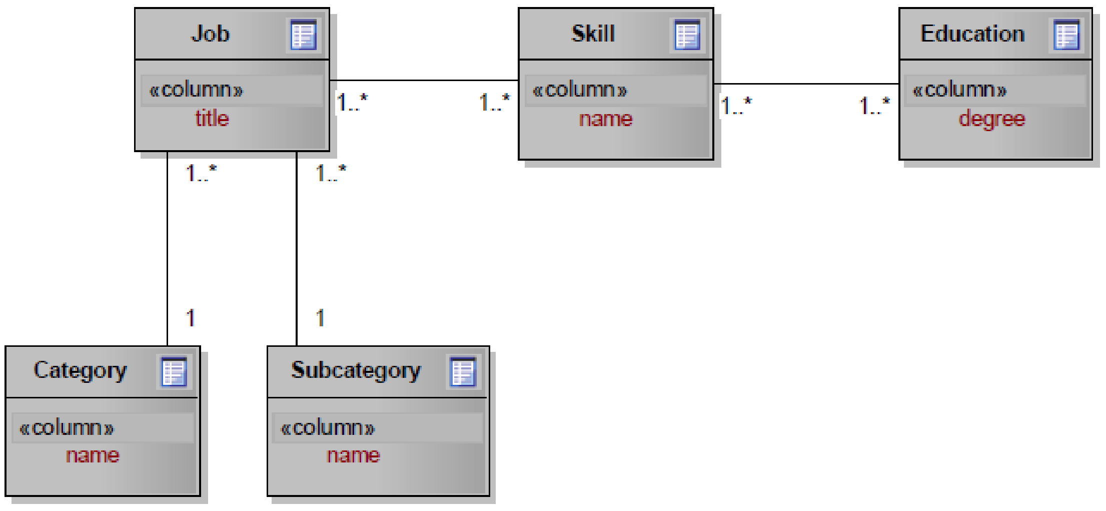

Figure 3.3: Data model

3. Collect jobs from Upwork using the job search - the sob search was made using titles of already collected jobs, as well as categories, subcategories and skills.

The data is stored in a PostgreSQL database[9].

3.3.1 Data Cleaning

The immediate step to take after collecting the data is to perform an exploratory data analysis, where the main goal is to identify and remove outliers. There is a need to do this because there is no human control on what jobs are stored or not. Some of them are not related with computer science.

The presence of outliers in the recommender system can lead to poor recommendations. To prevent this, their identification and elimination will be done in several ways:

- Compute the frequency of each skill, identify skills with less frequency and remove from jobs that have those skills.
- Analyse the job title length: titles of considerable size are usually outliers.
- Abstract rules that represent outliers and remove jobs that match those rules.

To prevent the elimination of false outliers, some examples need to be manually validated first.

3.4 Conclusions

Table 3.1 summarises the information that can be extracted from the business networks.

The data set construction is one of the most crucial steps in the implementation of the recommender system desired and has been a challenging task due to the limitations of the APIs provided

[9]http://www.postgresql.org/

Table 3.1: Data sources summary. Crosses represent data that is present in each API.

	Education and Skills	**Jobs and Skills**
Landing.jobs		x
LinkedIn	x	x
StackOverflow Careers	x	x
Upwork		x
Xing	x	

by different business networks (lack of information or required payment). We will attempt at mitigating this problem by combining data from different social networks.

It is important to reflect on the impact of collecting information from different data sources.

On one hand, there is the possibility that the skills that characterise jobs and education programs will be written differently. With the rise of business networks, people have multiple profiles and tend to focus their attention on a few of them. The information available in the other business networks might not be the most recent and accurate one.

On the other hand, different sources of information provide information that is not dependable on the individual. We can achieve more independence from specific cases by having multiple data sources.

Chapter 4

Conclusions and Future work

The goals established until the date of this report are considered to be accomplished: the literature review and some practical work.

Regarding the literature review the focus was on recommender systems, different recommender categories, different approaches to the recommendation task and the identification of characteristics of the higher education recommendation system.

The practical work focus was on the identification of viable information sources, followed by an exploration of the information that could be extracted from each information source and ending in the actual information extraction and storage. The last one is an ongoing work and it should be finished by fifteenth of March as stated in A.

