Improving the recruitment process through ontology-based querying

Malgorzata Mochol
Freie Universität Berlin
Institut für Informatik
AG Netzbasierte Informationssysteme
Takustr. 9, 14195 Berlin
Germany
mochol@inf.fu-berlin.de

Holger Wache
Vrije Universiteit Amsterdam
Artificial Intelligence Department
de Boelelaan 1081a, 1081HV Amsterdam
The Netherlands
holger@cs.vu.nl

Lyndon Nixon
Freie Universität Berlin
Institut für Informatik
AG Netzbasierte Informationssysteme
Takustr. 9, 14195 Berlin
Germany
nixon@inf.fu-berlin.de
http://nbi.inf.fu-berlin.de

Abstract: While using semantic data can enable improved retrieval of suitable jobs or applicants in a recruitment process, cases of inconsistent or overly specific queries which would return no results still have to be dealt with. In this paper the extension of a semantic job portal with a novel query relaxation technique is presented which is able to return results even in such cases. Subsymbolic methods estimate the (quantitative) similarity between job or applicant descriptions. Symbolic approaches allow a more intuitive way to formulate and handle preferences and domain knowledge. But due to their partial preference order they can not rank all results in practice like subsymbolic approaches. In this paper we propose a query relaxation method which combines both methods. This method demonstrates that by having data based on formal ontologies, one can improve the retrieval also in a user-friendly way.

1 Introduction

Human resources management, like many business transactions, is increasingly taking place on the Internet. In Germany 90% of human resource managers rated the Internet as an important recruitment channel [Ke05] and over half of all personnel recruitment is the result of online job postings [Mo03]. Although job portals are an increasingly important source for job applicants and recruitment managers, they still exhibit shortcomings in retrieval and precision as the stored job offers are in syntactic formats, i.e. searches are subject to the ambiguities of natural language and job descriptions and characteristics lack relations to similar or interdependant concepts. Particularly, queries which are over-

specified or inconsistent return no matches while relevant job offers could actually still be found if the consistency or specificity problem were to be resolved. Extending job offers in a portal with semantics can enable improved search and precision based on the use of ontologies and semantic matchings. A further extension of such a semantics-based job portal with novel query relaxation techniques additionally solves the problem of inconsistent and overspecified querying, which can be expected to occur in a real world setting.

In this paper we report on such results from the German national project *Wissensnetze* (Knowledge Nets[1]) and *European Network of Excellence Knowledge Web*[2]. Having identified the importance of online recruitment processes for businesses today, Section 1.1 will introduce requirements that arise when seeking to match appropriate applicants to vacancies or vice versa. *Wissensnetze* is working together with a German job portal provider to develop an innovative new approach to online recruitment based on the use of semantically annotated job and applicant descriptions. In Section 2 we introduce the web-based prototype of a semantic job portal that has been developed in this project. To meet the identified requirements, we introduce two semantic-based approaches which rely on the formal logical basis of ontology languages. Semantic matching techniques (Section 3) are applied to determine the level of similarity between a job and an applicant. However, we note that this technique alone can not meet real world requirements, which include the possibility of overly specific or inconsistent queries. Hence, in a co-operation supported by *Knowledge Web*, we are extending the prototype with state of the art query relaxation techniques (Section 4). As a result we are in a position to evaluate both the emerging semantic technique as well as the real world viability of the semantic job portal (Section 5). In general, we are able to demonstrate the value of ontology-based approaches to a typical business activity as well as the added value of the use of a formal logic based approach in that new semantic techniques are easily applicable to Semantic Web systems to solve open requirements.

1.1 Requirements

Currently, a large number of online job portals divide the online labour market into information islands and making it close to impossible for a job seeker to get an overview of all relevant open positions. In spite of a large number of portals employers still publish their openings on a rather small number of portals assuming that a job seeker will visit multiple portals while searching for open positions. Alternatively, companies can publish job postings on their own website [Mü00]. This way of publishing, however, makes it difficult for job portals to gather and integrate job postings into their database. Furthermore, the quality of search results depends not only on the search and index methods applied but also on the processability of the used web technologies and the quality of the automatic interpretation of the company-specific terms occurring in the job descriptions. The deficiencies of a website's machine processability result from the inability of current web technologies, such as HTML, to semantically annotate the content of a given website. Therefore, com-

[1] `http://wissensnetze.ag-nbi.de`
[2] `http://knowledgeweb.semanticweb.org`

puters can easily display the content of a HTML site, but they lack the ability to interpret the content properly.

