Metadata of the chapter that will be visualized in SpringerLink

Book Title	Foundations of Information and Knowledge Systems	
Series Title		
Chapter Title	Semantic Matching Strategies for Job Recruitment: A Comparison of New and Known Approaches	
Copyright Year	2016	
Copyright HolderName	Springer International Publishing Switzerland	

Author	Family Name	**Rácz**
	Particle	
	Given Name	**Gábor**
	Prefix	
	Suffix	
	Division	Alfréd Rényi Institute of Mathematics
	Organization	Hungarian Academy of Sciences
	Address	P.O.Box 127, Budapest, 1364, Hungary
	Email	gabee33@gmail.com
Corresponding Author	Family Name	**Sali**
	Particle	
	Given Name	**Attila**
	Prefix	
	Suffix	
	Division	Alfréd Rényi Institute of Mathematics
	Organization	Hungarian Academy of Sciences
	Address	P.O.Box 127, Budapest, 1364, Hungary
	Email	sali.attila@renyi.mta.hu
Author	Family Name	**Schewe**
	Particle	
	Given Name	**Klaus-Dieter**
	Prefix	
	Suffix	
	Division	
	Organization	Software Competence Center Hagenberg
	Address	Softwarepark 21, 4232, Hagenberg, Austria
	Email	kd.schewe@scch.at

Abstract	A profile describes a set of skills a person may have or a set of skills required for a particular job. Profile matching aims to determine how well a given profile fits to a requested profile. The research reported in this paper starts from exact matching measure of [21]. It is extended then by matching filters in ontology hierarchies, since profiles naturally determine filters in the subsumption relation. Next we take into consideration similarities between different skills that are not related by the subsumption relation. Finally, a totally different approach, probabilistic matching based on the maximum entropy model is analyzed.

Keywords (separated by '-')	Semantic matching - Ontology - Lattice filters - Probabilistic matching - Maximum entropy model

Semantic Matching Strategies for Job Recruitment: A Comparison of New and Known Approaches

Gábor Rácz[1], Attila Sali[1(✉)], and Klaus-Dieter Schewe[2]

[1] Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, P.O.Box 127, Budapest 1364, Hungary
gabee33@gmail.com, sali.attila@renyi.mta.hu
[2] Software Competence Center Hagenberg, Softwarepark 21, 4232 Hagenberg, Austria
kd.schewe@scch.at

Abstract. A profile describes a set of skills a person may have or a set of skills required for a particular job. Profile matching aims to determine how well a given profile fits to a requested profile. The research reported in this paper starts from exact matching measure of [21]. It is extended then by matching filters in ontology hierarchies, since profiles naturally determine filters in the subsumption relation. Next we take into consideration similarities between different skills that are not related by the subsumption relation. Finally, a totally different approach, probabilistic matching based on the maximum entropy model is analyzed.

Keywords: Semantic matching · Ontology · Lattice filters · Probabilistic matching · Maximum entropy model

1 Introduction

A profile describes a set of properties and profile matching is concerned with the problem to determine how well a given profile fits to a requested one. Profile matching appears in many application areas such as matching applicants for job requirements, matching system configurations to requirement specifications, etc.

The simplest idea of profile matching is to consider profiles as sets of unrelated items. There are several ways to define distances of sets, such as Jaccard or Sørensen-Dice measures [14] turned out to be useful in ecological applications. However, many dependencies between skills or properties included in profiles exist and need to be taken into account. In the human resources area several taxonomies for skills, competences and education such us DISCO [7], ISCED [9] and ISCO [10] have been set up. Based on these taxonomies a lattice structure

The research reported in this paper has been [partly] supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

M. Gyssens and G. Simari (Eds.): FoIKS 2016, LNCS 9616, pp. 1–20, 2016.
DOI: 10.1007/978-3-319-30024-5_9

of the individual properties can be assumed. Popov and Jebelean [21] exploited this by defining an asymmetric matching measure on the basis of filters in such lattices.

However, there are other relations between skills and properties than the ones given by the ontolgies above. Having some skills imply that the applicant may have some other skills with certain probabilities, or of some (not complete) proficiency level. For example, we may reasonably assume that knowledge of Java implies knowledge of Netbeans up to a grade of 0.7 or with probability 0.7. Our new approach incorporates such relations in the following way. The subsumption hierarchy of the ontology of skills is considered as a directed graph with edge weights 1. A lattice filter generated by a profile corresponds to the set of nodes reachable from the profile's nodes in the directed graph. Then extra edges are added with weights representing the probability/grade of the implication between skills or properties. This may introduce directed cycles. Filters of application profiles are replaced by nodes reachable in the extended graph from the profile's nodes. However, for each vertex x reached a probability/grade is assigned, the largest probability of a path from the profile's nodes to x. Path probability is defined as the product of probabilities of edges of the path. At first sight it seems that determination of the grade of a vertex involves finding a longest path in weighted directed graph, known to be an NP-complete problem. However, in our case the weighting is multiplicative and less than 1, so Dijkstra's Algorithm [5] can be applied. This process results in a set of nodes with grades between zero and one, so it can naturally be interpreted as a fuzzy set. In fact, we prove that it is a fuzzy filter as defined in [8,16].

Considering the grades as probabilities suggests another approach. They can be handled from an information theoretic point of view, with probabilistic logic programs [11] or from set theoretic point of view, with probabilistic models [25], as well. In the present paper the maximum entropy model is used which adds the lowest amount of additional information between single elementary probabilities to the system. In order to apply probabilistic model the information represented by the extended directed graph is translated into sentences over an appropriate measurable space. The matching value of a job offer O and an application A is the result of the probabilistic query obtained from the sentences.

Our paper is organized as follows.

Section 2 contains the description of our novel model of extending the ontology hierarchy with cross relations in the form of weighted directed edges. Two new ranking algorithms are given for job applications and a connection with fuzzy theory is mentioned.

Section 3 is devoted to the comparison of the different approaches. Our findings show that these are basically independent of each other apart from some natural dependences as some of the approaches are extensions of some other ones.

Section 4 discusses related work and how our approach fits into the broad area of semantic matching.

Finally, Sect. 5 contains conclusions.

2 Semantic Matching

Let $S = \{s_1, s_2, \ldots, s_n\}$ be a finite set of skills. Let a job offer $O = \{o_1, o_2, \ldots, o_k\}$ be a subset of S which contains the skills that are required for the job. Then, an application $A = \{a_1, a_2, \ldots, a_l\}$ is also a subset of S which means the applicant possesses these skills. Our task is to find the most suitable applicant for a job offer. Example 2.1 shows a job offer with four applications.

Example 2.1 (A job offer and four applications)

$$\begin{aligned} Offer_1 &= \{Java, Netbeans, XML\} \\ Application_1 &= \{Java, PHP, Eclipse\} \\ Application_2 &= \{Java, Netbeans, HTML\} \\ Application_3 &= \{C, PHP, XML\} \\ Application_4 &= \{C, Netbeans, XML\} \end{aligned}$$

Note, that skills are not graded (e.g., basic/medium/expert knowledge in Java) in our examples. If we need such differentiation, we have to handle the grades as separate skills.

