A Set-Based Approach to Qualitative and Quantitative Estimation of Competence

Tudor Jebelean and Nikolaj Popov

Abstract We present a novel strategy for making non-trivial matches of position openings versus job applications. The strategy forms a logically based framework for the development of a real system which ensures a correct and an efficient matching. In our framework, we translate the notions of knowledge and skills into abstract mathematical sets. The sets are classified and ordered, as the skills are in the real life. The operations of finding similarities between skills are then made as set intersections based on the mathematical apparatus for set operations. Moreover, this theoretical model gives not only the possibility to decide if two objects are similar or not, but also it defines a distance between them. Based on that distance, one may compute the effort needed for a job applicant to fulfill precise job descriptions.

1 Introduction

Human Resource Management and in particular competence management is considered nowadays as highly important in the developed world. This is the main motivation for computerizing and in general automating many of the operations needed for a successful operation of the HR-Management.

We are convinced that in order to increase the quality of the HR-Management automation, mathematical logic and formal methods should play a bigger role during the design and the composition of its software components.

Semantic Matching (in this paper we say simply matching) is a technique that takes two structures and produces a mapping between elements of the two structures

Nikolaj Popov
Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, e-mail: popov@risc.uni-linz.ac.at

Tudor Jebelean
Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, e-mail: jebelean@risc.uni-linz.ac.at

that correspond semantically to each other (see e.g. [3],[4]). In order, however, to define the semantics of our objects (skills and competencies), we need a precise hierarchical taxonomy and this is broadly discussed in 3.

In this paper we are mostly concerned with matching of position openings versus job applications. In order to automate this process of matching, we first formalize the notions of required skills and available qualifications. We translate the notions of knowledge and skills into abstract mathematical sets. Although, it is very difficult to enumerate the elements of those sets, the sets do exist and we take them as parameters which are then used for further computations. The detailed expose is presented at 4.

Moreover, the paper at hand offers an algorithm for semantic matching which is not only able to make non-trivial matches but has the possibility to cope with varying proficiency level of competencies – the details are at 5.1 and 5.2.

The strategy forms a logically based framework for the development of a real system which ensures a correct and an efficient matching.

2 Trivial Match

The main contribution of our work into the field is the specific organization of the matching. Matching, in general, is the core action in automated job search engines.

There are two kinds of objects we want to match against each other: job applicants and job offers. Since job applicants are humans, they may not be equivalent to a list of competencies, however, in order to find a suitable job, any job applicant is required to provide a list of his/her competencies. Human specific issues are highly important, however they are out of the scope of this paper.

The first kind of matching objects are the lists of skills, provided by the applicants – we call them lists of competencies. To every applicant is associated a list of his/her competencies. The second kind of matching objects are the job offers provided by companies. To each job offer is associated a list of required skills, which is in fact again a list of competencies.

For given collection of applicants and job offers, job search engines are trying to find matches between them. In particular, there are two kinds of queries one can perform with the search engine:

- Given a concrete applicant, find all the job offers matching his/her qualifications.
- Given a concrete job offer, find all the applicants matching its requirements.

For the first case the matching is done by taking as an input the list of competencies of the applicant and finding all the job offers whose lists of requirements are subsets of the initial list.

For example, there is an applicant John Smith who knows Java, C, English and Chinese. For him, any job offer whose list of requirements is a subset of the above four skills, is a match.

John Smith	Java, C, English, Chinese

Similarly for the second case, there is a match when for a given list of required skills, the applicants list is a superset. For example, a company ABC Inc has a job offer for a programmer knowing C, English and Chinese, then John Smith is a match.

ABC Inc.	C, English, Chinese

The match described above is nowadays a build-in in all the job search engines. For the purpose of our project, we call it *trivial match*.

The motivation of our study comes from the fact that in many cases, in the real-world examples, it is very unlikely to find a trivial match, but still, some *partial* match is available. For example, a company BCD Inc has a job offer for a programmer knowing C++, English and Chinese.

BCD Inc.	Java, C++, English, Chinese

John Smith is then not a *trivial* match, because he does not have C++ in his list of competencies. If there are no other *trivial* matches, the company BCD Inc would need to search for job applicants who have certain percentage of the required list of competencies.

In the literature (see e.g. [5]) this kind of ranking non-perfect candidates is known as *gap analysis*. Gap analysis is in general a technique for determining the steps to be taken in moving from a current state to a desired future-state [2].

For example, let us have one more job applicant Peter James, who knows Java, Prolog, English and Chinese.

Peter James	Java, Prolog, English, Chinese

After performing *gap analysis*, we would obtain that John Smith and Peter James are equally close to the required list of competencies, namely, both of them obey the language requirements, and both of them obey 50% of the programming skills. However, as the reader may see, the two job applicants are not equally close to the required list of competencies, because when required C++, knowing C is much better than knowing Prolog.

One of the main goals of this research is the automation of such kind of observations: John Smith is a better candidate, because his knowledge of C makes him familiar with C++.

3 Taxonomy and Inheritance of Skills

The structure and the organization of the available database is nowadays considered as one of the key points in the field of data manipulation and in general in IT. In order to provide a consistent and well structured system of skills DB, the skills are clustered upon similarities, in a hierarchical taxonomy.

As a starting point for our hierarchical taxonomy, we take the structure as defined in the DISCO project (see e.g. [7], [8]). The next figure contains (due to space restrictions only some part of) the DISCO taxonomy for IT.