In our opinion using Semantic Web technologies in the domain of online recruitment can overcome the problems of distribution, heterogeneity and machine non-processability of existing job information and substantially increase market transparency, lower transaction costs and speed up the procurement process for businesses. For this reason, in *Wissensnetze* we have developed job portal which is based on Semantic Web technologies as a basis for exploring the potential of the Semantic Web from a business and a technical viewpoint by examining the effects of the deployment of Semantic Web technologies for particular application scenarios and market sectors. Every scenario includes a technological component which makes use of the prospected availability of semantic technologies in a perspective of several years and a deployment component assuming the availability of the required information in machine-readable form. The combination of these two projections allows us, on the one hand, to build e-business scenarios for analysis and experimentations and, on the other hand, to make statements about the implications of the new technology on the participants of the scenario in the current early stage of development.

In the first version of the *semantic job portal* we concentrated on the modelling of the human resource ontology and development of a matching approach for comparisons of applicant profiles and job openings with focus on skills, occupations as well as industry sector descriptions [BHM+05]. The further specification of the complete job portal contains the comparison of the applicants and openings not only under consideration of skills and their levels but also professional experiences of a job seeker in relation to the requirements of the hiring company. We want to express not only the duration of a particular experience (3 years java programming) but also to deliver these job applications which maybe do not fit in 100% to the defined requirements but are still acceptable for the employer (3 years instead of 5 years industry experience). Furthermore, to verify the consistency of the job opening descriptions we also have to avoid the definition of nonsensical requirements like job postings which demand only very young (under 25) yet highly qualified people (with at least 10 years work experience). Following this, we need an additional method which starts checking the data with the strongest possible query that is supposed to return the "best" answers satisfying most of the given conditions and then weaken the query if the returned result set is either empty or contains unsatisfactory results.

Since we have been working very close with one German job search engine we were able to define several exemplary use cases which focuses on the definition of such requirements and issues. From the practical point of view the use-cases may represent the kind of queries which happen in the real world. However from the scientific point of view these use-cases are challenges to the techniques which we want to apply.

When implementing a (Semantic Web) job portal the requirements of the system depend on the meaningful use cases which are derived by the industrial project partner from its day to day business practices within the HR-domain. To clarify the still outstanding problems we will briefly present one of such use cases which (at first view) seems to be quite simple. However if we look closer and try to represent the data in an ontology or satisfy the requirements in the semantic portal we will meet some difficulties which at the same time show the complexity of such "simple" queries.

We are looking for a person which:

1. has an degree in computer science
2. wants to work in software consulting and development,
3. is an expert in C, Java, PHP, UML, .Net and WindowsNT,
4. has worked for at least 5 years in an industrial and 5 year in a research project,
5. should have experience as project or team manager,
6. should be not older then 25

This example serve as a guideline and a thread in the rest of the article.

2 Semantic Job Portal

In a Semantic Web-based recruitment scenario the data exchange between employers, applicants and job portals is based on a set of vocabularies which provide shared terms to describe occupations, industrial sectors and job skills [MPB06]. In this context, the first step towards the realization of the Semantic Web e-Recruitment scenario was the creation of a *human resources ontology (HR-ontology)*. The ontology was intended to be used in a job portal by allowing not only a uniform representation of job postings and job seeker profiles but first of all the semantic matching (a technique that combines annotations using controlled vocabularies with background knowledge about a certain application domain) in job seeking and procurement tasks. Another important requirement was development of the Semantic Web-based job portal concerning the user needs under consideration of the already common practice in the industry. Accordingly to this specification we focused on how vocabularies can be derived from standards already in use within the recruitment domain and how the data integration infrastructure can be coupled with existing non-RDF human-resource systems.