4.1 Work plan

The next immediate task is to continue the collection of relevant information, which includes collecting the data, identifying possible outliers and analysing the data in order to identify some natural groupings.

After these tasks are completed, the implementation of the actual recommender system will begin and it will comprise an iterative cycle of:

1. Implementing the algorithm;
2. Experimenting a similarity measure;
3. Evaluating the solution obtain trough appropriate metrics.

When the cycle is finally finished, a more subjective evaluation of the recommender system will be necessary. It might include an evaluation by a group of people using a graphical interface build for that purpose.

Finally, one month is reserved for writing the thesis.

A graphical view of the work plan with planned start and finish dates can be found in A.

References

[AT05] G Adomavicius and a Tuzhilin. Toward the Next Generation of Recommender Systems: a Survey of the State of the Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[BS97] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[CAL] Susana B Cruz, Ana Aguiar, and Daniel E Lucani. Generation of Trimmed Similarity Lists with Applications in Electronic Program Guides. pages 1–6.

[CH01] Mark Connor and Jon Herlocker. Clustering items for collaborative filtering, 2001.

[cKBR99] Jonathan L. cker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[DK04] Mukund Deshpande and George Karypis. Recommendation Algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, 2004.

[GS09] Asela Gunawardana and Guy Shani. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *The Journal of Machine Learning Research*, 10:2935–2962, 2009.

[HCZ04] Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142, January 2004.

[HKR00] Jonathan L Herlocker, Joseph a Konstan, and John Riedl. Explaining collaborative filtering recommendations. *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250, 2000.

[HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[Jai10] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[JMF99] a. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[JSB11] Oliver Jojic, Manu Shukla, and Niranjan Bhosarekar. A probabilistic definition of item similarity. *Proceedings of the fifth ACM conference on Recommender systems - RecSys '11*, page 229, 2011.

[NKH04] Hyunju Noh, Minjung Kwak, and Ingoo Han. Improving the prediction performance of customer behavior through multiple imputation. *Intell. Data Anal.*, 8(6):563–577, 2004.

[oxf16] Oxford Dictionaries, 2016.

[Paz99] Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, December 1999.

[RIS$^+$94] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John 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.

[RS05] Jasson Dm M Rennie and Nathan Srebro. Fast Maximum Margin Matrix Factorization for Collaborative Prediction. *Proceedings of the 22Nd International Conference on Machine Learning*, pages 713–719, 2005.

[SK09] Xiaoyuan Su and Taghi M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009(Section 3):1–19, 2009.

[SKKR00] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000.

[SM86] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[SSSHT10] Choi Seung-Seok, Cha Sung-Hyuk, and Charles C Tappert. A Survey of Binary Similarity and Distance Measures. *Journal of Systemics, Cybernetics & Informatics*, 8(1):43–48, 2010.

[UF98] Lyle H. Ungar and Dean P. Foster. Clustering methods for collaborative filtering. *AAAI Workshop on Recommendation Systems*, pages 114–129, 1998.

Appendix A

Work Plan

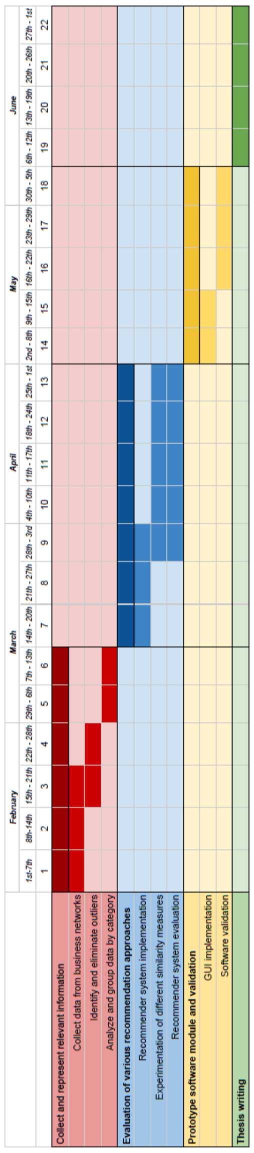

Figure A.1: Work plan