OPENSKIMR A JOB- AND LEARNING-PLATFORM

Andreas Kofler[1] and Marianne Prast[2]

[1,2]Management Center Innsbruck (MCI), Innsbruck, Austria
andreas.kofler@mci.edu, marianne.prast@mci.edu

ABSTRACT

This paper is concerned with the mathematical aspects of the development of the job- and learningplatform OPENSKIMR. The platform should enable users to be matched with jobs based on their individual skill profile and the job skill requirements. Further, once users have chosen jobs they like, possible learnings in order to better fit the job's requirements are recommended. We give a short introduction to the data model we use and show the mathematical framework we act within. Further, we present our Route Planner Algorithm and discuss its functionality.

KEYWORDS

OpenSkiMr, knowledge engineering, data mining, digitalization, e-learning, learning roadmap, matching, recommender systems

1. INTRODUCTION

OPENSKIMR (Open European Skill Match Maker) is a project funded by the European Union. The goal is to develop a matching system between users, jobs and education. In doing so, the platform should help stabilize the European labor market (especially for young people), foster lifelong learning and vocational education and enhance geographic mobility within Europe. The philosophy standing behind OPENSKIMR is the picture of a route. Users should be able to find new ways, new goals as well as get information on how to reach them.

The platform should perform the following tasks:

- *Submitting skills or dream job:* OPENSKIMR offers two different ways to submit personal information that builds the basis for the subsequent matching. The user can either sketch a personal skill profile or submit a dream job.

- *Matching the users according to their skill profile with occupations and jobs:* Skill mismatch is a complex problem which does affect not only those who are looking for a job but most of the workforce. The phenomenon of skill mismatch involves underskilling as well as overskilling. Both can undermine the long-term potential of the workforce [1].

- *Suggesting an education route*: The user receives recommendations for learnings which it might be interested in in order to better fit a job's requirements

David C. Wyld et al. (Eds) : CSITA, ISPR, ARIN, DMAP, CCSIT, AISC, SIPP, PDCTA, SOEN - 2017
pp. 95– 106, 2017. © CS & IT-CSCP 2017 DOI : 10.5121/csit.2017.70110

2. DATA MODEL

The functionality of OPENSKIMR is based on ESCO (European Skills, Competences and Occupations) catalogue. The outcome is a standardized data set with a common understanding of occupations and their related skills and knowledge, as well as the relationships between these concepts. It is part of the Europe 2020 strategy [2].

2.1. NOTATION

Let $\mathcal{S} = \{s_1, \ldots, s_n\}$ denote the set of skills and tools and $I = \{1, 2, \ldots, n\}$ the set of skills and tools indices. From a semantic point of view, it is necessary to distinguish between skills and tools. Skills such as *programming* or *data mining* refer to the ability to apply knowledge and use know-how in a certain area, while so-called skill applications or tools denote, for example, concrete programming languages such as C++ or Java. Therefore, the set $\mathcal{S}$ is given as $\mathcal{S} = \mathcal{S}_{\text{skills}} \cup \mathcal{S}_{\text{tools}}$ with $n_{\text{skills}} + n_{\text{tools}} = n$ and $I = I_{\text{skills}} \cup I_{\text{tools}}$.

By the italic letter O we denote the set of occupations with cardinality N_O. The *ESCO catalogue* delivers information about occupations and their related skills. In particular, it is stated which skills are required by the occupations. Therefore, we can identify each occupation $o \in O$ with its binary vector representation $\mathbf{o}$, where $o_i = 1$ denotes that the skill or tool s_i is required and $o_i = 0$ denotes that the skill s_i is not necessary for the occupation o. Note that we define all occupations only by skills and not by tools, i.e. $o_i \in \{0, 1\}$ for $i \in I_{\text{skills}}$ and $o_i = 0$ for $i \in I_{\text{tools}}$.

The italic letters $\mathcal{U}$ and $\mathcal{V}$ denote the set of users and jobs on the platform. Again, we denote by $N_{\mathcal{U}}$ and $N_{\mathcal{V}}$ the cardinality of the respective sets. Note that, while the sets $\mathcal{U}$ and $\mathcal{V}$ are sets whose cardinality and elements vary in time when users and jobs are added to or deleted from the platform or user profiles change due to qualification acquirements, the cardinality and the elements of the set O are constant over time.