In the process of ontology building we first identified the sub-domains of the application setting (skills, types of professions, etc.) and several useful knowledge sources covering them (approx. 25)[BMW04]. As candidate ontologies we selected some of the most relevant classifications in the area, deployed by federal agencies or statistic organizations: German Profession Reference Number Classification (BKZ), Standard Occupational Classification (SOC), German Classification of Industrial Sector (WZ2003), North American Industry Classification System (NAISC), German version of the Human Resources XML (HR-BA-XML) and Skill Ontology developed by the KOWIEN Project[SBA03]. These sources are represented in different formats and languages with various levels of the formality (textual descriptions, XML-schemes, DAML) and cover different domains at different precision levels. Since these knowledge sources were defined in different languages (English/German) we first generated (depending on the language) lists of concept names. Except for the KOWIEN ontology, additional ontological primitives were not supported by the candidate sources. In order to reduce the computation effort required to compare

and merge similar concept names we identified the sources which had to be completely integrated to the target ontology. For the remaining sources we identified several thematic clusters for further similarity computations. For instance BKZ was directly integrated to the final ontology, while the KOWIEN skill ontology was subject of additional customization. To have an appropriate vocabulary for a core skill ontology we compiled a small conceptual vocabulary (15 concepts) from various job portals and job procurement Web sites and matched them against the comprehensive KOWIEN vocabulary. Next, the relationships extracted from KOWIEN and various job portals were evaluated by HR experts and inserted into the target skill sub-ontology. The resulting conceptual model was translated mostly manually to OWL (since except for KOWIEN the knowledge sources were not formalized using a Semantic Web representation language) [PBM05].

We have been still evaluating and refining the preliminary HR-ontology for the purpose of further development and to calculate the costs of reusing existing sources. About 15% of the total engineering time were spent on source gathering and about 30% were spent on customizing the selected source ontologies. Several ontologies have been fully integrated into the resulting ontology, while KOWIEN and the XML-based sources required additional customization effort. Although the entire integration took up over 45% of the total engineering time the reusing classification schemes like BKZ or WZ2003, which did not require any customization effort, definitely resulted in significant cost savings, while guaranteeing a comprehensive conceptualization of occupations and industrial sectors, respectively. The last phase of the building, refinement and evaluation process will require 10% of the overall time. The aggregation of knowledge from different domains and the evaluation of available, large-sized ontologies were tedious and time-consuming. Although, the benefits of using standard classification systems in this application setting outweigh the costs of the ontology reuse. The reuse process could be significantly optimized in terms of costs and quality of the outcomes if provided the necessary technical support.

Having modelled the HR-ontology and prepared the RDF-Repository to store applicant profiles and job description, we developed the matching engine which as the core component of the system plays the crucial role in the procurement process. The function of the matching engine is focused on in the following chapter.

3 Semantic matching

Semantic matching is a technique which combines annotations using controlled vocabularies with background knowledge about a certain application domain. In our prototypical implementation, the domain specific knowledge is represented by concept hierarchies like skills, skill level classification, occupational classification, and a taxonomy of industrial sectors. Having this background knowledge of the recruitment domain (i.e. formal definition of various concepts and specification of the relationships between these concepts) represented in a machine-understandable format allows us to compare job descriptions and applicant profiles based on their semantic similarity [PC95] instead of merely relying on the containment of keywords like most of the contemporary search engines do.

In our HR-scenario, our matching approach[3] utilizes metadata of job postings and candidate profiles and as the matching result, a ranked list of best candidates for a given job position (and vice versa) is generated.

Inside both a job posting as well as an applicant profile we group pieces of information into "*thematic clusters*", e.g. information about skills, information regarding industry sector and occupation category, and finally job position details like salary information, travel requirement, etc. Each thematic cluster from a job posting is to be compared with the corresponding cluster from an applicant profile (and the other way round). The total similarity between a candidate profile and a job description is then calculated as the average of the cluster similarities. The *cluster similarity* itself is computed based on the similarities of semantically corresponding concepts from a job description and an applicant profile. The *taxonomic similarity* between two concepts is determined by the distance between them which reflects their respective positions in the concept hierarchy. Following this, the distance `d` between two given concepts in a hierarchy e.g. `.Net` and `DCOM` (cf. Fig. 1) represents the path from one concept to the other over the closest common parent. The semantic differences between upper level concepts are bigger than those between concepts on lower hierarchy levels (in other words, two general concepts like `object oriented programming languages` and `imperative procedural programming languages` are less similar than two specialized like `C#` and `Java`) and the distance between siblings is greater than the distance between parent and child (`d(Java,C#) > d(Java,PureObjectOrientedLanguages)`).

Since we also provide means for specifying *competence levels* (e.g. expert or beginner) in applicants profiles as well as job postings we compare these levels in order to find the best match. Furthermore, our approach also gives employers the opportunity to specify the importance of different job requirements. The concept similarity is then justified by the indicated weight (i.e. the similarity between more important concepts) like the skills crucial for a given job position and will have greater influence on the similarity between a job position posting and an applicant profile.