2.1 Perfect Matching

A simple idea to decide how much an application fits to a job offer is to compute the number of the matching skills the ones that the applicant possesses and that are required for the job. The result can be normalized with the number of the required skills [21]. Formally,

$$match(O, A) = \frac{|O \cap A|}{|O|}. \tag{1}$$

In the following example, we compute the matching values of the job offer and the applications from Example 2.1.

Example 2.2 (Perfect matching)

$$\begin{aligned} match(O_1, A_1) &= \frac{|O_1 \cap A_1|}{|O_1|} = \frac{|\{Java\}|}{|\{Java, Netbeans, XML\}|} = \frac{1}{3} \\ match(O_1, A_2) &= \frac{|O_1 \cap A_2|}{|O_1|} = \frac{|\{Java, Netbeans\}|}{|O_1|} = \frac{2}{3} \\ match(O_1, A_3) &= \frac{|O_1 \cap A_3|}{|O_1|} = \frac{|\{XML\}|}{|O_1|} = \frac{1}{3} \\ match(O_1, A_4) &= \frac{|O_1 \cap A_4|}{|O_1|} = \frac{|\{Netbeans, XML\}|}{|O_1|} = \frac{2}{3} \end{aligned}$$

As it can be seen, this matching function cannot sufficiently distinguish between the applications. It assigned A_2, A_4 and A_1, A_3 the same values, respectively. However, as our goal is to find the most suitable applicants, we want to avoid that two or more candidates get the same values. Therefore, we need extra knowledge to be able to distinguish A_2 from A_4 and A_1 from A_3.

2.2 Matching Using Ontology Edges

Let us suppose that the skills in S form a hierarchy. We can represent a hierarchy several ways, for example, with Description Logic [1], with Resource Description Framework Schema [3] or with Logic programming [17]. We use the description logic approach here, so let the skills corresponds to concepts and we define a specialization relation over them.

Let $\preceq$ be a binary specialization (subsumption) relation over S. Let $s_i, s_j \in S$ be two skills, then $s_i \preceq s_j$ if s_i is a *specialization* of s_j. It means if an applicant possesses the skill s_i, he also possesses the skill s_j as s_i is a more specific skill than s_j (s_j is more general than s_i). Let $\preceq_d$ denote the direct specialization, i.e., $s_i \preceq_d s_j$ if and only if $s_i \preceq s_j$ and $\nexists s_k \in S$ such that $s_k \neq s_i, s_k \neq s_j$ and $s_i \preceq s_k \preceq s_j$. Note, that $\preceq$ is a reflexive, antisymmetric and transitive relation, i.e., it is a partial order over S.

We can always add a top (respectively a bottom) element to the hierarchy that represents all the skills that everybody (nobody) possesses. In addition, let us suppose the concepts define a lattice, i.e., for each pair of skills has unique ancestor (and descendant) that is more general (specific) then both elements of the pair. Let this lattice be denoted by $(S, \preceq)$.

In Fig. 1, the blue edges form a hierarchy among computer science skills.

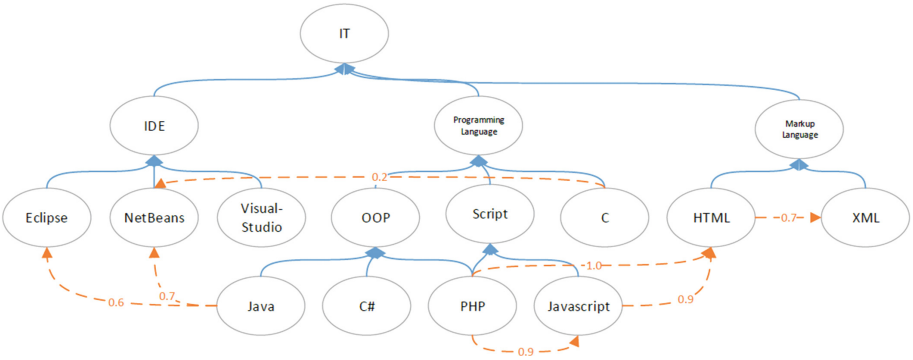

Fig. 1. A hierarchy of skills. The ontology edges are the blue ones (solid) and the extra edges are the orange ones (dashed) (Color figure online).

Definition 2.1. *A filter in a lattice $(S, \preceq)$ is a non-empty subset $F \subseteq S$, such that for all $s, s' \in S$ with $s \preceq s'$ whenever $s \in F$ holds, then also $s' \in F$ holds.*

An $A \subseteq S$ application defines in a natural way an F filter of the $(S, \preceq)$ lattice:

$$F = \{s \in S \mid \exists a \in A, a \preceq s\}.$$

This filter is an extension of the original application with the skills that are more general than the ones in the application. It is reasonable, because if an application possesses a skill, then he must possesses all the skills that are more

general by the definition. Note, that a job offer can be extended in the same way as an application. And then, we can apply, for example, the perfect matching function on the extended sets.

The next example shows the filters defined by the job offer and of the applications from Example 2.1, and the values of the perfect matching function applying on the extensions. As can be seen, this method was already able to distinguish A_2 from A_4.

Example 2.3 (Matching based on an ontology)

$$
\begin{aligned}
F_{O_1} &= \{Java, Netbeans, XML, OOP, PL, IT, IDE, ML\} \\
F_{A_1} &= \{Java, PHP, Eclipse, OOP, PL, IT, Script, IDE\} \\
F_{A_2} &= \{Java, Netbeans, HTML, OOP, PL, IT, IDE, ML\} \\
F_{A_3} &= \{C, PHP, XML, PL, IT, Script, OOP, ML\} \\
F_{A_4} &= \{C, Netbeans, XML, PL, IT, IDE, ML\}
\end{aligned}
$$

$$
\begin{aligned}
match(F_{O_1}, F_{A_1}) &= \frac{|\{Java, OOP, PL, IT, IDE\}|}{|F_{O_1}|} = \frac{5}{8} \\
match(F_{O_1}, F_{A_2}) &= \frac{|\{Java, OOP, PL, IT, Netbeans, IDE, ML\}|}{|F_{O_1}|} = \frac{7}{8} \\
match(F_{O_1}, F_{A_3}) &= \frac{|\{XML, PL, IT, OOP, ML\}|}{|F_{O_1}|} = \frac{5}{8} \\
match(F_{O_1}, F_{A_4}) &= \frac{|\{Netbeans, XML, PL, IT, IDE, ML\}|}{|F_{O_1}|} = \frac{6}{8}
\end{aligned}
$$

2.3 Maximal Length Matching

In this section, we are adding extra knowledge to the hierarchy in form of extra edges, and we are investigating how it can be used to find the most suitable application for a job. However, these extra edges can form cycles in the hierarchy, therefore the traditional filters are not applicable in this case. Immediate example for a cycle could be two skills that are connected with two extra edges which are pointing in opposite direction. For this reason, we describe a graph based approach to extend the applications and the offer. Then, we show that this approach corresponds to the definition of fuzzy filters [8,16].