Finally, let $u \in \mathcal{U}$ denote a user and let $\mathbf{u}$ denote its respective vector representation where each entry u_i of the vector corresponds to the level of knowledge of the user u with respect to the skill or tool s_i. Analogously, a job is denoted by $v \in \mathcal{V}$ with vector representation $\mathbf{v}$ where each entry v_i corresponds to the level of knowledge with respect to the skill or tool s_i required for the job. A user u builds up its profile by rating a skill or a tool s_i with a score from 1 (low) to 5 (expert). If a user does not deliver any information about its knowledge with respect to the skill s_i than we set $u_i = 0$. By $R = \{0, 1, 2, 3, 4, 5\}$ we denote the set of possible skill or tool ratings for jobs and users. The choice of the set R relies both on usability reasons as well as on the outlook of a possbile implementation of recommender systems where we might benefit from the exhaustive research that has been conducted in the last years on the basis of the Netflix data set [3]. Recommender systems might be applied to recommend users skills they might be interested to rate.

For a user $u \in \mathcal{U}$ we denote by $I_u \subset I$ the set of skills and tools indices it has rated. For a job $v \in \mathcal{V}$ and an occupation $o \in O$ the sets $I_v, I_o \subset I$ are analogously defined and determine the job's and occupation's skills requirements, respectively.

As the total number of skills and tools is expected to be quite high ($n > 12000$) we can consider users as well as jobs and occupations to be represented by sparse vectors in a n-dimensional space with $\mathbf{u}, \mathbf{v} \in R^n$ and $\mathbf{o} \in \{0,1\}^n$.

3. SKILLS AND TOOLS ASSESSMENT

Since we base the matching of users and job profiles on the skills and tools required, the first problem we are faced with is to offer the user an accessible way to build its profile by rating some skills and tools.

On the one hand, a user may have a quite clear idea of the area it wants to work in and therefore might rate skills which are related to certain occupations. On the other hand, a user may not have a specific choice on the area but would rather like to get an overview of jobs it would fit to. In this case the user might be interested in rating its skills by choosing them from some skills clusters. When a user rates its skills by browsing occupations, it finds tools when they are related to skills. For example, when a user states that it has knowledges about programming, the user has the possibility to rate the programmning languages C++ or Java. Recruiters are treated the same as users when they create a job advertisement.

3.2. CLUSTERING OF SKILLS AND OCCUPATIONS

We build the occupations clusters as well as the skills clusters from which users may choose and rate their skills using an agglomerative hierarchical clustering algorithm.

An Agglomerative hierarchical clustering algorithm requires an initial dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_O}\}$, where each data point $\mathbf{x}_k$ is considered to be a single cluster. Then, the algorithm repeats merging the two closest clusters according to some similarity measure between clusters. The procedure is repeated until all data points are contained in one single cluster. The output of the algorithm is a sequence of nested partitions of the dataset.

Let $\mathbf{O}$ denote the binary $N_O \times n$ matrix whose rows $\mathbf{o}_1, \ldots, \mathbf{o}_{N_O}$ correspond to the binary occupations vectors. Analogously, the rows of the transposed matrix $\mathbf{O}^T$ contain the binary skills vectors whose entries define by which occupations the skills are required.

Let C_i and C_j denote two non-empty, non-overlapping clusters of data points which might represent two clusters of occupations or skills. We merge occupations clusters as well as skill clusters according to the complete link method, i.e. the distance measure $D(\cdot,\cdot)$ between the clusters C_i and C_j is given by

$$D(C_i, C_j) = \max_{\mathbf{x} \in C_i,\ \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y}) \tag{1}$$

where $d(\cdot,\cdot)$ is given by the cosine distance measure

$$d(\mathbf{x}, \mathbf{y}) = 1 - \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|} \tag{2}$$

Note that, though $d(\cdot,\cdot)$ is not a proper distance measure since it does not fulfill the triangle inequality property, the similarity measure given by $c(\mathbf{x},\mathbf{y}) = 1 - d(\mathbf{x},\mathbf{y})$ is a commonly used measure to compute the similarity between points, especially when the data one handles can be represented as binary data, e.g. text documents [4]. Furthermore, the computation of the similarity measure between all the occupations vectors and the skills vectors is extremely efficient as it reduces to the multiplication $\mathbf{O}_{\text{norm}}\mathbf{O}_{\text{norm}}^T$ and $\mathbf{O}_{\text{norm}}^T\mathbf{O}_{\text{norm}}$, where $\mathbf{O}$ is sparse.