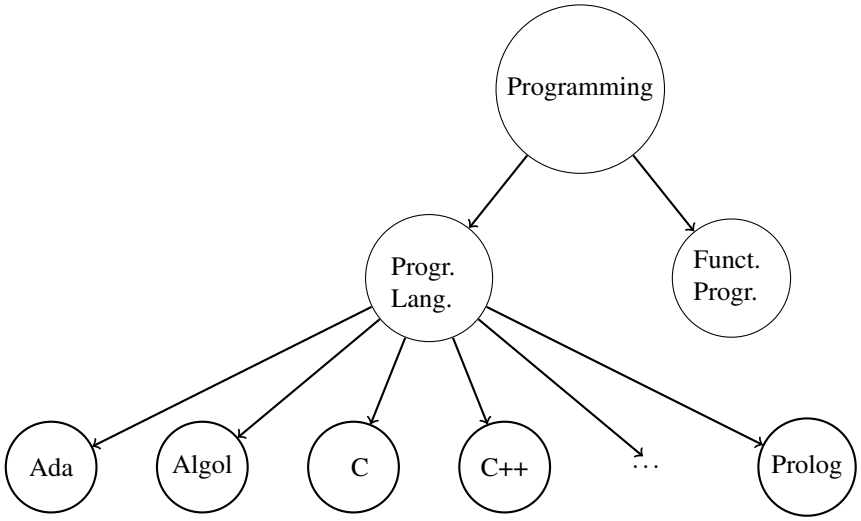

Fig. 1 DISCO IT

In order to illustrate and also apply our ideas, we first need to make some modifications on the existing taxonomy tree, by further clustering of skills upon similarities. Note that after the modification, the structure is not anymore tree-like, but it becomes oriented-graph-like structure, due to its multiple inheritance.

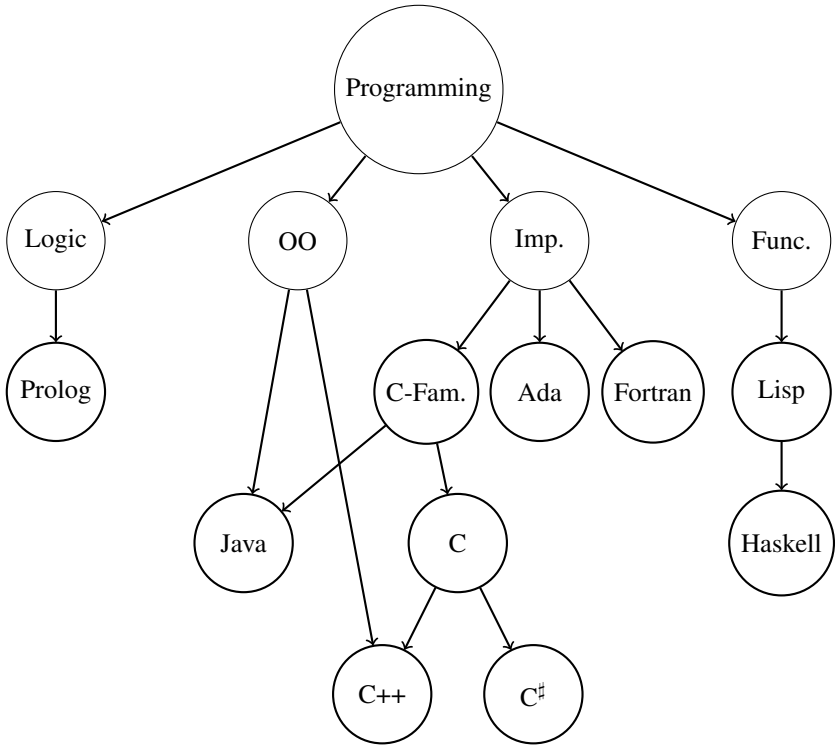

Fig. 2 Modified IT graph

There is a general rule, well observed in [6], which is valid for such hierarchical taxonomy structures, namely that the semantic differences between upper level concepts are bigger than those from lower levels.

For example, the difference between Object Oriented languages and Imperative languages is bigger than the difference between C++ and Java.

Another interesting observation is that not only the leaves may represent skills on their own, but also some of the lower nodes. For example, we have various kinds of vehicle drivers: bus driver, truck driver, pick-up driver, car driver, etc. The next figure contains (due to space restrictions only some part of) the DISCO taxonomy for vehicle drivers.

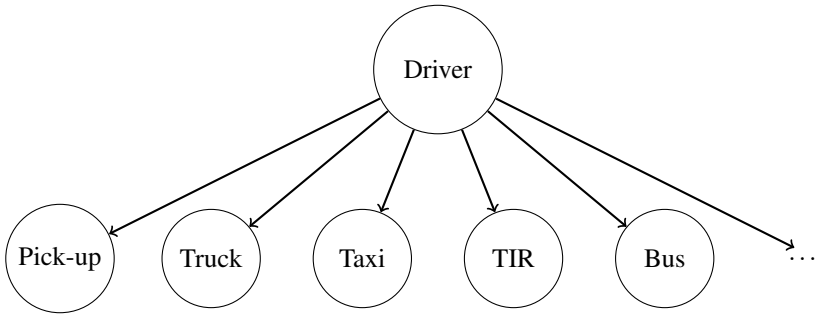

Fig. 3 DISCO Drivers

After modifying the existing taxonomy tree, by further clustering of skills upon similarities and inheritance, we get the following structure.