Let $G = (V, E)$ be a directed weighted graph where V is a finite non-empty set of nodes and $E = \{E_O \cup E_E\} \subseteq V \times V$ is a set of edges. Each node represents a skill and an $e_o = (v_i, v_j) \in E_O$ directed edge is added from the skill v_i to v_j if v_i is a specialization of v_j (this type of edges are called ontology edges). Moreover, an $e_e = (v_i, v_j) \in E_E$ represents a conditional dependency between the skills v_i and v_j. Namely, if a person has the skill v_i then he may have the skill v_j (this type of edges are called extra edges). Let $w : E \to (0, 1]$ be a weighting function that assigns a weight to all edges such that for all ontology edges $e_o \in E_O$ $w(e_o) = 1$ and for all extra edges $e_e \in E_E$ let the weight $w(e_e)$ represents the conditional probability between the start and the end node of the edge. The edge weight can come, for example, from Human Resources experiments or from domain experts.

Definition 2.2. *Let $G = (V, E_O \cup E_E)$ a directed weighted graph with a $w : E \to (0, 1]$ weighting function and let $(S, \preceq)$ be a lattice. We say that G is built on $(S, \preceq)$ if $V = S$ and for all $v_i, v_j \in V$ $(v_i, v_j) \in E_O$ if and only if $v_i \preceq_d v_j$.*

Let $s, t \in V$ be two nodes, then denote by $p_E(s,t)$ all the directed paths between s and t, i.e.,

$$p_E(s,t) = \{(s = v_1, v_2, \ldots, v_n = t) \mid v_i \in V \text{ and } (v_i, v_{i+1}) \in E\}.$$

The extra edges and the directed paths are used to extend the applications with skills that the applicant possibly possesses. Since both sets and uncertainty occur, fuzzy sets [28] are suitable to model the extended applications. A fuzzy set assigns a value for each element that expresses the certainty of that the element is in the set or not.

Definition 2.3. *A fuzzy set in S is a mapping $f : S \to [0,1]$. A fuzzy set is called empty if f is identically zero on S. Let t be a real number such that $t \in [0,1]$, then the set $f_t = \{s \in S \mid f(s) \geq t\}$ is called a level subset of f.*

Note that, as S is a finite set, we can define a fuzzy set by enumerating all elements in S with their assigned values (called the grade of membership) if that value is greater than zero, i.e.,

$$f = \{(s, f(s)) \mid s \in S \text{ and } f(s) > 0\}.$$

The intersection and the union of tho fuzzy sets can be defined axiomatically with t-norms and t-conorms [20]. For clarity we use *min* and *max* as t-norm and t-conorm, i.e., for fuzzy sets f, g in S, we define the intersection and the union operations as $(f \cap g)(s) := \min\{f(s), g(s)\}$ and $(f \cup g)(s) := \max\{f(s), g(s)\}$ for all $s \in S$, respectively.

We extend the job offer and the applications to fuzzy sets in the following way. Let $O \subseteq V$ be a set of skills. The extension of O w.r.t. E_O is defined as the set of all the skills that are available from O via directed paths containing edges only from E_O. We assign 1.0 to each element of the extension to create a fuzzy set, that is

$$\mathit{extend}_{E_O}(O) = \widehat{O} = \{(v, 1.0) \mid v \in V \text{ and } \exists o \in O : |p_{E_O}(o, v)| \geq 1\}.$$

Let $A \subseteq V$ be a set of skills. The extension of A w.r.t. E is defined as the set of all the skills that are available from A via directed paths containing ontology or extra edges, and we assign the length of the longest path between the node and the elements of A to each element of the extended set, namely

$$\mathit{extend}_E(A) = \widehat{A} = \{(v, \mu_v) \mid v \in V \text{ and } \exists a \in A : |p_E(a, v)| \geq 1 \text{ and } \mu_v = \max_{a \in A, p \in p_E(a,v)} \mathit{length}(p)\},$$

where $\mathit{length}(p) = \prod_{i=1}^{n-1} w((v_i, v_{i+1}))$ is the product of the weights of the edges on the path p. The μ_v is the length of the longest path from A to v. If the edge weights mean uncertainty or probability, the length of longest path means the joint probability of the applicant possessing all the skills on the path (if we assume some independence), which seems a rational decision.

Note that finding the longest path between two nodes in a graph is generally a hard problem. However, in our case the length of a path is defined as the product of the weight of the edges on the path. Therefore, because of the strict monotonicity of the logarithm function, we can apply the following transformations:

$$\max_{p\in p_E(v,a)} \prod_{i=1}^{n-1} w((v_i, v_{i+1})) = \max_{p\in p_E(v,a)} \log\{\prod_{i=1}^{n-1} w((v_i, v_{i+1}))\} =$$
$$\max_{p\in p_E(v,a)} \sum_{i=1}^{n-1} \log w((v_i, v_{i+1})) = -\min_{p\in p_E(v,a)} \sum_{i=1}^{n-1} -\log w((v_i, v_{i+1})).$$

Moreover, the weighting function w assigns a weight from the $(0, 1]$ interval to each edge, thus $-\log w((v_i, v_{i+1})) \in [0, +\inf)$. With these transformation, we got a single-source shortest path problem with non-negative edge weights. That problem can be solved with Dijkstra's algorithm in $O(|E| + |V| \log |V|)$ time [5].

Definition 2.4. *Let $(S, \preceq)$ be a lattice, and let f be a fuzzy set in S. f is called a fuzzy filter if for all $t \in [0, 1]$, f_t is either empty or a filter of S.*

Next, we show that the extensions presented above define fuzzy filters in S.

Lemma 2.1. *Let $(S, \preceq)$ be a lattice, let $G = (V, E_O \cup E_E)$ be a directed weighted graph built on the lattice with the weighting function w and let $s, s' \in S$ be a skill pair such that $s \preceq s'$. Then, there is a $p \in P_E(s, s')$ path such that length$(p) = 1$.*

Proof. Since $s \preceq s'$, an $s = s_1, \ldots, s_k = s'$ sequence of skills exists such that $s_i \preceq_d s_{i+1}$ holds for $i = 1, \ldots, k-1$. As G is built on the lattice, each skill is represented by a node in the graph, and $(s_i, s_{i+1}) \in E_O$ and $w((s_i, s_{i+1})) = 1$ for $i = 1, \ldots, k-1$. Let $p = (s_1, \ldots, s_k)$ be a path between v_1 and v_n. Consequently, $length(p) = \prod_{i=1}^{k-1} w((s_i, s_{i+1})) = 1$ □

Theorem 2.1. *Let $(S, \preceq)$ be a lattice, let $G = (V, E_O \cup E_E)$ be a directed weighted graph built on the lattice with weighting function w, let $A \subseteq S$ be a non-empty application, and let $\widehat{A}$ be the extension of A w.r.t. E. Then, $\widehat{A}$ is a fuzzy filter in S.*

Proof. For $t \in [0, 1]$, $\widehat{A}_t = \{s \in S | \widehat{A}(s) \geq t\}$. $\widehat{A}_t$ is a filter in S if for all $s, s' \in S$ with $s \preceq s'$ whenever $s \in \widehat{A}_t$ holds, then also $s' \in \widehat{A}_t$ holds. It means if $\widehat{A}(s) \geq t$, then $\widehat{A}(s') \geq t$.

Let s be in $\widehat{A}_t$ and let $p_{a,s} = (a = s_{i_1}, s_{i_2}, \ldots, s_{i_k} = s)$ be one of the maximal length path between A and s. We have to show that if an $s' \in S$ is a generalization of s, i.e., $s \preceq s'$, then a $p_{a,s'}$ path exists such that $length(p_{a,s'}) \geq length(p_{a,s})$.