Having the example from Section 1.1 we can apply the developed semantic matching engine to compare the requirements defined within a job opening with the job applicant descriptions (and another way round comparing the applicants profiles with the job descriptions). The results of the comparisons are presented in the form of a ranked list where each applicant profile can be separately viewed and considered in detail (cf. Fig. 1).

The approach described above allows comparisons of job openings with applicant profiles based on verifying occupation and industry descriptions, skills and their competence levels as well as some general information like salary and job type. Hence, the prototype can satisfy the first three points of the specification from the above mentioned job description (cf. Sec. 1.1) we are not able to deliver an answer to the other requirements especially concerning the minimal required experiences in different projects or experiences as team manager. To tackle this problem we have decided to extend our prototype by applying not only the semantic matching technique but also using the query relaxation methods to compare the job description with applicants.

[3]For further information about used matching framework SemMF see [OB05]

Comparison: Application - Job offer

Matching result (71.6% Total similarity)

Details of job offer (59.0% similarity, weighted with 0.2)

	Job offer	Applicant	Similarity
Travel desired:	no	no	100.0%
Job type:	part time	full time	0.0%
Salary:	35000 € / year	39000 €/year	77.1%

Details of occupation (88.0% similarity, weighted with 0.2)

	Job offer	Applicant	Similarity
Industrial sector:	Software consulting and development	Software consulting	100.0%
Occupation:	Computer sciences assistant	Computer sciences assistant - Software technology	76.0%

Required Competencies (70.3% similarity, weighted with 0.6)

Required competencies	Available competencies	Similarity
C (Expert)	C (Medium knowledge)	82.1%
DotNET (Expert)	DCOM (Beginner)	80.1%
WindowsNT (Expert)	CGI (Expert)	59.8%
UML (Expert)	XML (Beginner)	68.2%
Java (Expert)	C (Medium knowledge)	82.1%
PHP (Expert)	CGI (Expert)	59.8%

Figure 1: Matching result

4 Query relaxation

The previous approach basically uses a similarity measure which calculates the similarity between the job posting and candidate profile. Such a function $f(jp, cp) \mapsto [0..1]$ directly provide a ranking between the results because answers which are more similar can be higher ranked. In order to ensure that the job portal ranks certain answers higher than others similarity measures normally can be biased in that way that weights w_i are attached to some parts of the calculation, i.e. $f(p, r) = \sum_{i=1}^{n} w_i * f_i(p, r)$. With such weights users also can ensure that the system will respect his preferences during the ranking.

However, similarity functions also have their drawbacks. Like all subsymbolic approaches, similarity functions do not explain *how* the job posting and the request differ. They only return a value like 0.78 but they do not provide any information how the job posting and the candidate profile differ in detail. For example, they can not explain that the candidate has only three years experiences instead of requested five years.

Furthermore the similarity function is also not able to explain the differences betweens the answers. Another answer with a very close similarity value, say 0.79, may give the impression of a similar good candidate but it may differ on an absolutely different thematic cluster, e.g. the candidate has no experiences in leading a project. The difference with this answer is not obvious and is not explained. The similarity function suggests a ranking

but in fact the result is an unordered listing; a grouping of semantically similar answered would improve the acceptance of and the usability by the user.

On the other hand the user is directly able to specify how he wants to relax his request. The user may specify directly: "if nobody have 5 years industrial experiences then I will also accept 3 years experiences". Furthermore the system can also explain how this set of returned answers is related to the original query, e.g. here comes now the answers not with 5 but with 3 years experiences (cf. Section 1.1).

Such preferences can be specified in symbolic approaches. Advanced methods like [BBD+04] also allow conditional preferences where the preferences depends on some other decisions. However most of these approaches assume implicitly a flat data structures like e.g. a set of variables [BBD+04]. But here we are focussed with the need of the full expressive power of advanced object-centred representation of job descriptions and candidate profiles which may be difficult to encode in symbolic approaches.

In the following we describe an approach which use rewriting rules to capture preference and domain knowledge explicitly and show how this knowledge is used to relax the original query into a set of approximated queries. Rewriting rules are able to work on complex data structures. We propose an approach for query rewriting based on conditional rewriting rules to solve the problem of incorporating domain knowledge and user preferences for similar matching job request and openings. This rewriting relaxes the over-constrained query based on rules in an order defined by some conditions. This has an advantage that we start with the strongest possible query that is supposed to return the "best" answers satisfying most of the conditions. If the returned result set is either empty or contains unsatisfactory results, the query is modified either by replacing or deleting further parts of the query, or in other words relaxed. The relaxation should be a continuous step by step, (semi-)automatic process, to provide a user with possibility to interrupt further relaxations.