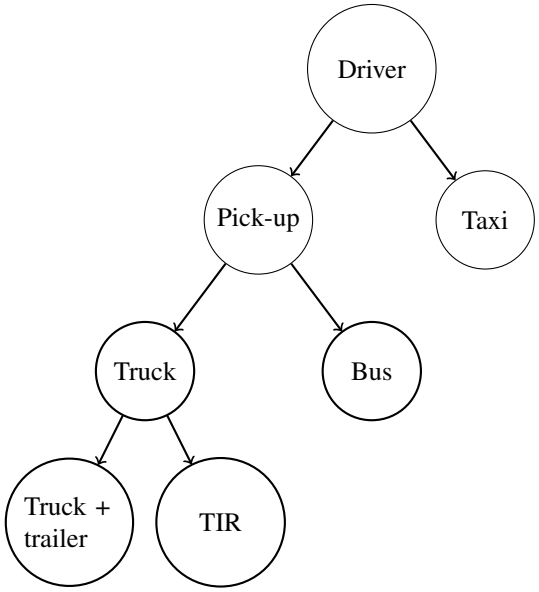

Fig. 4 Modified Drivers Graph

The observation is that not only the leaves, e.g., *Truck + Trailer driver*, but also some of the nodes, e.g., *Pick-up driver* represent skills on their own. Moreover, the following implications are valid:

- Everybody who is a *Truck + Trailer driver* is a *Truck driver* as well;
- Everybody who is a *Truck driver* is a *Pick-up driver* as well;
- etc.

The general rule is: For any available skill, given by an applicant, all the upper nodes in the taxonomy tree representing skills on their own, become skills of the applicant; similarly, for any required skill given by a job offer, all the lower nodes and leaves become replacements of the initially required skill.

This observation gives the possibility of an automated deduction of skills being not listed among the skills of an applicant. For example, if an applicant is a *Truck + Trailer driver* he/she is also *Truck driver*, *Pick-up driver*, etc.

It also gives the possibility of an automated reformulation of a job offer, by replacing some of the skills by ones from lower nodes and leaves. For example, if a job offer requires a *Pick-up driver*, then any *Truck + Trailer driver*, *Truck driver*, *Bus driver*, etc. may serve as well.

4 Set Representation and Semantics

In this section we develop the semantics of the modified taxonomy tree including its multiple inheritance properties. The tree, or more precisely the oriented graph, consists of nodes, leaves and oriented edges (arrows). Nodes from which there are no outgoing arrows are called leaves.

We represent any node as a set. Although it is very difficult to enumerate the elements of most of the nodes, they do exist. For example, the node *Imperative* is a set containing a specific knowledge for all the imperative languages.

In general, for given leaves, any non-leave node should be a subset of the intersection of its descendants.

In the example with the modified IT taxonomy tree, the node *Imperative* is the intersection of the *C-family*, the *Fortran*, etc. nodes. Intuitively, all the imperative languages have something in common, which is very specific only for them. Then, the notion of imperative languages is exactly this intersection of all of them.

Another example is the Natural Languages taxonomy tree, divided into groups and families.

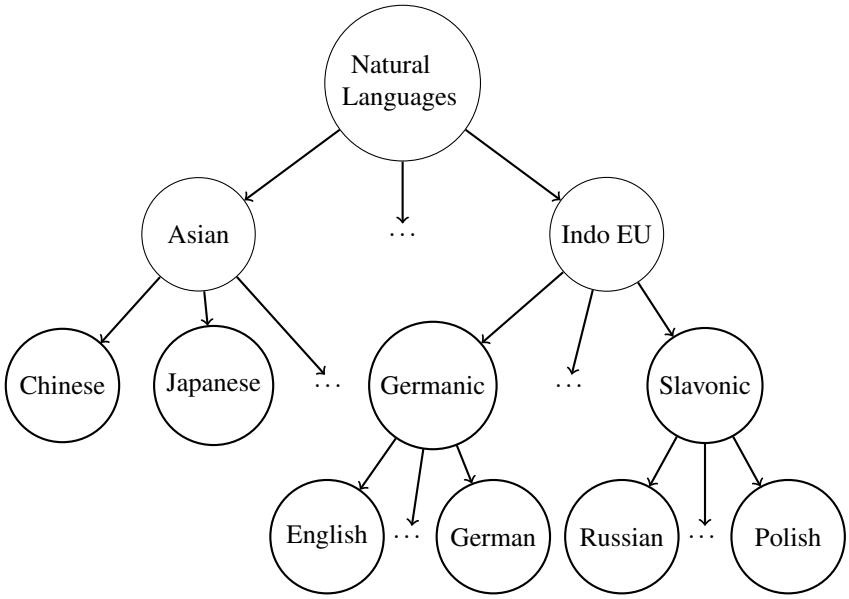

Fig. 5 Modified Natural Languages Graph

The group of Germanic languages is represented by the corresponding node *Germanic* which itself is the intersection of all the languages from the group. In fact, all Germanic languages have something in common, e.g., interrogative sentences use inversion, which is not the case for the group of Slavonic languages.

In order to allow easier computations, we associate sets to the nodes in the semantic tree – each node gets its associated set. The choice is based on the set analysis – which set is a subset of which set. The construction is done on levels. The level of the root element is 0. The level of any other node from the graph is the number corresponding to the number of arrows from the root element to the node in question. Since, we consider an oriented graph, which in general may not be a tree, there will be nodes with several paths from the root. For them, as level is taken the number of arrows of the longest path from the root. For example, the level of the node Java from the modified IT taxonomy is 3.