Lemma 2.1 states that a $p_{s,s'} = (s = s_{j_1}, s_{j_2}, \ldots, s_{j_l} = s')$ path exists such that $length(p_{s,s'}) = 1$. If $p_{a,s}$ and $p_{s,s'}$ are disjoint, so they do not have any node in common except s, then we can concatenate them to $p_{a,s'} = (a = s_{i_1}, \ldots, s_{i_k}, s_{j_1}, \ldots, sj_l = s')$ and its length is $length(p_{a,s'}) = length(p_{a,s}) * 1 \geq$

length$(p_{a,s})$. Otherwise recursively, let $t = \min_{x\in[1,k]}\{s_{i_x}|\exists y \in [1,l] : s_{i_x} = s_{j_y}\}$ be the first common node. Then, consider the $p_{a,t} = (a = s_{i_1}, 'ldots, s_{i_x} = t)$ and the $p_{t,s'} = (t = s_{j_y}, s_{j_{y+1}}, \ldots, s_{j_l} = s')$ paths. Where *length*$(p_{a,t}) \geq$ *length*$(p_{a,s})$ as $w(e) \leq 1$ for all $e \in E$ and *length*$(p_{t,s'}) = 1$ as it contains ontology edges only. If these paths are disjoint except t then we can concatenate them, otherwise repeat this step. Since the number of nodes that are contained in the paths are limited and every step reduces that number, the iteration will stop in finite step. □

Let $\widehat{O}$ be an extended job offer w.r.t. E_O and let $\widehat{A}$ be an extended application w.r.t. E, both are fuzzy sets. We can define a matching value between $\widehat{O}$ and $\widehat{A}$ in a similar way as we did in Eq. 1.

$$match(\widehat{O}, \widehat{A}) = \frac{||\widehat{O} \cap \widehat{A}||}{||\widehat{O}||}, \tag{2}$$

where $||.||$ denotes the sum of the grades of membership of the elements in a fuzzy set, formally $||\widehat{A}|| = \sum_{(a,\mu_a)\in A} \mu_a$. Note that, besides the sigma cardinality used here, many other options are available on the cardinality of a fuzzy set [27].

Algorithm 1 shows how can we find the most suitable applicant for a job using the extensions defined above. The algorithm works as follows:

Algorithm 1. MaximalLengthMatching

Input: a graph $G = (V, E = \{E_O \cup E_E\})$, a job offer $O \subseteq V$, and a set of applications $\boldsymbol{A} = \{A_0, A_2, \ldots, A_n\}$.
Output: the most suitable application A_m for the job

```
1: O_e ← extend_{E_O}(O)
2: A_m ← A[0], max ← extend_E(A[0])
3: for i = 1 → n do
4:     A_e ← extend_E(A[i])
5:     if match(O_e, A_e) > match(O_e, max) then
6:         A_m ← A[i], max ← A_e
7:     end if
8: end for
9: return A_m
```

First, we extend the job offer O with the skills that are available from O via ontology edges (line 1). It is a reasonable extension because the ontology edges represent specialization relation between to skills. And if an applicant possesses a more general skill than the required one, then he could specialize faster than an applicant that does not possess even that general skill. As everyone possesses the skill at the top of the hierarchy, that does not distinguish one applicant from another.

Next, we also extend each application (line 2, 4). In this step, however, we take into consideration the extra edges as well. This is because, if the skill v_i is

a specialization (or conditionally depends, respectively) of another skill v_j, and a person possesses the skill v_i, then he also possesses (may possess) the skill v_j. The matching value of the extended job offer and the extended application is then computed using Eq. 2 (line 5). The most suitable application is stored in the variable A_m (line 6).

The next example shows the extension of the $Offer_1$ and the $Application_1$ from Example 2.1, and their matching value.

Example 2.4 (Maximal length matching).

$$\widehat{O_1} = \{(Java, 1.0), (Netbeans, 1.0), (XML, 1.0), (OOP, 1.0), (PL, 1.0), (IT, 1.0), (IDE, 1.0), (ML, 1.0)\}$$

$$\widehat{A_1} = \{(Java, 1.0), (PHP, 1.0), (Eclipse, 1.0), (OOP, 1.0), (PL, 1.0), (IT, 1.0), (Script, 1.0), (IDE, 1.0), (Netbeans, 0.7), (Javascript, 0.9), (HTML, 1.0), (ML, 1.0), (XML, 0.7)\}$$

$$\widehat{O_1} \cap \widehat{A_1} = \{(Java, 1.0), (OOP, 1.0), (PL, 1.0), (IT, 1.0), (IDE, 1.0), (Netbeans, 0.7), (XML, 0.7), (ML, 1.0)\}$$

$$m(\widehat{O_1}, \widehat{A_1}) = \frac{||\widehat{O_1} \cap \widehat{A_1}||}{||\widehat{O_1}||} = \frac{7.4}{8}$$

As one can see, OOP, PL, IT, IDE and ML appeared in the offer as they are available from the originally specified skills (Java, Netbeans, XML) via ontology edges (the solid ones) as Fig. 1 shows. In addition, the $extend_{E_O}$ method assigned 1.0 to each skills in the offer and transformed $\widehat{O_1}$ to fuzzy set. The application was also extended but the $extend_E$ method used the extra edges (the dashed ones) as well during the extension. The intersection of the two extended sets consists of those elements that appeared in both sets and the assigned values are computed with the *min* function as t-norm. Finally, the ratio of the sum cardinalities is calculated.

2.4 Probabilistic Matching

The ontology and the extra edges can be handled from an information theoretic point of view, with probabilistic logic programs [11] or from set theoretic point of view, with probabilistic models [25] as well. In this paper, we use the latter, the set theoretical approach and we apply the maximum entropy model to give a probabilistic matching method. The following definitions were presented in [25].

2.4.1 Preliminaries

Let Θ be a finite set. Let $R := \{a_1, \ldots, a_l\}$ be a set of subsets of the power set $\mathcal{P}(\Theta)$ of Θ, namely $a_i \in \mathcal{P}(\Theta)$, $i = 1, \ldots, l$. The elements of R are called *events*.

Definition 2.5. *Let X be some set. Let $\mathcal{A}$ be a subset of $\mathcal{P}(X)$. Then, $\mathcal{A}$ is a σ-algebra over X, denoted by $\mathcal{A}(X)$, if*

- $X \in \mathcal{A}$;
- *if $Y \in \mathcal{A}$, then $(X \setminus Y) \in \mathcal{A}$; and*
- *if $Y_1, Y_2, \ldots$ is a countable collection of sets in $\mathcal{A}$, then their union $\bigcup_{n=1}^{\infty} Y_n$ is in $\mathcal{A}$ as well.*

The set of full conjunction over R is given by

$$\Omega := \left\{ \bigcap_{i=1}^{l} e_i \mid e_i \in \{a_i, \neg a_i\} \right\},$$

where $a_i \in \mathcal{P}(\Theta)$, $i = 1, \ldots, l$, and $\neg a_i = \Theta \setminus a_i$. It is well known that the 2^l elements of Ω are mutually disjoint and span the set R (any a_i can be expressed by a disjunction of elements of Ω). Therefore the smallest (σ-) algebra $\mathcal{A}(R)$ that contains R is identical to $\mathcal{A}(\Omega)$. For that reason we restrict the set of *elementary events* (set of possible worlds) to Ω instead of the underlying Θ.