Since we are going to use the cosine similarity later within the paper, we denote the cosine similarity measure by $c(\mathbf{x},\mathbf{y}) = 1 - d(\mathbf{x},\mathbf{y})$. We apply the clustering algorithm to the set of occupations with initial set $\mathcal{D}_{\text{occupations}} = \{\mathbf{o}_1,\ldots,\mathbf{o}_{N_O}\} \in \{0,1\}^n$ and to the set of skills $\mathcal{D}_{\text{skills}} = \{\mathbf{o}_1^T,\ldots,\mathbf{o}_{n_{\text{skills}}}^T\} \in \{0,1\}^{N_O}$. Be aware that tools are not related to occupations and therefore are not clustered. With the help of the constructed set of clusters we can decide how we group the data and approximately how large the clusters shall be in order to guarantee best usability. We use the output of the clustering algorithm as an initial suggestion to group occupations and skills. Still, the correctness of the clustering from a semantic point of view is checked by hand and the clusters are given an appropriate name.

4. FUNCTIONALITY OF OPENSKIMR

Within the mathematical framework just presented, the process of matching a user u and a job v corresponds to the calculation of a similarity measure between their vectors $\mathbf{u}$ and $\mathbf{v}$ which expresses how the user's skills profile satisfies the job's requirements.

4. 1 MATCHING FUNCTION

In order to measure how good the user u fits the job v, we are looking for a function $r_{\mathbf{v}} : R^n \to [0,1]$ for each job $v \in \mathcal{V}$ which is dependent on the job's requirements given by the vector $\mathbf{v}$. Thereby, $r_{\mathbf{v}}(\mathbf{u}) = 0$ should denote a complete mismatch of the user u with respect to the job v and $r_{\mathbf{v}}(\mathbf{u}) = 1$ should indicate that the user u would be the perfect candidate.

As often used for job advertisements, a job v might require some skills and tools to be essential in order to successfully perform the job related tasks and others to be optional. Therefore, let us assume that for every job $v \in \mathcal{V}$ we are given information about a skill or a tool to be essential or optional and let the sets $I_{v,\text{essential}} \subset I$ and $I_{v,\text{optional}} \subset I$ denote the indices of the essential and the optional skills and tools, respectively. When a job profile is created and a skill or a tool is declared to be only optional for the job v, recruiters are not able to rate the skill s_i, i.e. $v_i \in \{0,1\}$ for $i \in I_{v,\text{optional}}$ which means that the type of information is binary as for occupations.

Let $s : R^2 \to [0,1]$ be a function with $s(x,y) = 1$ whenever $x = y$ and $s(x,y) < 1$ otherwise. Then, for a fix index set $\varnothing \neq J \subset I$ we define the function $q_{s,J} : R^n \times R^n \to [0,1]$

$$q_{s,J}(\mathbf{x},\mathbf{y}) = \frac{1}{|J|}\sum_{i\in J} \text{sign}(x_i)\, s(x_i,y_i). \tag{3}$$

Finally, for a given job v with vector representation $\mathbf{v}$ and related information about which skills and tools are essential and which are optional, we define the matching function $r_{\mathbf{v}}$ by

$$r_{\mathbf{v},\lambda_1,\lambda_2}(\mathbf{u}) = \lambda_1 \left(q_{s,I_{v,\text{essential}}}(\mathbf{u},\mathbf{v}) \right) + \lambda_2 \left(\frac{|I_u \cap I_{v,\text{optional}}|}{|I_{v,\text{optional}}|} \right) \tag{4}$$

with parameters $\lambda_1, \lambda_2 > 0$ with $\lambda_1 + \lambda_2 = 1$. Here, we assumed that $I_{v,\text{optional}} \neq \varnothing$. Otherwise, the second term is set to zero and $\lambda_1 = 1$. We measure the matching with respect to the essential and optional requirements separately and use the parameters λ_1 and λ_2 to weight them differently. Typically, a parameter setting with $\lambda_1 > \lambda_2$ will be chosen in order to give the matching score regarding the essential skills and tools higher importance.

4.2. MATCHING USERS WITH JOBS AND OCCUPATIONS

Once a user u has completed the skills rating process, it can search for jobs or occupations it would fit to. Since an occupation vector $\mathbf{o}$ is a binary vector, we define the vector $\bar{\mathbf{o}}$ as a representative average of the jobs related to the occupation $o \in O$. As we do not define an occupation by concrete tools as they are rather jobs related than occupation related, we restrict the calculation of the mean to the indices which refer to the skills. For an index set $J \subset I$ we define the restriction of a vector $\mathbf{x} \in R^n$ to the entries with indices in the set J by

$$\mathbf{x}_J = (x_{j_1}, \ldots, x_{j_{n_J}})^T \in R^{n_J} \tag{5}$$

where $n_J = |J|$ denotes the cardinality of the index set J. Then, we calculate the restriction of the vector $\bar{\mathbf{o}}$ to the entries indexed by the elements of I_o by