Query rewriting with rewriting rules helps to incorporate domain knowledge and user preferences in the semantic matching in an appropriate way. It comes back with a set of rewritten queries. However the results of each rewritten query may be a set of answers which is not ordered. How to order or rank them? For this part we can fall back to the similarity function with their implicitly encoded knowledge and rank the answers for each rewritten query. To summarize query rewriting provides a high level relaxation including grouping the results according the domain knowledge and the user preferences and the similarity function provides some kind of fine tuning when the results in one group is ranked.

Before we investigate concrete relaxation strategies in the context of our example domain, we first give a general definition of the framework for re-writing an RDF query very briefly.

4.1 Formal definition

The RDF data model foresees sets of statements which are in the form of triples [Ha04]. In [DSW06] Dolog et.al. proposed a rule-based query rewriting framework for RDF queries independent of a particular query language which we summarize here. The framework is based on the notion of triple patterns (RDF statements that may contain variables) as the

basic element of an RDF query and represents RDF queries in terms of three sets:

- triple patterns that must be matched by the result (mandatory patterns)
- triple patterns that may be matched by the results (optional triple patterns).
- Conditions in terms of constraints on the possible assignment of variables in the query patterns.

More precisely Dolog et.al. define a (generic) RDF query as a triple of these three sets.

Definition 1 *RDF Query*
Let $\mathcal{T}$ be a set of terms, $\mathcal{V}$ a set of variables, $\mathcal{RN}$ a set of relation names, and $\mathcal{PN}$ a set of predicate names. The set of possible triple patterns $\mathcal{TR}$ is defined as $\mathcal{TR} \subseteq (\mathcal{T} \cup \mathcal{V}) \times (\mathcal{RN} \cup \mathcal{V}) \times (\mathcal{T} \cup \mathcal{V})$. A query Q is defined as the tuple $\langle M_Q, O_Q P_Q \rangle$ with $M_Q, O_Q \in \mathcal{TR}$ and $P_Q \subseteq \mathcal{P}$ where M_Q is the set of mandatory pattern (patterns that have to be matched by the result, O_Q is a set of optional pattern (patterns that contribute to the result but do not necessarily have to match the result) and $\mathcal{P}$ is the set of predicates with name $\mathcal{PN}$, defined over $\mathcal{T}$, and $\mathcal{V}$.

A result set of such a RDF query is set of substitutions. Formally a substitution τ is a list of pairs (X_i, T_i) where each pair tells which variable X_i has to be replaced by $T_i \in \mathcal{T} \cup \mathcal{V}$. A ground substitution replaces variables X_i by a term and not by another variable, i.e. $T_i \in \mathcal{T}$ for all i. The (ground) substitution τ replaces variables in M_Q and O_Q with appropriate terms. If $\tau(M_Q)$ is equal to some ground triples then the substitution is valid. All valid ground substitutions for M_Q plus existing ground substitutions for O_Q constitute answers to the query. Additionally the predicates P_Q restrict these substitutions because only those bindings are valid answers where the predicates, i.e. $\tau(P_Q)$, are also satisfied. The predicates additionally constraint the selection of appropriate triples.

Re-writings of such queries are described by transformation rules $Q \xrightarrow{R} Q^R$ where Q the original and Q^R the rewritten query generated by using R. Rewriting rules consists of three parts:

- A matching pattern represented by a RDF query in the sense of the description above
- A replacement pattern also represented by an RDF query in the sense of the description above
- A set of conditions in terms of special predicates that restrict the applicability of the rule by restricting possible assignments of variables in the matching and the replacement pattern.

Based on the abstract definition of an RDF query, we can now define the notion of a rewriting rule. We define rewriting in terms of rewriting rules that take parts of a query, in particular triple patterns and conditions, as input (PA) and replace them by different elements (RE).

Definition 2 *Rewriting Rule*
A rewriting rule R is a 3-tuple $\langle PA, RE, CN\rangle$ where PA and RE are RDF queries according to Definition 1 and CN is a set of predicates.