Definition 2.6. *Over a set $R := \{a_1, \ldots, a_l\}$, a measurable space $(\Omega, \mathcal{A})$ is defined by*

- $\Omega := \left\{ \bigcap_{i=1}^{l} e_i \mid e_i \in \{a_i, \neg a_i\} \right\}$; *and*
- $\mathcal{A} = \mathcal{A}(\Omega) = \mathcal{P}(\Omega)$.

Definition 2.7. *Let $(\Omega, \mathcal{A})$ be a measurable space over R with $\Omega = \{\omega_1, \ldots, \omega_n\}$. A discrete probability measure P or a probability model (P-model) is an assignment of non-negative numerical values to the elements of Ω, which sum up to unity. Formally,*

$$p_i := P(\omega_i) \geq 0, \ i = 1, \ldots, n \text{ and } \sum a_i = 1.$$

The n-tuple $\boldsymbol{p} = (p_1, \ldots, p_n)$ is called a probability vector (P-vector). W_Ω (respectively, V_Ω) denotes the set of all possible P-models (P-vectors) for $(\Omega, \mathcal{A})$.

Definition 2.8. *For given $(\Omega, \mathcal{A})$, $P \in W_\Omega$ $a, b \in \mathcal{A}, P(a) > 0$ and $[l, u] \subseteq [0, 1]$ the term*[1]

$$\langle P(b|a) = \delta, \ \delta \in [l, u] \rangle \text{ or } P(b|a)[l, u]$$

is called a sentence in $(\Omega, \mathcal{A})$, where $P(b|a) = P(a \cap b)/P(a)$ denotes the conditional probability of the event b given a. The sentence given above is called true in $P \in W_\Omega$, if and only if $P(b|a) \in [l, u]$. Otherwise it is called false.

[1] $P(a) = P(a|\Omega)$.

A sentence $P(b|a)[l,u]$ defines two inequalities, namely

- $P(b|a) \leq u$ (be less than the upper bound); and
- $P(b|a) \geq l$ (be greater than the lower bound).

These inequalities can be further transformed in the following way:

$$P(b|a) \leq u \Leftrightarrow P(a \cap b) \leq u \cdot P(a) \quad \Leftrightarrow P(a \cap b) \leq u \cdot (P(a \cap b) + P(a \cap \neg b))$$
$$P(b|a) \geq l \Leftrightarrow P(a \cap b) \geq l \cdot P(a) \quad \Leftrightarrow P(a \cap b) \geq l \cdot (P(a \cap b) + P(a \cap \neg b))$$

Rearranging the inequalities and using the elementary probabilities $p_i, i = 1, \ldots, n$ yields

$$P(b|a) \leq u \Leftrightarrow (1-u) \cdot \sum_{i:w_i \in a \cap b} p_i + u \cdot \sum_{j:w_j \in a \cap \neg b} p_j \geq 0$$
$$P(b|a) \geq l \Leftrightarrow (1-l) \cdot \sum_{i:w_i \in a \cap b} p_i - l \cdot \sum_{j:w_j \in a \cap \neg b} p_j \geq 0$$

Note, that, if $u = 1$ (respectively, $l = 0$), then the first (second) inequality is always satisfied as $p_i \geq 0$.

Definition 2.9. *Let $DB := \{c_1, \ldots, c_m\}$ be a set of m sentences in $(\Omega, \mathcal{A})$. W_{DB} is defined as the set of all P-models $P \in W_\Omega$ in which $c_1, \ldots, c_m$ are true. We call $c_1, \ldots, c_m$ constraints on W_Ω, and W_{DB} denotes the set of all elements of W_Ω that are consistent with the constraints in DB.*

If W_{DB} is empty, the information in DB is inconsistent. If W_{DB} contains more than one element, the information in DB is incomplete for determining a single P-model.

In the next section, we discuss how the maximum entropy model copes with incomplete information.

2.4.2 Maximum Entropy Model

If W_{DB} contains more than one element, the information in *DB* is incomplete for determining a single P-model. Therefore, we must add further constraints to the system to get a unique model.

It was shown in [25] that the maximum entropy model adds the lowest amount of additional information between single elementary probabilities to the system. Moreover, the maximum entropy model also satisfies the principle of indifference and the principle of independence. The principle of indifference states that if we have no reason to expect one event rather than another, all the possible events should be assigned the same probability. The principle of independence states the if the independence of two events a and b in a P-model ω is given, any knowledge about the event a does not change the probability of b (and vice verse) in ω, formally $P(b|a) = P(b)$.

To get the consistent P-model to a DB that has the maximum entropy, we have to solve the following linear optimization problem:

Definition 2.10. *Let* $DB := \{c_1, \ldots, c_m\}$ *be a set of* m *sentences in* $(\Omega, \mathcal{A})$ *with* $\Omega = \{\omega_1, \ldots, \omega_n\}$. *Let* W_{DB} *(respectively* V_{DB}*) be the set of all P-models* $P \in W_\Omega$ *(P-vectors* $\boldsymbol{p} \in V_\Omega$*) in which* $c_1, \ldots, c_m$ *are true. The maximum entropy problem is*

$$\max_{\boldsymbol{v}=(p_1,\ldots,p_n)\in[0,1]^n} -\sum_{i=1}^{n} p_i \log p_i$$

subject to

$$\sum_{i=1}^{n} p_i = 1$$

$$(1-u)\cdot \sum_{i:w_i\in a\cap b} p_i + u \cdot \sum_{j:w_j\in a\cap\neg b} p_j \geq 0 \ (for\, all\, c = P(b|a)[l,u] \in DB, l > 0)$$

$$(1-l)\cdot \sum_{i:w_i\in a\cap b} p_i - l \cdot \sum_{j:w_j\in a\cap\neg b} p_j \geq 0 \ (for\, all\, c = P(b|a)[l,u] \in DB, u < 0)$$

$$p_i \geq 0 (i = 1, \ldots, n)$$

Denote by $me[DB]$ the P-model that solves the maximum entropy problem if such model exists.

Definition 2.11. *Let DB be a set of sentences and let* $c = \langle P(b|a)[l,u]\rangle$ *be a sentence. We say that c is a maximum entropy consequence of DB, denoted by* $DB \,\|\!\sim^{me} c$*, if and only if either*

- *DB is inconsistent, or*
- $me[DB](b|a) \in [l,u]$.

Definition 2.12. *A probabilistic query is an expression* $QP_{DB}(b|a)$ *where a and b are two events, i.e.,* $a, b \in \mathcal{A}$*, and DB is a set of sentences. The query means, what is the probability of b given a with respect to DB.*

Definition 2.13. *Let DB be a set of sentences and let* $QP(b|a)$ *be a probabilistic query. Then, the answer* δ *to the query is*

$$\delta := me[DB](b|a) = \frac{me[DB](a\cap b)}{me[DB](a)}$$

if $DB \,\|\!\sim^{me} P(a)(0,1]$. *Otherwise,* $\delta := -1$ *means that the set* $DB \cup \{P(a)(0,1]\}$ *is inconsistent.*

The next section shows how an ontology can be transformed into a set of sentences, and how the semantic matching problem can be expressed with probabilistic queries.