$$\bar{\mathbf{o}}_{I_o} = \left[\frac{1}{|\mathcal{N}(\mathbf{o})|} \sum_{\mathbf{v} \in \mathcal{N}(\mathbf{o})} \mathbf{v}_{I_o} \right] \tag{6}$$

where $[\,\cdot\,]$ denotes the function which rounds each entry of the vector to its nearest integer. The set $\mathcal{N}(\mathbf{o})$ denotes the set of job vectors whose binary vector representations are most similar to $\mathbf{o}$ according to the cosine similarity measure $c(\cdot,\cdot)$, i.e

$$\mathcal{N}(\mathbf{o}) = \left\{ \mathbf{v} \;\middle|\; \mathbf{o} = \underset{\tilde{\mathbf{o}} \in \mathcal{D}_{\text{occupations}}}{\arg\max}\; c\left(\tilde{\mathbf{o}}_{I_{\tilde{o}}}, \text{Sign}(\mathbf{v}_{I_{\tilde{o}}})\right) \right\}. \tag{7}$$

where, in this context, the function Sign shall denote the componentwise application of the signum function. For all $i \notin I_o$ it holds $\bar{o}_i = 0$. Let $\bar{O}$ denote the set of the average of the jobs whose vector representations are calculated as in 6. Then, for each $\bar{o} \in \bar{O}$ we can match users with occupations by calculating $r_{\bar{\mathbf{o}}}(\mathbf{u})$. Note that in the calculation no distinction is made between the set $I_{\bar{o},\text{essential}}$ and $I_{\bar{o},\text{optional}}$ since occupations do not distinguish between essential and optional skills. Therefore, the matching of a user u with an occupation described by $\bar{\mathbf{o}}$ reduces to calculating

$$r_{\bar{\mathbf{o}}}(\mathbf{u}) = q_{s,I_{\bar{o}}}(\mathbf{u}, \bar{\mathbf{o}}). \tag{8}$$

We can calculate $r_{\bar{o}}(\mathbf{u})$ for all occupations $\bar{o} \in \bar{O}$ and recommend the user u the K_1 best occupations where the matching scores are the highest. This recommendation only serves as an initial suggestion in order to show users which occupations they fit to. Then, the user is able to select one of the shown occupations. For the chosen occupation we compute the matching scores $r_{\mathbf{v},\lambda_1,\lambda_2}(\mathbf{u})$ for all jobs $v \in \mathcal{N}(\bar{o})$ and show the user u the K_2 jobs with the highest matching score.

4.3. ROUTE PLANNER ALGORITHM

Once a user u has chosen a concrete job v it is able to see its matching score $r_{\mathbf{v}}(\mathbf{u})$. As a next step, we would like to recommend the user content and learnings which increase its level of knowledge regarding several skills and tools required by the job.

4.3.1. DEFINITION OF A LEARNING

Formally speaking, a learning or a training transforms a user u into a user $\tilde{u}$ with different skills profile. Due to the type of learning or training the user u attends, different type of knowledge is gained. This is expressed by the fact that the user's vector representation $\mathbf{u}$ turns into $\tilde{\mathbf{u}}$ where the entries which indeed change their value are the ones who denote the skills indices of the skills which the learning or training is focused on.

Note that, due to the requirements we set for a matching function $r_{\mathbf{v}}$ for a job $v \in \mathcal{V}$, there is no guarantee that a learning actually increases the matching score of a user u with respect to the job v. Consider the case, for example, that a job v denotes a junior software developer job role which only requires moderate qualifications for certain skills. If a learning or a training turns the user into an expert with respect to those skills it will be overqualified and get a lower contribute for the rating regarding those skills or tools. We assume that in general a learning might cover several topics and therefore might increase your knowledge in more than one single skill or tool.

We denote a learning or a training by the capital letter L and the set of learnings and trainings by $\mathcal{L}$.

Let $\varnothing \neq \mathcal{S}_L \subset \mathcal{S}$ be a subset of skills or tools with indices in an index set I_L and $R_L := R_{L,1} \times R_{L,2} \subset R^2$ be a subset of rating pairs with

$$\forall\, i \in I_L : \ \forall\, \big(r_1(i), r_2(i)\big) \in R_L : r_1(i) < r_2(i). \tag{9}$$

The set $\mathcal{S}_L$ is the set of skills which are related to the learning L and the set R_L can be seen as a set of rating pairs where the first rating denotes the minimal knowledge a user should have with respect to a certain skill in order to be able to participate in the course and the second entry denotes the level of knowledge a user reaches when it succesfully completes the learning. Given the two sets I_L and R_L we define a learning L by the set

$$L = I_L \times R_L. \tag{10}$$

Therefore, the learning L can intuitevely be seen as a set of triples $\big(i, r_1(i), r_2(i)\big)$ where i is the index of the skill or tool in which a user increases its knowledge from level $r_1(i)$ to level $r_2(i)$ by attending the learning.