For conditions the same constructs as for queries are used where the possible results are also constrained by predicates. Patterns and replacements formally have the same structure like queries. They also consist of a set of triples and predicates. But patterns normally do not address complete queries but only a subpart of a query. Normally the subpart addresses some triples as well as some predicates in the query. In order to write more generic rewriting rules the pattern must be instantiated which is done by an substitution.

Definition 3 *Pattern Matching*
A pattern PA of a rewriting rule R is applicable to a query $Q = \langle M_Q, O_Q, P_Q\rangle$ if there are subsets $M'_Q \subseteq M_Q$, $O'_Q \subseteq O_Q$ and $P'_Q \subseteq P_Q$ and a substitution θ with $\langle M'_Q, O'_Q, P'_Q\rangle = \theta(PA)$.

In contrast to term rewriting systems [BN98] the definition of a query as two sets of triples and predicates differentiate the pattern matching. The identification of the right subpart of the query for the pattern match is simplified because of the use of sets. Only a subset of both sets has to be determined which must be syntactically equal to the instantiated pattern. Please note that due to set semantics, the triples and predicates in the pattern may be distributed over the query.

A re-writing is now performed in the following way: If the matching pattern matches a given query Q in the sense that the mandatory and optional patterns as well as the conditions of the matching pattern are subsets of the corresponding parts of Q then these subsets are removed from Q and replaced by the corresponding parts of the replacement pattern. The application of R is only allowed if the predicates in the conditions of R are satisfied for some variable values in the matching and the replacement pattern.

Definition 4 *Query Rewriting*
If a rewriting rule $R = \langle PA, RE, CN\rangle$

- *matches a query $Q = \langle M_Q, O_Q, P_Q\rangle$ with subsets $M'_Q \subseteq M_Q$, $O'_Q \subseteq O_Q$ and $P'_Q \subseteq P_Q$ substitution θ and*
- *$\theta(CN)$ is satisfied*

then the rewritten query $Q^R = \langle M^R_Q, O^R_Q, P^R_Q\rangle$ can be constructed with $M^R_Q = (M_Q \setminus M'_Q) \cup \theta(M_{RE})$, $O^R_Q = (O_Q \setminus O'_Q) \cup \theta(O_{RE})$ and $P^R_Q = (P_Q \setminus P'_Q) \cup \theta(P_{RE})$ with $RE = \langle M_{RE}, O_{RE}, P_{RE}\rangle$.

The former definition clarifies formally how to generate a rewritten query Q^R from Q with the help of R, i.e. $Q \xrightarrow{R} Q^R$. We denote with $Q^{\mathcal{R}}$ all queries which can generated from Q with all rules $R \in \mathcal{R}$. On each query from $Q^{\mathcal{R}}$ the rewriting rules can be applied again. We denote with $Q^{\mathcal{R}*}$ all queries which can be generated by application of the rules in $\mathcal{R}$ successively.

4.2 Application in the job portal

Each job request and opening is annotated with an RDF description which is a set of triples. A query over these job openings is formulated as triple patterns and a set of conditions that restrict the possible variables bindings in the patterns. Each triple pattern represents a set of triples. The corresponding abstract definition of a query focuses on the essential features of queries over RDF.

To clarify the approach we take the example 1 from the Section 1: someone who has experience in C, Java, PHP, UML, .Net and WindowsNT. Looking for such a person requires from the system to translate this free text description into an instance retrieval problem. The query must be translated into a concept expression. The retrieval process will return all job seekers which belong to that concept expression, i.e. satisfying all the requirement in the concept expression. The following OWL[4] expression shows the concept expression for some person who has experience in some of (the intersectionOf property) the OWL classes C, Java, PHP or UML [5] .

```
<owl:Class rdf:ID="Query">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Person"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Class rdf:about="C"/>
            <owl:Class rdf:about="Java"/>
            <owl:Class rdf:about="PHP"/>
            <owl:Class rdf:about="UML"/>
          </owl:intersectionOf>
        </owl:Class>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasExperience"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

In the following we give some examples for the rewriting rules which use the aforementioned example as a basis.

[4]OWL is an extension of RDF allowing for more expressive features than RDF like number restrictions etc.

[5]Originally we modelled these as nominals (enumerations like Week =Monday, Tuesday, ...). Nominals are instances and classes at the same time. However current DL systems have problems with nominals therefore we use classes in the current approach.

A very simple rewriting rule takes into account the required skill e.g. Java. It relax the some requirements in the experiences, i.e. instead of `JAVA` the `PureObjectOrientedLanguages` or even the `ObjectOrientedLanguages` could be possible weakenings of the original query:[6]

```
pattern(<owl:Class rdf:about="Java"/>)
==>
replace(<owl:Class rdf:about="PureObjectOrientedLanguages"/>)
&& true.
```

This means that whenever anywhere the term representing the Java language in a query appears then it can be replaced by a more general term representing pure object oriented languages, of which Java is one.

Making use of the predicates we can generalize previous rewriting rules and generate generic rules that are guided by information from the ontology. The predicate `subsumed` for example is satisfied when X is more specific than Y. With the following rewriting rule we are able to consider the knowledge in the ontology.