2.4.3 Probabilistic Matching

Let $G = (V, \{E_O \cup E_E\})$ be a directed weighted graph as it was defined in Sect. 2.3. We construct a DB from G in the following way:

- We assign a new set (event) a_v to each node v of G which contains the applicants who possess the skill v.
- Next, for each ontology edge $(v_i, v_j) \in E_O$ we add a new sentence s_{ij} to the DB in the form of $P(a_{v_j}|a_{v_i})[1,1]$. The sentence means if an applicant possesses the skill v_i (is an element of a_{v_i}), then the applicant possesses v_j (is an element of a_{v_j}) as well.
- Then, for every extra edge $(v_i, v_j) \in E_E$, we also add a new statement in the form of $P(a_{v_j}|a_{v_i}) = [l, u]$. The weight of an edge can be handled in two different ways. In the first approach, let the lower bound of the interval l be equal to the weight of the edge $w(v_i, v_j)$ and let the upper bound of the interval u be equal to 1. In the second approach, let $l = u = w(v_i, v_j)$. The latter is the stricter approach as it adds constraints to the upper bounds as well.

An application $A = \{v_1, \ldots, v_n\}$ is translated into the event $a = a_{v_1} \cap \cdots \cap a_{v_n}$. The conjunction means that the applicant possesses all the skills $v_1, \ldots, v_n$ at the same time. A job offer $O = \{v_1, \ldots, v_n\}$ is translated into $o = a_{v_1} \cap \ldots \cap a_{v_n}$. It represents the skills that required for the job. The matching value of an job offer O and an application A is the result of the probabilistic query $QP(o|a)$:

$$match(O, A) = QP_{DB}(o|a). \tag{3}$$

The formula gives the probability of that the applicant possesses the skills that required for the job (supposed that the constructed DB is consistent).

Example 2.5 shows a part of the transformed ontology from Fig. 1 and the matching value of the $Offer_1$ and $Application_1$.

Example 2.5 (Probabilistic matching).

$$\begin{aligned} DB = \{ & \\ & \#\ ontology\ edges && extra\ edges\ (l = w, u = 1.0) \\ & (it(R)|ide(R))[1,1], && (eclipse(R)|java(R))[0.6, 1.0], \\ & (ide(R)|eclipse(R))[1,1], && (nb(R)|java(R))[0.7, 1.0], \\ & \quad \vdots && \quad \vdots \\ & (ml(R)|xml(R))[1,1], && (xml(R)|html(R))[0.7, 1.0] \quad \} \end{aligned}$$

$$Q_{O_1,A_1} = QP(java(a) \wedge nb(a) \wedge xml(a)|java(a) \wedge php(a) \wedge eclipse(a)).\ \delta = 0.51$$

The next algorithm shows how the probabilistic model and the probabilistic matching can be used to find the most suitable applicant for a job. It works similarly to the *MaximalLengthMatching* algorithm, but it construct a DB from G first (line 1). Then, instead of extending the offer and the applications, it translates them to probabilistic sentences (line 2, 3, 5) as described above. Next, the algorithm computes the matching values by solving the corresponding probabilistic queries (line 7), and it stores the most suitable application in A_m (line 8).

Algorithm 2. ProbabilisticMatching

Input: a graph $G = (V, E = \{E_O \cup E_E\})$, a job offer $O \subseteq V$, and a set of applications $\boldsymbol{A} = \{A_1, A_2, ..., A_n\}$.
Output: the most suitable application A_m for the job

```
DB ← constructDBFrom(G)
O' ← translate(O)
A_m ← A[0], max ← translate(A[0])
for i = 1 → n do
    A' ← translate(A[i])
    Q_{O',A'} = ∃(φ_{O'}(a')|φ_{A'})[X, Y]
    if solve(Q_{O',A'}, P) > solve(Q_{O',max}, P) then
        A_m ← A[i], max ← A'
    end if
end for
return A_m
```

3 Comparison

We compared the presented methods on the job offer and applications from Example 2.1. The skills, their hierarchy, and the conditional dependencies are shown in Fig. 1. The matching values of the applications to the job can be seen in the value columns and the order of the applications based on the different methods can be seen in the order columns in Table 1, where PM, UO, ML, PrM denote the Perfect Matching, the Matching using ontology edges, the Maximal Length Matching and the Probabilistic Matching, respectively. The probabilistic matching values were computed with SPIRIT [23] and PIT [24].

We investigated the matching values of the different algorithms from multiple aspects see [30–32]. We examined whether there is some kind of regularity among the values of the algorithms. We compared how the methods sort the applicants, and how the methods distinguish the applicants from each other. We tried to abstract from the concrete examples and to describe general observation about the algorithms.

Table 1. Comparison of the different matching values on the offer {*Java*, *Netbeans*, *XML*} and the applications from Example 2.1

Application	PM		UO		ML		PrM_{E_O}		$PrM_{l=w,u=1}$		$PrM_{l=u=w}$	
	value	order	value	order	value	order	value	order	value	order	value	order
A_1	0.33	3,4	0.63	3,4	0.93	2	0.20	4	0.51	2	0.51	3
A_2	0.66	1,2	0.88	1	0.96	1	0.50	1	0.70	1	0.70	1
A_3	0.33	3,4	0.63	3,4	0.68	4	0.22	3	0.32	4	0.14	4
A_4	0.66	1,2	0.75	2	0.75	3	0.43	2	0.49	3	0.54	2

3.1 Perfect Matching vs. Matching Using Ontology Edges

As we saw, the Perfect Matching method assigns the same values to too many applicants as it defines too strictly the matching between an offer and an application. In Sect. 2.2 we introduced a specialization relation among the skills which forms a hierarchy from the skills. We can use that additional knowledge to find the most suitable applicant for a job. We extended the job offer and the applications using the edges of the ontology, and then we applied the perfect matching on the extended sets. This method was called the matching using ontology edges. If only the applications were extended, the matching values on the extended sets would always be greater or equal than on the original sets. Formally, let O an job offer and let A an application, then

$$PM(O, A) \leq UO_{apps}(O, A), \tag{4}$$

where UO_{apps} means that only the applications are extended. It is because the application appears only in the numerator in Eq. 1. However, if the job offer is also extended, the inequality above generally does not hold as the offer appears in the denominator as well. For this reason, the two methods can give different order between the applicants. Note, that the extension of the job offer is reasonable when the skills in the job offer are more specific as in the applications. In this case, we would not get different result from the result of the perfect matching if only the applications were extended.

3.2 Matching Using Ontology Edges vs. Maximal Length Matching

The maximal length matching is a generalization of the matching that uses the ontology edges to extend the applications and the job offer. It also extends the original sets, however, it takes into account the so called extra edges as well. The extra edges could form cycles in the hierarchy graph but it does not affect the computation. Note, that the following connection immediately follows from the definitions:

$$UO(O, A) \leq ML(O, A). \tag{5}$$

Furthermore, if no extra edge is given, then $UO = ML$. However, the order of the applicants can be changed using the two methods as it can be seen in Table 1. For example, A_1 has the lowest matching value in UO, but in ML it has the second greatest value because of the $(Java, Netbeans)$ and $(PHP, HTML, XML)$ paths.