Further, for each learning L we define a learning function

$$\begin{aligned} l_L \;:\; R^n &\longrightarrow R^n, \\ \mathbf{u} &\longmapsto \tilde{\mathbf{u}} = \begin{cases} \mathbf{u}, & \exists\, i \in I_L : u_i < r_1(i), \\ t_L(\mathbf{u}), & \text{else} \end{cases} \end{aligned} \tag{11}$$

where $t_L(\mathbf{u}) = \Big(t_L^{(1)}(u_1), \ldots, t_L^{(n)}(u_n)\Big)$ and

$$t_L^{(i)}(u_i) = \begin{cases} r_2(i), & i \in I_L,\ u_i \leq r_2(i) \ \text{ for } \ r_2(i) \in R_{L,2}, \\ u_i, & \text{else}\ . \end{cases} \tag{12}$$

The function l_L acts on the vector representation $\mathbf{u}$ of the user and maps it to a vector $\tilde{\mathbf{u}}$. Its entries which correspond to the skills and tools which are related to the learning L are updated if the user is qualified to attend the learning. Entries which correspond to skills and tools not related to the learning L are left unchanged.

4.3.2. CONSTRUCTION OF A LEARNING ROUTE

The function l_L describes how the vector representation of a user profile is updated when a learning or a training L is successfully completed by the user. Of course, a user might be interested in attending more than one learning in order to increase its matching score with respect to a job v.

Therefore, we define a learning route as a finite sequence of learnings $(L_q)_{q=1}^N$ with $L_q \in \mathcal{L}$ and $\mathcal{L}^N$ as the set of all sequences of learnings of length N. Let $N_{\mathcal{L}}$ denote the number of learnings and trainings. The best updated profile a user u with respect to a job v is given by

$$\mathbf{u}_{\mathbf{v}}^{*} = \underset{N \in \{1,\ldots,N_{\mathcal{L}}\}}{\arg\max} \ \underset{(L_q)_{q=1}^N \in \mathcal{L}^N}{\arg\max}\, r_{\mathbf{v}}\Big(\big(l_{L_1} \circ \ldots \circ l_{L_N}\big)(\mathbf{u})\Big). \tag{13}$$

Of course, the calculation of $\mathbf{u}_{\mathbf{v}}^{*}$, as just defined, is a computationally too expensive is not applicable. We now discuss our approach in order to still be able to recommend a the user a proper learning route.

We formulate the problem of finding an optimal learning route for a user u with respect to a job v by finding the shortest path of a graph $G = (V, E)$ with initial node u and destination node $\tilde{u}$ for some node $\tilde{u}$ with $r_{\mathbf{v}}(\tilde{\mathbf{u}})$ as close as possible to one.

We start with a graph $G = (V, E)$ with $V = \{u\}$ and $E = \varnothing$ where u represents the initial user. Let l_L be a learning function for a given learning $L \subset \mathcal{L}$ and let u_L denote the user after it has successfully attended the learning L, i.e. $\mathbf{u}_L = l_L(\mathbf{u})$. Then, if $r_{\mathbf{v}}\big(l_L(\mathbf{u})\big) > (1+\beta) r_{\mathbf{v}}(\mathbf{u})$ for a given matching function $r_{\mathbf{v}}$ and $\beta \geq 0$, we add the edge $e_{u,u_L} = (u, u_L)$ to the set of edges E and we add the node u_L to the set V. This step can now be repated by applying further learnings to the user profile u. After having added additional nodes and edges to the respective sets E and V we repeat the procedure by applying learnings to the nodes which represent already updated user profiles.

Algorithm (2) describes the construction of the graph. Note that we might weight an edge (u, u_L) using any numerical attributes related to the learning L where there exists a order relation such as

Algorithm 1 Route Planner Algorithm

Require: $\mathbf{u}, \mathbf{v}, r_{\mathbf{v},\lambda_1,\lambda_2}, \mathcal{L}, l_L, d_{\max}, \alpha_1, \alpha_2, \beta$