Determining the associated sets of the nodes is done in the following way:

- Starting from the root, we assume the set has just one element – say $\{S_0\}$.
- For a given level, consider in turn all the nodes. The associated set of each node is obtained by taking the union of the associated sets of the parents and by adding new element to the set. For example, we have a node whose parent nodes associated sets are $\{S_0, \ldots, S_p\}$ and $\{S_0, \ldots, S_q\}$, its associated set will be $\{S_0, \ldots, S_q, S_p, S_n\}$, where S_n is a new symbol.

In order to illustrate the method of associating sets, we consider the example with the modified IT taxonomy.

Programming	$\{S_0\}$
Logic	$\{S_0, S_1\}$
OO	$\{S_0, S_2\}$
Imperative	$\{S_0, S_3\}$
Functional	$\{S_0, S_4\}$
Prolog	$\{S_0, S_1, S_5\}$
C-Family	$\{S_0, S_3, S_6\}$
Ada	$\{S_0, S_3, S_7\}$
Fortran	$\{S_0, S_3, S_8\}$
Lisp	$\{S_0, S_4, S_9\}$
Java	$\{S_0, S_3, S_6, S_2, S_{10}\}$
C	$\{S_0, S_3, S_6, S_{11}\}$
Haskell	$\{S_0, S_4, S_9, S_{12}\}$
C++	$\{S_0, S_3, S_6, S_{11}, S_2, S_{13}\}$
C$^\sharp$	$\{S_0, S_3, S_6, S_{11}, S_{14}\}$

Let us develop the table with associated sets for the Natural Languages taxonomy – we will need it in a later example for demonstrating the matching algorithm. In order to distinguish the sets from the IT and the Natural Language taxonomies, we use now L-names instead of the S-names.

Natural Languages	$\{L_0\}$
Asian	$\{L_0, L_1\}$
Indo-EU	$\{L_0, L_2\}$
Chinese	$\{L_0, L_1, L_3\}$
Japanese	$\{L_0, L_1, L_4\}$
Germanic	$\{L_0, L_2, L_5\}$
Slavonic	$\{L_0, L_2, L_6\}$
English	$\{L_0, L_2, L_5, L_7\}$
German	$\{L_0, L_2, L_5, L_8\}$
Russian	$\{L_0, L_2, L_6, L_9\}$
Polish	$\{L_0, L_2, L_6, L_{10}\}$

After having introduced the semantics of the taxonomy, we go to defining the matching algorithm.

5 Matching Algorithm and Computation

In semantic search, the matching algorithm is the essence of the whole business. For nontrivial matches, we need to make some king of ranking in order to determine who (or which) is better than who (or which).

5.1 Basic Algorithm

In order to formalize our lists of competencies and skills, we introduce the following sets:

$\mathbb{REQUIRE}$

For any job offer, we associate a set $\mathbb{REQUIRE}$, defined as the union of all the associated sets for the requirements for that job offer. Intuitively, this set represents the skills a job offer is requiring.

$\mathbb{AVAL}$

For any applicant, we associate a set $\mathbb{AVAL}$, defined as the union of all the associated sets for the competencies for that applicant. Intuitively, this set represents the competencies an applicant possesses.

$\mathbb{MISS}$

For any pair applicant and job offer, we associate a set $\mathbb{MISS}$, defined as the difference between the required skills and the available competencies: $\mathbb{REQUIRE}\setminus\mathbb{AVAL}$. Intuitively, this set represents the required skills, which the applicant does not possesses.

In order to perform the ranking, we introduce a rational function $\mathbb{MATCH}$ which takes two arguments: applicant and job offer. The values of the function are between 0 and 1, with the property: The higher the $\mathbb{MATCH}$, the better the match is. Depending on the value, we say:

- $\mathbb{MATCH} = 1$ trivial match;
- $\mathbb{MATCH} = 0$ no match;
- $0 < \mathbb{MATCH} < 1$ partial match.

The function $\mathbb{MATCH}$ is defined as follows:

$$\mathbb{MATCH} = 1 - \frac{\|\mathbb{MISS}\|}{\|\mathbb{REQUIRE}\|}.$$

Let us consider again the example for John Smith and compute his match for the BCD Inc. job offer.

John Smith	Java, C, English, Chinese
BCD Inc.	Java, C++, English, Chinese

Then according to our taxonomy, and the associated sets, we get the following:

John Smith	$\mathbb{AVAL}$	$\{S_0, S_2, S_3, S_6, S_{10}, S_{11}, L_0, L_1, L_2, L_3, L_5, L_8\}$
BCD Inc.	$\mathbb{REQUIRE}$	$\{S_0, S_2, S_3, S_6, S_{11}, S_{13}, L_0, L_1, L_2, L_3, L_5, L_8\}$
Smith vs. BCD	$\mathbb{MISS}$	$\{S_{13}\}$

$$\mathbb{MATCH} = 1 - \frac{\|\{S_{13}\}\|}{\|\{S_0, S_2, S_3, S_6, S_{11}, S_{13}, L_0, L_1, L_2, L_3, L_5, L_8\}\|} = \frac{11}{12}.$$

Let us now compute the match of Peter James for the BCD Inc. job offer.