3.3 Probabilistic Matchings

The PrM method uses a totally different strategy to compute the matching value. We tried three different versions; the results are shown in Table 1. In of PrM_{E_O}, only the ontology edges were translated into probabilistic sentences while in the other two cases the extra edges were also translated; in $PrM_{l=w,u=1}$, the lower bound of a sentence was equal to the weight of the edge that the constraint was

generated from and the upper bound of the sentence was 1; and in $PrM_{l=u=w}$, both the lower and the upper bound of a sentence were also equal to the weight of the edge.

Generally, the matching values of the three versions are not comparable because of the selection of the underlying probabilistic model. Each algorithm selects a probabilistic model that satisfies the set of sentences generated from the edges, and that has the maximum entropy value among such models. However, the sets of the sentences of the three versions are different from each other. I.e., $PrM_{l=w,u=1}$ added extra constraints to PrM_{E_O} that were generated from the extra edges, and $PrM_{l=u=w}$ added further constraints that came from the limitation of the upper bounds as well. As expected, the entropy of the maximum entropy model of the first version was the greatest (81.68), the entropy of the second model was the next (16.48), and the entropy of third model was the lowest (16.37).

The different probabilistic models give different matching values and it is not guaranteed that the models preserve the order of the applicants. For example, the applicant A_1 is the forth best in PrM_{E_O}, is the second in $PrM_{l=w,u=1}$ and is the third in $PrM_{l=u=w}$ in Table 1.

However, adding extra constraints results in that the algorithms assign the same value to fewer applicants, therefore they distinguish them from each other. It is because while we have no reason to expect one event rather than another, all the possible events should be assigned the same probability as the principle of indifference states.

3.4 Matching Using Ontology Edges vs Probabilistic Matching

When the skills form a hierarchy and no extra edges are given, then the matching using ontology edges and the probabilistic matching PrM_{E_O} can be used.

Unfortunately, there is no connection between the orders of the applicants that the two methods give as Table 1 shows. Although the table suggests that the values of ML are always greater than the values of PrM_{E_O}, but it is generally not true. In that example, all the required skills in the offer and the skills in the applications were selected from the bottom of the hierarchy, therefore the extensions covered large parts of the ontology and the intersections contained many elements. Table 2 shows how the same applications match to another offer which is $Offer_2 = \{IDE, OOP, XML\}$. This offer contains skills from the inner nodes of the ontology as well. It can be seen in Table 2 that all the probabilistic matching methods gave higher values for A_3 and A_4 than the ML method.

3.5 Maximal Length Matching vs. Probabilistic Matching

When the skills form a hierarchy and there are extra edges given as well, then the maximal length matching and the probabilistic matchings $PrM_{l=w,u=1}$ and $PrM_{l=u=w}$ can be used. As we saw the maximal length matching is a generalization of the matching using ontology edges which uses the extra edges too.

Table 2. Comparison of the different matching values on the offer $\{IDE, OOP, XML\}$ and the applications from Example 2.1

Application	PM		UO		ML		PrM_{E_O}		$PrM_{l=w,u=1}$		$PrM_{l=u=w}$	
	value	order	value	order	value	order	value	order	value	order	value	order
A_1	0	3,4	0.67	4	0.95	1,2	0.40	4	0.70	3,4	0.70	3,4
A_2	0	3,4	0.83	1,2,3	0.95	1,2	0.50	3	0.70	3,4	0.70	3,4
A_3	0.33	1,2	0.83	1,2,3	0.87	3	0.89	1	0.92	1	0.85	2
A_4	0.33	1,2	0.83	1,2,3	0.83	4	0.87	2	0.86	2	0.88	1

The $PrM_{l=w,u=1}$ and $PrM_{l=u=w}$ methods also use the extra edges in the transformations, and $PrM_{l=u=w}$ generates additional constraints to the linear optimization problem of $PrM_{l=w,u=1}$ that comes from the limitation of the upper bounds.

As it can be seen in Table 1 and in Table 2, these methods can give totally different orders, and the matching values are incomparable. However, our experiments suggests that the probabilistic matching algorithms can distinguish more applicants from each other than the *ML* method.

4 Related Work

Semantic matchmaking has become a widely investigated topic recently, due to broad applicability in todays competitive business environment. Its origins go back to Vague query answering, proposed by Motro [19] that was an initial effort to overcome limitations of relational databases, using weights attributed to several search variables. More recent approaches along these lines aim at extending SQL with "preference" clauses (Kießling [12]).

Our main focus in the present paper is facilitating the management of available human resources' competencies. Fully or partially automated techniques were developed (see Colucci et al. [4], Bizer et al. [2], Malinowski et al. [18]).

Several matchmaking approaches exist in the literature that could be applied for matching job applications for job offers. Text based information retrieval techniques such as database querying and similarity between weighted vectors of terms were used by Veit et al. [26]. Ontology based skill profile matching was considered in many papers. Lau and Sure [13] propose an ontology based skill management system for eliciting employee skills and searching for experts within an insurance company. Liu and Dew [15] gives a system that integrates the accuracy of concept search with the flexibility of keyword search to match expertise within academia. Colucci et al. [4] proposes a semantic based approach to the problem of skills finding in an ontology based framework. They use description logic inferences to handle background knowledge and deal with incomplete information. They use profile descriptions sharing a common ontology, our approach is based on this, as well. A fundamental difference between the aforementioned works and our paper is that they facilitate search for matching profiles, while we focus on ranking already given applications.

Di Noia et al. [6] places matchmaking on a consistent theoretical foundation using description logic. They define matchmaking as "an information retrieval task whereby queries (also known as demands) and resources (also known as supplies) are expressed using semi-structured data in the form of advertisements, and task results are ordered (ranked) lists of those resources best fulfilling the query." They also introduce match types and rank individual profiles using penalty function. However, they do not apply the filter approach used in our paper.

Fuzzy techniques are introduced in Ragone et al. [22] where they consider a peer-to-peer e-marketplace of used cars. Also a form of logic programming is applied using fuzzy extension of Datalog. Papers [6,22] contain extensive lists of further references concerning practical algorithms, related areas of multiobjective decision making, logic programming, description logic, query reformulation and top-k query answering.

The manuscript [29] uses exact match in ontologies extended with Euclidean-like distance or similarity measure. They apply different levels of given skills, furthermore the job offer may contain "nice-to-have requirements." In our paper we do not apply different grade levels of skills, instead, we consider them separate skills, so we "blow up" the ontology.

Finally, the use of filters in the ontology hierarchy lattice was initiated by Popov et al. [21].

5 Conclusion

In this paper we described the problem of the semantic matching by examples from the field of human resources management, namely matching job offers with applications. However, the presented methods can be used for other fields as well. We investigated the problem from different aspects with different models.

First, we represented the offers and the applications with set of skills, and we introduced the perfect matching. It is a naive approach that computes the matching value of a job offer and an application based on the intersection of two sets. However, it could not be able to sufficiently distinguish the applicants from each other because of its simplicity.