$m = 0$
$V_{\text{seen}} = \varnothing$
$V^{(m)} = \varnothing$
$V^{(m+1)} = \{u\}$,
$E = \varnothing$,
Pre-process the learnings with $\alpha_1, \alpha_2 > 0$
$\mathcal{L}_{\alpha_1} = \{L \in \mathcal{L} \mid c((\mathbf{x}_L, H_0(\mathbf{v}-\mathbf{u})) \geq \alpha_1\}$
$\mathcal{L}_{\alpha_2} = \{L \in \mathcal{L} \mid 1 - p_{\mathbf{x}_{\mathbf{g}_{L,2}}}(\mathbf{v}) \geq \alpha_2\}$
$\mathcal{L}_{\alpha_1,\alpha_2} \leftarrow \mathcal{L}_{\alpha_1} \cap \mathcal{L}_{\alpha_2}$
while $V^{(m+1)} \backslash V^{(m)} \neq \varnothing \ \wedge \ m < d_{\max}$ **do**
  $V^{(m)} \leftarrow V^{(m+1)}$
  $V_{\text{temporary}} \leftarrow V^{(m+1)}$
  for $\tilde{u} \in V^{(m+1))} \backslash V_{\text{seen}}$ **do**
    for $L \in \mathcal{L}_{\alpha_1,\alpha_2}$ **do**
      $\tilde{\mathbf{u}}_L \leftarrow l_L(\tilde{\mathbf{u}})$
      if $r_{\mathbf{v},\lambda_1,\lambda_2}(\tilde{\mathbf{u}}_L) > (1+\beta) r_{\mathbf{v},\lambda_1,\lambda_2}(\tilde{\mathbf{u}})$ **then**
        $V^{(m+1)} \leftarrow V^{(m+1)} \cup \{\tilde{u}_L\}$
        $V_{\text{temporary}} \leftarrow V_{\text{temporary}} \cup \{\tilde{u}_L\}$
        $E \leftarrow E \cup \{(u, \tilde{u}_L)\}$
      end if
    end for
    $V_{\text{seen}} \leftarrow V_{\text{seen}} \cup \{\tilde{u}_L\}$
  end for
  $V^{(m+1)} \leftarrow V_{\text{temporary}}$
  $m \leftarrow m + 1$
end while
$V \leftarrow V^{(m)}$
$G \leftarrow (V, E)$
$\mathbf{u}^*_{\mathbf{v},d_{\max}} \leftarrow \arg\max_{\tilde{\mathbf{u}} \in V} r_{\mathbf{v}}(\tilde{\mathbf{u}})$
Apply Dijkstra Algorithm with initial node u and destination node $u^*_{v,d_{\max}}$

Ensure: $G = (V, E)$, shortest path and $r_{\mathbf{v}}(\mathbf{u}^*_{\mathbf{v},d_{\max}})$

price of the learning or the duration of the learning. Then, applying Dijkstra's algorithm [5] we find the shortest path from the initial node u to the node which has the best matching score with respect to the job v.

Also note that we need to pre-process the learnings we take into consideration for the calculation as an initial step. This step is crucial in order to save computational time.

Let the vector $\mathbf{x}_L \in \{0,1\}^n$ be the binary learning vector with $x_{L,i} = 1$ if $i \in I_L$ and $x_{L,i} = 0$ otherwise. Then, let $H_0(\mathbf{v}-\mathbf{u})$ denote the binary vector whose i-th component is equal to one if the user has a knowledge gap with respect to the i-th skill, where H_0 denotes the componentwise application of the Heaviside function with $H_0(\mathbf{x}) = \big(h_0(x_1),\dots,h_0(x_n)\big)$ and $h_0(x) = 1$ for $x > 0$ and $h_0(x) = 0$ for $x \leq 0$. By applying only learnings where the cosine of the angle between a learning vector $\mathbf{x}_L$ and the user's skill gaps vector $H_0(\mathbf{v}-\mathbf{u})$ exceeds a chosen threshold $\alpha_1 > 0$, we avoid considering learnings which are not related to the skills or tools where the user has a gap. By doing so, we mainly focus on the learnings which cover topics needed for the job v. Further, the computation of the angle between the learnings and the skill gap vector can performed again extremely efficiently since it corresponds to matrix-vector multiplication. After the computation has been performed, we define the set of learnings $\mathcal{L}_{\alpha_1}$ by

$$\mathcal{L}_{\alpha_1} = \big\{L \in \mathcal{L} \mid c\big((\mathbf{x}_L, H_0(\mathbf{v}-\mathbf{u})\big) \geq \alpha_1\big\} \tag{14}$$