Peter James	Java, Prolog, English, Chinese
BCD Inc.	Java, C++, English, Chinese

Then according to our taxonomy, and the associated sets, we get the following:

Peter James	$\mathbb{AVAL}$	$\{S_0, S_2, S_3, S_6, S_{10}, S_1, S_5, L_0, L_1, L_2, L_3, L_5, L_8\}$
BCD Inc.	$\mathbb{REQUIRE}$	$\{S_0, S_2, S_3, S_6, S_{11}, S_{13}, L_0, L_1, L_2, L_3, L_5, L_8\}$
James vs. BCD	$\mathbb{MISS}$	$\{S_{11}, S_{13}\}$

$$\mathbb{MATCH} = 1 - \frac{\|\{S_{11}, S_{13}\}\|}{\|\{S_0, S_2, S_3, S_6, S_{11}, S_{13}, L_0, L_1, L_2, L_3, L_5, L_8\}\|} = \frac{10}{12}.$$

The observation that John Smith is more suitable than Peter James for the BCD Inc. job offer is now formally confirmed. Intuitively, as pointed out earlier, this is so, because when required C++, knowing C is much better than knowing Prolog.

With the next example, we demonstrate that an applicant without having any of the required skills may still be better than another one having some of the required skills.

CDE Inc.	English, Chinese
Cun Cun	Chinese
Boris Orlovsky	German, Japanese

According to our taxonomy, and the associated sets, for Cun Cun we get the following:

Cun Cun	$\mathbb{AVAL}$	$\{L_0, L_1, L_3\}$
CDE Inc.	$\mathbb{REQUIRE}$	$\{L_0, L_1, L_2, L_3, L_5, L_7\}$
Cun vs. CDE	$\mathbb{MISS}$	$\{L_2, L_5, L_7\}$

$$\mathbb{MATCH} = 1 - \frac{\|\{L_2, L_5, L_7\}\|}{\|\{L_0, L_1, L_2, L_3, L_5, L_7\}\|} = \frac{1}{2}.$$

Let us now compute the match of Boris Orlovsky for the CDE Inc. job offer.

Boris Orlovsky	$\mathbb{AVAL}$	$\{L_0, L_1, L_2, L_4, L_5, L_8\}$
CDE Inc.	$\mathbb{REQUIRE}$	$\{L_0, L_1, L_2, L_3, L_5, L_7\}$
Orlovsky vs. CDE	$\mathbb{MISS}$	$\{L_4, L_8\}$

$$\mathbb{MATCH} = 1 - \frac{\|\{L_4, L_8\}\|}{\|\{L_0, L_1, L_2, L_3, L_5, L_7\}\|} = \frac{2}{3}.$$

Although the second applicant does not have any of the required skills, his expertise is closer to the required due to the fact that he knows one Asian language and also one language from the Germanic group.

5.2 "Must have" Requirements and Proficiency Level of Competencies

In order to allow companies to define obligatory requirements, we make here a small modification of the algorithm. If for example, the CDE Inc. would have required Chinese as a must, then the second applicant would not have even entered the ranking.

The modification is namely: for any given job offer, before computing the value of the match of an applicant, check if there are "must have" requirements. If yes, then ignore the applicant if he/she does not obey all of them. If he/she does obey all of them, then compute the match based on the basic algorithm.

Another modification we want to introduce here is the possibility of varying the proficiency level of the competencies of an applicant. For example, if an applicant is at advanced level of English, say 70%, then we give this factor to all the ingredients of the associated set, e.g.,

$$\{70\%\ L_0, 70\%\ L_2, 70\%\ L_5, 70\%\ L_7\}.$$

The cardinality function than changes and counts the elements with their weights, e.g.,

$$\|\{70\%\ L_0, 70\%\ L_2, 70\%\ L_5, 70\%\ L_7\}\| = 4\ *\ 70\% = 2.8,$$

instead of

$$\|\{L_0, L_2, L_5, L_7\}\| = 4.$$

It may happen that different competencies whose associated sets are overlapping, have different factors. In such cases, we take the maximal factor for the relevant ingredients. For example, consider the two competencies 70% English and 40% German. Then the common ingredients L_0, L_2, L_5 are counted with 70% and the cardinality of the union of the associated sets is

$$\|\{70\%\ L_0, 70\%\ L_2, 70\%\ L_5, 70\%\ L_7, 40\%\ L_8\}\| = 4\ *\ 70\%\ +\ 40\% = 3.2.$$

The match algorithm remains the same, the only difference is that the cardinality function is adopted to the varying proficiency level.

In order to illustrate our idea, we consider the example with John Smith, slightly modified to John Smith-junior, versus the job opening BCD Inc.

John Smith-junior	Java 70%, C 50%, English, Chinese
BCD Inc.	Java, C++, English, Chinese

Then we get the following:

$\mathbb{AVAL}$	$\{70\%\ S_0,\ 50\%\ S_2,\ 70\%\ S_3,\ 70\%\ S_6,\ 70\%\ S_{10},\ 50\%\ S_{11},\ L_0, \ldots\}$
$\mathbb{REQUIRE}$	$\{S_0, S_2, S_3, S_6, S_{11}, S_{13}, L_0, L_1, L_2, L_3, L_5, L_8\}$
$\mathbb{MISS}$	$\{30\%\ S_0,\ 50\%\ S_2,\ 30\%\ S_3,\ 30\%\ S_6,\ 30\%\ S_{10},\ 50\%\ S_{11},\ S_{13}\}$

$$\mathbb{MATCH} = 1 - \frac{\|\{30\%\ S_0,\ 50\%\ S_2,\ 30\%\ S_3,\ 30\%\ S_6,\ 30\%\ S_{10},\ 50\%\ S_{11},\ S_{13}\}\|}{\|\{S_0, S_2, S_3, S_6, S_{11}, S_{13}, L_0, L_1, L_2, L_3, L_5, L_8\}\|} =$$

$$= 1 - \frac{3.2}{12} = \frac{8.8}{12}.$$

As expected, John Smith-junior has lower match than John Smith whose match is $11/12$. Moreover, if we compute the match of John Smith using the last algorithm,

John Smith	Java 100%, C 100%, English 100%, Chinese 100%
BCD Inc.	Java, C++, English, Chinese

we get the same value as before – $11/12$. This is in order to demonstrate that the modified algorithm is applicable in this case as well.