```
pattern(<owl:Class rdf:about="X"/>)
==>
replace(<owl:Class rdf:about="Y"/>)
&& subsumed(X,Y).
```

In the same way some number restrictions can be applied. In our example the requirement that a person has experiences in a five year industrial project is encoded with the help of the (artificial) class `FiveYearsOrMore`. This class represents all Numbers representing years which are larger or equal to five. This class can be replaced by the class `TwoYearsOrMore` which obviously is more general (weaker) then the former. Furthermore we can restrict the replacement in that way that we only allow this for the restriction on property `hasDuration`. The corresponding rewriting rule look like:

```
pattern(<owl:Restriction>
           <owl:onProperty rdf:resource="#hasDuration"/>
           <owl:someValuesFrom>
                <owl:Class rdf:ID="FiveYearsOrMore"/>
           </owl:someValuesFrom>
        </owl:Restriction>)
==>
replace(<owl:Restriction>
            <owl:onProperty rdf:resource="#hasDuration"/>
            <owl:someValuesFrom>
                <owl:Class rdf:ID="TwoYearsOrMore"/>
            </owl:someValuesFrom>
        </owl:Restriction>)
&& true.
```

[6] For the sake of readability the examples are simplified.

The main problem of the re-writing approach to query relaxation is the definition of an appropriate control structure to determine in which order the individual rewriting rules are applied to general new queries. In other words how to explore $Q^{\mathcal{R}*}$. Different strategies can be applied to deal with the situation where multiple re-writings of a given query are possible. Example is a Divide and Conquer strategy: The best results of each possible combinations of re-writings is returned. In the current version of the system we have implemented a simple version with similarities to skylining [KK02, LL87] which is well-known in database query relaxation.

In particular, we interpret the problem of finding relaxed queries as a classical search problem with small adaptations. The search space is defined by the set $Q^{\mathcal{R}*}$ of all possible queries. Each application of a rewriting rule R on a query Q is a possible action denoted as $Q \xrightarrow{R} Q^R$. A query represents a goal state in the search space if it does have answers. In the current implementation we use breadth-first search for exploring this search space. Different from classical search, however, the method does not stop when a goal state is reached; each goal state has to be determined because each goal state represent one sequence of successful rewriting when answers are provided. However a goal state need not to be explored further (i.e. no further re-writings must be applied to a goal state) but each search branch has to be closed by a goal state (or by a predefined depth). Breadth-first search ensures that each goal state represents the best solution to the relaxation problem with respect to a certain combination of re-writings. The goal states form a "skyline" for the rewriting problem and each of them is returned to the user together with the query answers.

The second difference to classical search is that we do not allow the same rule to be applied more than once with the same parameters in each branch of the search tree. The only kind of rules that can in principle be applied twice are rules that add something to the query (Rules that delete or replace parts of the query disable themselves by removing parts of the query they need to match against). Applying the same rule that extend the query twice leads to an unwanted duplication of conditions in the query that do not change the query result, but only increase the complexity of query answering.

5 Conclusions and Future Work

We have shown the use of semantic techniques in a prototype job portal in which both job offers and applicants are described according to ontologies. While preparing the ontologies from non-ontological sources (classification schemes) was time consuming and tedious, by doing so we are enabled to use ontology-based querying approaches to match relevant job applicants to vacancies and vice versa, as well as rank them in terms of similarity. Furthermore, we have shown that semantic matching alone does not allow for levels of similarity to be differentiated and for inconsistent or overly specific queries to be resolved. Hence we introduce another technique called query relaxation, in which queries can be rewritten to allow similarity to be ranked in different directions (e.g. in terms of subjects the applicant has experience in, or the number of years total experience he or she

has) or to widen the scope of the query in order to find matches (e.g. by searching on a superclass of the class searched for, or by making cardinalities less restrictive).