The set $\mathcal{L}_{\alpha_1}$ mainly contains learnings which are related to skills where the user has indeed deficits. The second pre-processing step ensures that the skill goals a user u reaches by attending a learning L are as close as possible to the skills the job v requires. Thus, analogously as for the matching function $r_{\mathbf{v}}$, for a learning $L \in \mathcal{L}$ and respective *rating-to* vector $\mathbf{x}_{R_{L,2}}$ given by

$$x_{R_{L,2},i} = \begin{cases} r_2(i), & \text{if } i \in I_L \\ 0, & \text{else}\ , \end{cases}$$

we define a penalty function $p_{\mathbf{x}_{R_{L,2}}}$ by

$$p_{\mathbf{x}_{R_{L,2}}}(\mathbf{v}) = \frac{1}{5n_{v,L}} \sum_{i \in I_v \cap I_L} |v_i - x_{R_{L,2},i}|,$$

where $n_{v,L} = |I_v \cap I_L|$. For a learning L with vector $\mathbf{x}_{R_{L,2}}$ and a job vector $\mathbf{v}$, it holds $p_{\mathbf{v}}(\mathbf{x}_{R_{L,2}}) \in [0,1]$. The smaller the value, the nearer the vector $\mathbf{x}_{R_{L,2}}$ is to the job's requirements given by $\mathbf{v}$. Given a job and the penalty function $p_{\mathbf{x}_{R_{L,2}}}$ we define the subset of learnings $\mathcal{L}_{\alpha_2} \subset \mathcal{L}$ as

$$\mathcal{L}_{\alpha_2} = \big\{L \in \mathcal{L} \mid 1 - p_{\mathbf{x}_{R_{L,2}}}(\mathbf{v}) \geq \alpha_2\big\} \tag{15}$$

Further, we allow the graph only to reach a maximal length $d_{\max}$ for a learning route.

All in all, for $\mathcal{L}_{\alpha_1}$ and $\mathcal{L}_{\alpha_2}$ as described in (14) and (15) for $\alpha_1, \alpha_2 \in (0,1)$, we set the the pre-processed learning set $\mathcal{L}_{\alpha_1,\alpha_2}$ as

$$\mathcal{L}_{\alpha_1,\alpha_2} = \mathcal{L}_{\alpha_1} \cap \mathcal{L}_{\alpha_2}. \tag{16}$$

For the case that $\mathcal{L}_{\alpha_1} \cap \mathcal{L}_{\alpha_2} = \varnothing$, we might choose less restrictive parameters α_1 and α_2.

5. TESTING

To give an example, we create a sample user u with vector representation $\mathbf{u} = (1,\ldots,1)^T \in R^{15}$ which chooses to receive a recommendation for a learning path with respect to a job v with

$$\begin{aligned} \mathbf{v} &= (3,0,5,0,5,4,0,5,3,5,1,1,0,1,0)^T \in R^{15}, \\ I_{v,\text{essential}} &= \{1,3,5,6,8,9,10\}, \\ I_{v,\text{optional}} &= \{11,12,14\} \end{aligned} \tag{17}$$

Note that $v_{11} = v_{12} = v_{14} = 1$ does not denote a low requirement of the skills but just that the skills are optionally required and therefore no weight is considered for the computation of the matching score.

The matching function $r_{\mathbf{v},\lambda_1,\lambda_2}$ is chosen according to (4) with

$$s(x,y) = s_\sigma(x,y) = \exp\{-|x-y|^2/\sigma\} \tag{18}$$

with parameters $\sigma = 3.6$, $\lambda_1 = 0.8$ and $\lambda_2 = 0.2$ which gives an initial matching score $r_{\mathbf{v},\lambda_1,\lambda_2}(\mathbf{u}) = 0.2900$.

We randomly generate 300 learnings of the form (10) by choosing the skills indices $i \in \{1,\ldots,15\}$ and the values $r_1(i), r_2(i) \in R$ uniformly at random and apply algorithm 2 with $d_{\max} = 5$.

The nearer α is to 1, the more restrictive is the pre-processing and the less learnings we may be able to apply to the user's vector representation $\mathbf{u}$ in order to update its skills profile. With $\alpha = 0$ no pre-prcoessing step is performed and therefore all possible combinations allowed by the parameters setting are considered. The larger the parameter β the higher must be the increase of the matching score of the updated profile in order to add the new node to the graph. A smaller β allows learnings with less high impact to be part of the learning route.

The parameter β also has influence on the number of nodes the graph G might contain. By choosing β very small, every learning which increases the matching score when the profile is updated, is considered to bo a valid learning. By setting a higher β we only add a learning to a path if its impact is strong enough. Therefore, it is also possible to estimate the length of a learning path dependent on the current parameter β and the user vector $\mathbf{u}$. For example, if we had a user vector $\mathbf{u}$ with $r_{\mathbf{v},\lambda_1,\lambda_2}(\mathbf{u}) = 0.5$ for a job vector $\mathbf{v}$ and set $\beta = 0.2$, then, the maximal number of learnings of a recommended learning path cannot exceed 5, regardless of what the parameter $d_{\max}$ was actually chosen to be.