5.3 Overqualification

Overqualification is the state of being skilled or educated beyond what is necessary for a job. For some companies overqualification is considered as a negative factor, whereas for some others, e.g., academic or research institutions, this is usually considered as beneficial. We develop a specific strategy for computing an additional summand corresponding to the overqualification. Whether it will be taken for increasing the match or decreasing the match or not taken at all is a matter of companies decision and our algorithm covers all the three possibilities.

$\mathbb{OVER}$

For any pair applicant and job offer, we associate a set $\mathbb{OVER}$, defined as the difference between the available competencies and the required skills: $\mathbb{AVAL}\backslash\mathbb{REQUIRE}$. Intuitively, this set represents the skills which the applicant does possesses but are not required by the job offer.

If the overqualification is ignored then the matching function (in fact, its component for expressing the overqualification) $\mathbb{C}$ remains unchanged.

If the overqualification is taken positively then the component is defined as follows:

$$\mathbb{C}_{\mathbb{OVER}+} = \frac{\|\mathbb{OVER}\|}{\|\mathbb{REQUIRE}\| + \|\mathbb{OVER}\|}.$$

Note that the function $\mathbb{C}_{\mathbb{OVER}+}$ ranges between 0 and 1, with $\mathbb{C}_{\mathbb{OVER}+} \to 1$ if there is much overqualification and $\mathbb{C}_{\mathbb{OVER}+} = 0$ if there is no overqualification.

If the overqualification is taken negatively then the component is defined as follows:

$$\mathbb{C}_{\mathbb{OVER}-} = \frac{\|\mathbb{REQUIRE}\|}{\|\mathbb{REQUIRE}\| + \|\mathbb{OVER}\|}.$$

Note that the function $\mathbb{C}_{\mathbb{OVER}-}$ ranges between 0 and 1, with $\mathbb{C}_{\mathbb{OVER}-} \to 0$ if there is much overqualification and $\mathbb{C}_{\mathbb{OVER}+} = 1$ if there is no overqualification.

In order to illustrate how the overqualification component may change the value of matching, we consider the example with John Smith-junior versus the job opening DEF Inc.

John Smith-junior	Java 70%, C 50%, English, Chinese
BCD Inc.	Java, English, Chinese

Then we get the following:

$\mathbb{AVAL}$	$\{70\%\ S_0,\ 50\%\ S_2,\ 70\%\ S_3,\ 70\%\ S_6,\ 70\%\ S_{10},\ 50\%\ S_{11},\ L_0, \ldots\}$
$\mathbb{REQUIRE}$	$\{S_0, S_2, S_3, S_6, S_{10}, L_0, L_1, L_2, L_3, L_5, L_8\}$
$\mathbb{OVER}$	$\{50\%\ S_{11}\}$

When considered as positive, the value of the overqualification component is:

$$\mathbb{C}_{\mathbb{OVER}+} = \frac{\|\{\ 50\%\ S_{11}\}\|}{\|\{S_0, S_2, S_3, S_6, S_{10}, L_0, L_1, L_2, L_3, L_5, L_8\}\| \ +\ \|\{\ 50\%\ S_{11}\}\|} =$$

$$= \frac{0.5}{11.5} = \frac{5}{115}.$$

As expected, John Smith-junior is not much overqualified, and his $\mathbb{C}_{\mathbb{OVER}+} = \frac{5}{115}$ is not big – here applies the rule: The bigger and closer to one the better.

When considered as negative, the value of the overqualification component is:

$$\mathbb{C}_{\mathbb{OVER}-} = \frac{\|\{S_0, S_2, S_3, S_6, S_{10}, L_0, L_1, L_2, L_3, L_5, L_8\}\|}{\|\{S_0, S_2, S_3, S_6, S_{10}, L_0, L_1, L_2, L_3, L_5, L_8\}\| \ + \ \|\{\ 50\%\ S_{11}\}\|} =$$

$$= \frac{11}{11.5} = \frac{110}{115}.$$

As expected, John Smith-junior is not much overqualified, and his $\mathbb{C}_{\mathbb{OVER}-} = \frac{110}{115}$ is big – the rule is again: The bigger and closer to one the better.

5.4 Importance of Skills

Another extension of our research is providing the possibility for the job offers to have defined the importance of the skills. For example, we modify the job offer for a programmer by the BCD Inc into:

BCD Inc.	English – must have, C ++ – 70%, Java – 10%, Chinese – 20%

As in the previous modification, for any given job offer, before computing the value of the match of an applicant, check if there are "must have" requirements. If yes, then ignore the applicant if he/she does not obey all of them. If he/she does obey all of them, then compute the match based on the upcoming algorithm.