Next, we defined a specialization relation on the skills and built an ontology over them which was represented with directed graph. Then, we presented a method that can use this additional knowledge to find the best applicant for a job. It extends the set of skills of both the job offer and the applications with the more general skills that are available from the original sets on the edges of the ontology. And on the extended sets the perfect matching is already applicable. Beside the ontology edges, we introduced extra edges as well that express conditional dependencies between skills. And then, we generalized the extension of the sets of skills of the applications to use the extra edges too.

In Sect. 2.4, we presented the probabilistic models and we showed how the ontology edges and extra edges can be translated into probabilistic sentences, and how the problem of the semantic matching can be translated into probabilistic

queries. Two different approaches were discussed as the edges can be handled. We used the maximum entropy model to answer the probabilistic queries as it adds the lowest amount of additional information to the system when the given information is incomplete.

Finally, we compared the presented methods from various aspects. We investigated whether there is any connection of the matching values that give the methods. Furthermore, we examined how the methods sort the applicants and how they can distinguish the applicants form each other. We showed that the $PM(O, A) \leq UO_{apps}(O, A)$ and the $UO(O, A) \leq ML(O, A)$ connections hold between the matching values of the perfect matching PM the matching using ontology edges UO and the maximal length matching for arbitrary job offer O and application A. However, our results revealed that the algorithms can give totally different order among the same applicants matching to the same job offer. Therefore, it highly depends on the field which algorithm gives the most suitable order.

References

1. Baader, F., Nutt, W.: Basic description logic. In: Description Logic Handbook, pp. 43–95 (2003)
2. Bizer, C., Heese, R., Mochol, M., Oldakowski, R., Tolksdorf, R., Eckstein, R.: The impact of semantic web technologies on job recruitment processes. In: Proceedings of the 7th International Conference Wirtschaftsinformatik (2005)
3. Brickley, D., Ramanathan, V.G.: RDF Schema 1.1: W3C Recommendation 25, February 2014
4. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., Mongiello, M.: Concept abduction and contraction in description logics. In: Proceedings of the 16th International Workshop on Description Logics (DL 2003). CEUR Workshop Proceedings, vol. 81 (2003)
5. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
6. Di Noia, T., Di Sciascio, E., Donini, F.M.: Semantic matchmaking as non-monotonic reasoning: a description logic approach. J. Artif. Intell. Res. **29**, 269–307 (2007)
7. European Dictionary of Skills and Competences. http://www.disco-tools.eu
8. Hájek, P.: Mathematics of Fuzzy Logic. Kluwer Academic Publishers, Dordrecht (1998)
9. International Standard Classification of Education. http://www.uis.unesco.org/Education/Pages/international-standard-classification-of-education.aspx
10. International Standard Classification of Occupations (2008)
11. Kern-Isberner, G., Lukasiewicz, T.: Combining probabilistic logic programming with the power of maximum entropy. Artif. Intell. **157**(1), 139–202 (2004)
12. Kießling, W.: Foundations of preferences in database systems. In: Proceedings of the 28th International Conference on Very Large Data Bases, pp. 311–322 (2002)
13. Lau, T., Sure, Y.: Introducing ontology-based skills management at a large insurance company. In: Proceedings of the Modellierung, pp. 123–134 (2002)
14. Levandowsky, M., Winter, D.: Distance between sets. Nature **234**(5), 34–35 (1971)

15. Liu, P., Dew, P.: Using semantic web technologies to improve expertise matching within academia. In: Proceedings of I-KNOW 2004, pp. 370–378 (2004)
16. Liu, L., Li, K.: Fuzzy filters of BL-algebras. Inf. Sci. **173**(1), 141–154 (2005)
17. Lloyd, J.W.: Foundations of Logic Programming. Springer, Heidelberg (2012)
18. Malinowski, E., Zimányi, E.: Hierarchies in a multidimensional model: from conceptual modeling to logical representation. Data Knowl. Eng. **59**(2), 348–377 (2006)
19. Motro, A.: VAGUE: a user interface to relational databases that permits vague queries. ACM Trans. Off. Inf. Syst. **6**(3), 187–214 (1988)
20. Pedrycz, W., Gomide, F.: An Introduction to Fuzzy Sets: Analysis and Design. MIT Press, Cambridge (1998)
21. Popov, N., Jebelean, T.: Semantic Matching for Job Search Engines: A Logical Approach. Technical report 13–02, Research Institute for Symbolic Computation, JKU Linz (2013)
22. Ragone, A., Straccia, U., Di Noia, T., Di Sciascio, T., Donini, F.M.: Fuzzy matchmaking in e-marketplaces of peer entities using Datalog. Fuzzy Sets Syst. **160**(2), 251–268 (2009)
23. Rödder, W., Meyer, C.-H.: Coherent knowledge processing at maximum entropy by SPIRIT. In: Proceedings of the 12th International Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann Publishers Inc. (1996)
24. Schramm, M., Ertel, W.: Reasoning with probabilities, maximum entropy: the system PIT and its application in LEXMED. In: Inderfurth, K., Schwödiauer, G., Domschke, W., Juhnke, F., Kleinschmidt, P., Wäscher, G. (eds.) Operations Research Proceedings, vol. 1999, pp. 274–280. Springer, Heidelberg (2000)
25. Schramm, M., Greiner, M.: Non-Monotonic Reasoning on Probability Models: Indifference, Independence. Inst. für Informatik (1995)
26. Veit, D., Müller, J., Schneider, M., Fiehn, B.: Matchmaking for autonomous agents in electronic marketplaces. In: Proceedings of the International Conference on Autonomous Agents 2001, pp. 65–66 (2001)
27. Wygralak, M.: Cardinalities of Fuzzy Sets. Springer, Heidelberg (2003)
28. Zadeh, L.A.: Fuzzy sets. Inf. Control **8**(3), 338–353 (1965)
29. https://users.soe.ucsc.edu/~darrell/tmp/search_subm06.pdf
30. http://www.renyi.hu/~sali/sms/ide_oop_xml.xlsx
31. http://www.renyi.hu/~sali/sms/java_nb_xml.xlsx
32. http://www.renyi.hu/~sali/sms/queries.xlsx

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

Instruction to printer	*Textual mark*	*Marginal mark*
Leave unchanged	··· under matter to remain	
Insert in text the matter indicated in the margin		New matter followed by or
Delete	through single character, rule or underline or through all characters to be deleted	or
Substitute character or substitute part of one or more word(s)	through letter or through characters	new character or new characters
Change to italics	under matter to be changed	
Change to capitals	under matter to be changed	
Change to small capitals	under matter to be changed	
Change to bold type	under matter to be changed	
Change to bold italic	under matter to be changed	
Change to lower case	Encircle matter to be changed	
Change italic to upright type	(As above)	
Change bold to non-bold type	(As above)	
Insert 'superior' character	through character or where required	or under character e.g. or
Insert 'inferior' character	(As above)	over character e.g.
Insert full stop	(As above)	
Insert comma	(As above)	
Insert single quotation marks	(As above)	or and/or or
Insert double quotation marks	(As above)	or and/or or
Insert hyphen	(As above)	
Start new paragraph		
No new paragraph		
Transpose		
Close up	linking characters	
Insert or substitute space between characters or words	through character or where required	
Reduce space between characters or words	between characters or words affected	