Also, we might consider, to build the learnings route in such a way that the first learnings are the ones which have a higher impact on the matching score and to adapt the parameter β depending on the current learning step matchings score.

Table 2 also shows different results when the route planner algorithm is run with different pre-processing parameters α_1 and fixed $\alpha_2 = 0$ and $\beta = 0.2$, i.e. we only apply the cosine-pre-processing step. The computational times refer to an average over 5 runs with the same initial data. The first column of the table contains the pre-processing parameter α_1. In the second column we see the cardinality of the set $\mathcal{L}_{\alpha_i}$ which corresponds to the number of learnings which are indeed applied to a user profile. The variable $r_{\max}$ denotes the highest matching score $r_{\mathbf{v}}(\tilde{\mathbf{u}})$ for some updated user profile $\tilde{u}$ which we achieve with the given parameters.

The results motivate our believe that the algorithm should be applicable even with a higher number of learnings

TABLE 2 Variation of α_1 and α_2

α_1	$\lvert\mathcal{L}_{\alpha_1}\rvert$	$r_{\max}$	time [s]
0	300	0.9723	57.20
0.35	124	0.9723	6.7220
0.375	107	0.9723	2.46
0.4	106	0.9723	0.2.34
0.425	96	0.9723	1.64
0.45	86	0.9723	1.18
0.5	76	0.9445	0.66
0.525	69	0.7912	0.08

α_2	$\lvert\mathcal{L}_{\alpha_2}\rvert$	$r_{\max}$	time [s]
0	300	0.9723	57.20
0.5	187	0.9723	18.36
0.6	181	0.9723	16.35
0.7	138	0.9445	11.05
0.8	104	0.8039	0.5370
0.9	200	0.6697	0.01

Note that the random learnings are not distributed according to any particular distribution which we surely can expect from real data.

From this simple example we can state that the pre-processing parameters α_1 and α_2 have a crucial effect on the computational complexity of the graph construction algorithm.

We repeat the experiment with the parameter setting $\alpha_1 = 0, \beta = 0.2$ and vary α_2. Table 2 shows the results for the variation of α_2.

6. CONCLUSIONS AND FUTURE WORK

In the last pages we have presented the mathematical framework of the OpenSkiMr project. We have exposed our approach of formulating the problem of matching users and jobs as well as users and occupations. By introducing our route planner algorithm we have shown an intuitive way of building different learning routes which may lead a user to update its skills. The next steps of our project concern a further theoretical study of the algorithm's behaviour with respect to the different parameters as well as experiments envolving different functions s. In particular, the effects of the parameters α_1, α_2 and β should be carefully studied. Eventually, as already mentioned, we also might not choose a global β but let it be dependent on the current learning step.

Also, assuming a very high number of learnings, we may think of clustering the learnings and building the initial set $\mathcal{L}$ by adding only some of the learnings per cluster. This avoids the problem of having solely learnings which are similar in terms of skills they are related to after the pre-processing with the parameters α_1 and α_2.

In order to be able to recommend a learning route, we need the content of learnings to be analyzed and adapted to the form our algorithm can work with. Our developing team is engaged with the implementation of the platform whose goal is to collect real data about users as well as job advertisements as soon as possible. We expect our route planner algorithm to work on real data as well as it exploits the natural underlying structure of it.

FUNDING

This work has been supported by the European Union with Agreement Number ECOKT 2014-4-30-CE-0741117/00-28**.**

ACKNOWLEDGEMENTS

The authors would like to thank the whole OPENSKIMR project team for the pleasant ambience and the numerous discussions which led to this work

REFERENCES

[1] CEDEFOP (2015). Matching Skills and Jobs in Europe. European Centre for the Development of Vocational Training, Europe 123, 570 01 Thessaloniki (Pylea), Greece, 2nd edition.

[2] European Commission (2013). European Classification of Skills/Competences, Qualifications and Occupations. European Commission, Publications Office of the European Union: Luxembourg, 1st edition.

[3] Ricci, F., Rokach, L., and Shapira, B. (2015). Recommender Systems Handbook -Springer, Berlin, Heidelberg, 2nd edition.

[4] Liu, B. (2011). Web Data Mining - Exploring Hyperlinks, Contents, and Usage Data.Springer Science and Business Media, Berlin Heidelberg.

[5] Aric A. Hagberg, Daniel A. Schult, P. J. S. (2008). Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 7th Python in Science Conference (SciPy2008), pages 1115, Pasadena, CA USA.

[6] Gan, G., Ma, C., and Wu, J. (2007). Data Clustering - Theory, Algorithms, and Applications.SIAM, Philadelphia.