Assume now, that we are given a job offer not containing "must have" requirements and all the competencies listed there (say $r_1, r_2, \ldots r_n$) are accomplished with certain percentage (say $p_1, p_2, \ldots p_n$). Assume also that the sum of the respective percentages is 100, i.e., $p_1 + p_2 + \ldots + p_n = 100$.

Then for each single requirement r_i we compute the *i-th* match component m_i based on the basic algorithm:

$$m_i = 1 - \frac{\|miss_i\|}{\|r_i\|}.$$

The value of the final match $\mathbb{MATCH}$ is the sum of all the individual match components multiplied by the respective percentages:

$$\mathbb{MATCH} = m_1.p_1\% \ + \ m_2.p_2\% \ + \ \ldots \ + m_n.p_n\%.$$

Let us consider again the example for John Smith and compute his match for the modified job offer of BCD Inc.

John Smith	Java, C, English, Chinese
BCD Inc.	English – must have, C ++ – 70%, Java – 10%, Chinese – 20%

At first, John Smith passes trough the "must have" requirement. Then for each of the three requirements (C ++, Java and Chinese) we compute the match.

John Smith	$\mathbb{AVAL}$	$\{S_0,S_2,S_3,S_6,S_{10},S_{11},L_0,L_1,L_2,L_3,L_5,L_8\}$
BCD Inc.	r_1 (C++)	$\{S_0,S_2,S_3,S_6,S_{11},S_{13}\}$
Smith vs. BCD	$miss_1$	$\{S_{13}\}$

$$m_1 = 1 - \frac{\|\{S_{13}\}\|}{\|\{S_0,S_2,S_3,S_6,S_{11},S_{13}\}\|} = \frac{5}{6}.$$

John Smith	$\mathbb{AVAL}$	$\{S_0,S_2,S_3,S_6,S_{10},S_{11},L_0,L_1,L_2,L_3,L_5,L_8\}$
BCD Inc.	r_2 (Java)	$\{S_0,S_2,S_3,S_6,S_{10}\}$
Smith vs. BCD	$miss_2$	$\{\}$

$$m_2 = 1 - \frac{\|\{\}\|}{\|\{S_0,S_2,S_3,S_6,S_{11},S_{13}\}\|} = 1.$$

John Smith	$\mathbb{AVAL}$	$\{S_0,S_2,S_3,S_6,S_{10},S_{11},L_0,L_1,L_2,L_3,L_5,L_8\}$
BCD Inc.	r_3 (Chinese)	$\{L_0,L_1,L_3\}$
Smith vs. BCD	$miss_3$	$\{\}$

$$m_3 = 1 - \frac{\|\{\}\|}{\|\{L_0,L_1,L_3\}\|} = 1.$$

Summing now the match, we obtain:

$$\mathbb{MATCH} = \frac{5}{6}.70\% + 10\% \ + 20\% = \frac{53}{60}.$$

Last but not least, we want to point out that the presented here basic algorithm and its modifications, work fine with trivial matches. Therefore, the algorithm is not only applicable when no trivial match exists, but it is a general one for semantic job search engines.

As an outcome of the matching we get also an estimation of the effort needed for the job applicant in order to fulfill precisely a job description – this is the $\mathbb{MISS}$ set.

For the time being, we assume the sizes of the small portions of knowledge (the S_i sets) are always equal to each other. As pointed earlier in this paper, enumerating

precisely the tiny portions of knowledge is very difficult and therefore we use them here as parameters only.

However, a better way of measuring the size of the S_i sets is the corresponding amount of time need for transmitting the relevant knowledge. Then, in the matching algorithm, instead of counting the S_i sets, we would need to sum up the corresponding amount of time, say in teaching hours.

6 Comprehensive Matching: A Generalization

In the reality, when appropriateness of a candidate against a job offer has to be computed, there are various factors to be taken into account and the matching of the skills is just one of them. Normally, a certain level of a specific education is required. For example, for computer scientists, very often MSc or equivalent university degree is needed. Not only the level, but also the speciality is needed, e.g., Computer Science. In fact, a smart search engine should be capable of discovering that MSc in Technical Mathematics is very similar to MSc in Computer Science and also MSc in Informatics. Furthermore, MSc in a given area implies BSc in the same area and is implied by a relevant PhD in the field.

In order to include all the aspects and variety of situations, we propose the same matching algorithm applied to education. For the successful application we assume we have developed the appropriate taxonomy and inheritance for education. Then we make basically the same set representation of the tiny portions of education and define the semantics as before.

Taking into account all these factors, e.g., skills, education, overqualification, relevant experience, etc. we introduce the following formula:

$$\mathbb{APPROP} = \sum_{k=1}^{n} w_k c_k,$$

where the c_k are the concrete components, and the w_k are their weights. Each c_k is a function (the match for the concrete component) which takes two arguments – concrete job offer and concrete applicant, and computes a value ranging between 0 and 1. The weights w_k correspond to the importance of the components, and their values are defined by the company providing the job offer. Moreover, the sum of all the weights is equal to 1, i.e.,

$$\sum_{k=1}^{n} w_k = 1.$$

Based on the above restrictions, we observe that $\mathbb{APPROP}$ is a function which takes two arguments – concrete job offer and concrete applicant, and it also computes a value ranging between 0 and 1.

The value of the $\mathbb{APPROP}$ function is taken as the final value of appropriateness, or fitness for the pair job offer and job applicant.

Let us consider once again the example for John Smith and compute his appropriateness for the modified job offer of BCD Inc.

John Smith	
Skills	Java, C, English, Chinese
Education	PhD in Informatics
Experience	14 years

BCD Inc.	
Skills	English – must have, C ++ – 70%, Java – 10%, Chinese – 20%
Education	MSc in Computer Science
Experience	at least 10 years
Overqual	taken positively
Weights	Skills – 60%, Edu – 20%, Experience – 10%, Overqual – 10%

In order to compute the appropriateness of John Smith for the job offer of BCD Inc., that is to evaluate the $\mathbb{APPROP}$ function, we need first to compute the four components $c_1 \dots c_4$ corresponding to Skills, Education, Experience and Overqualification. As computed before, $c_1 = \frac{53}{60}$. The component corresponding to Education $c_2 = 1$ comes from the fact that PhD in Informatics implies MSc in Computer Science. We assume, the last observation is done automatically based on appropriate taxonomy for education. The component corresponding to Experience $c_3 = 1$ comes from the fact that John Smith has more than 10 years of experience, as required. The Overqualification component c_4 is computed by the function

$$\mathbb{C}_{\mathbb{OVER}+} = \frac{\|\mathbb{OVER}\|}{\|\mathbb{REQUIRE}\| + \|\mathbb{OVER}\|}.$$

When generating the sets $\mathbb{OVER}$ and $\mathbb{REQUIRE}$ we consider all the tiny ingredients from all the components, namely Skills, Education and Experience. In our case, the only ingredients in $\mathbb{OVER}$ come from the fact that John Smith holds PhD instead of MSc degree.

Assume

$$\mathbb{C}_{\mathbb{OVER}+} = \frac{2}{25} = c_4.$$

After having all the components, we are now ready to compute the appropriateness function

$$\mathbb{APPROP} = 60\% \ \frac{53}{60} + 20\% \ 1 + 10\% \ 1 + 10\% \ \frac{2}{25} = 0.838.$$

7 Related Research

Applications of semantic matching in HR-Management have been made for more than a decade. One of the first approaches presented at [1] is based on Description Logics formalization and reasoning. There the idea of ranking non-perfect job applicants has been introduced. Moreover, a prototype system has been developed.

A similar to our approach is presented at [6], where finding similarities is based on hierarchical taxonomy structures, however, their approach is essentially based on tree-like taxonomy and no multiple inheritance of concepts is allowed.

8 Conclusions and Further Work

The approach to semantic matching for job search engines presented here is a result of a theoretical work with the aim of developing a practical job search portal.

In fact, the method we have presented here is on its way to be fully implemented by an Industry-Academia consortium [9] including:

- Differential Psychology DIPS, University of Graz, Austria;
- Institute for Application Oriented Knowledge Processing FAW, Johannes Kepler University of Linz, Austria;
- Meine Karriere, Austria;
- Research Institute for Symbolic Computation RISC, Johannes Kepler University of Linz, Austria;
- Schenkenfelder Tailor-made Web Applications, Austria;
- Software Competence Center Hagenberg SCCH, Austria;
- Solunic Web Solutions, Austria.

As we pointed out earlier (at the end of subsection 5.2), a better way of measuring the sizes of the small portions of knowledge (the S_i sets) is by taking the corresponding amount of time need for transmitting the relevant knowledge. Such time estimations may be obtained from professional schools, universities, professional organizations, etc. After having such estimations, in the matching algorithm, instead of counting the S_i sets, we would need to sum up the corresponding amount of time, say in teaching hours.

Although the examples presented here appear to be relatively simple, they already demonstrate the usefulness of our approach in the general case. We aim at extending these experiments to more practical examples, because these are not more complex from the mathematical point of view.

References

1. S. Colucci, T. Di Noia, E. Di Sciascio, F. M. Donini, M. Mongiello, and, M. Mottola.
   A Formal Approach to Ontology-Based Semantic Match of Skills Descriptions. In *Journal of Universal Computer Science, Vol. 9, no. 12*, pp. 1437–1454, 2003.
2. Gap Analysis. http://www.businessdictionary.com/definition/gap-analysis.html.
3. Fausto Giunchiglia and Pavel Shvaiko.
   Semantic Matching. In *Technical Report DIT-03-013*, University of Trento, 2003.
4. F. Giunchiglia, P. Shvaiko, and M. Yatskevich.
   Semantic Matching. In *Encyclopedia of Database Systems*, pp. 2561–2566, 2009.
5. M. O. Lundqvist, K. Baker, and S. Williams.
   An Ontological Approach to Competency Management. In *Proceedings of iLearn 2007 (International Conference on Semantic Systems)*, pp.1-4, Paris, France, 29-31 January 2007.
6. M. Mochol, H. Wache, and, L. J. B. Nixon.
   Improving the Accuracy of Job Search with Semantic Techniques. In W. Abramowicz, editor, *Proceedings of the* 10*th 10th International Conference on Business Information Systems (BIS 2007)*, volume 4439 of *LNCS*, pp. 301–313, Poznan, Poland, April 2007.
7. Heidemarie Müller-Riedlhuber.
   The European Dictionary of Skills and Competences (DISCO): an example of usage scenarios for ontologies. In *Proceedings of i-Semantics 2009 (International Conference on Semantic Systems)*, Graz, 2-4 September 2009.
8. Heidemarie Müller-Riedlhuber, and Jörg Markowitsch, editors.
   DISCO Dictionary of Skills and Competencies. ISBN 978-3-902277-41-1. 3s, Vienna 2008.
9. OntoJob Project. http://www.ontojob.at/project/.