At present, the semantic job portal demonstrates improved precision and recall in the semantic matching, finding relevant job offers or applicants which would not be selected by a syntactic (text-based) query algorithm. However we have noted that alone this does not resolve more complex queries which can be expected in the real world. In the *European Network of Excellence Knowledge Web* the co-operation of leading Semantic Web researchers with selected industry partners with a real world business problem to solve is being supported. One of the first steps taken in the network was to collect industrial use cases where semantic technologies could form a potential solution, as well as derive from those use cases industry requirements for Semantic Web research [LPNS05]. The recruitment scenario was one of the industrial use cases provided and was identified by Semantic Web researchers as an ideal real world use case to test their results in query relaxation techniques. Within the Network the job portal is now being extended to support the rule rewriting approach described in this paper.

Our intention is to test the extended prototype against the original prototype (which supports only the semantic matching) using a set of benchmark queries. The HR ontology has been extended for this purpose with the property of experience and instances using this new property added to the semantic data used by the job portal. The rule rewriting tool has been implemented and an interface to the tool which is more general than the DIG interface used by most Semantic Web reasoners (in order to not limit ourselves to Description Logics) has been specified. We have also defined the first concrete technical details of rule rewriting. We plan to carry out the first tests at the beginning of 2007. This will provide us with a valuable real world test case to analyse the value of query relaxation techniques as an extension to semantic matching in ontology-based systems, in order to solve real world problems of inconsistent or overly specific queries. Performance is also be measured as another advantage of query relaxation is expected to be allowing more robust and efficient querying upon large knowledge bases which can scale to real world enterprise size.

Further research issues include determining a general guideline for the specification of rewriting rules and a generic framework for working with those rules. In combination, we believe this work not only demonstrates the use and value of Semantic Web techniques in a real world industrial test case, it indicates that any assessment of the cost of shifting to an ontology-based approach must also take into account the value to be gained from the availability of semantic techniques that is made possible when system data is based on formal logic through an ontology. In this paper we have introduced two such techniques: semantic matching and query relaxation, which are shown to be of value to the online recruitment process.

Acknowledgement The work described in this paper is supported by the EU Network of Excellence KnowledgeWeb (FP6-507482) and Knowledge Nets Project which is a part of the InterVal-Berlin Research Centre for the Internet Economy funded by the German Ministry of Research BMBF.

References

[BBD+04] Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H., und Poole, D.: Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of AI Research*. 21:135–191. 2004.

[BHM+05] Bizer, C., Heese, R., Mochol, M., Oldakowski, R., Tolksdorf, R., und Eckstein, R.: The Impact of Semantic Web Technologies on Job Recruitment Processes. In: *International Conference Wirtschaftsinformatik (WI'05)*. 2005.

[BMW04] Bizer, C., Mochol, M., und Westphal, D. Recruitment, report. April 2004.

[BN98] Baader, F. und Nipkow, T.: *Term rewriting and all that*. Cambridge University Press. New York, NY, USA. 1998.

[DSW06] Dolog, P., Stuckenschmidt, H., und Wache, H.: Robust query processing for personalized information access on the semantic web. In: *7th International Conference on Flexible Query Answering Systems (FQAS 2006)*. Number 4027 in LNCS/LNAI. Milan, Italy. June 2006. Springer.

[Ha04] Hayes, P.: RDF Semantics. Recommendation. W3C. 2004.

[Ke05] Keim, T. e. a. Recruiting Trends 2005. Working Paper No. 2005-22. efinance Institut. Johann-Wolfgang-Goethe-Universität Frankfurt am Main. 2005.

[KK02] Kießling, W. und Köstler, G.: Preference sql - design, implementation, experiences. In: *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*. S. 990–1001. Morgan Kaufmann. 2002.

[LL87] Lacroix, M. und Lavency, P.: Preferences; putting more knowledge into queries. In: Stocker, P. M., Kent, W., und Hammersley, P. (Hrsg.), *VLDB'87, Proceedings of 13th International Conference on Very Large Data Bases, September 1-4, 1987, Brighton, England*. S. 217–225. Morgan Kaufmann. 1987.

[LPNS05] Leger, A., Paulus, F., Nixon, L., und Shvaiko, P.: Towards a successful transfer of knowledge-based technology to European Industry. In: *Proceedings of the 1st Workshop on Formal Ontologies Meet Industry (FOMI 2005)*. 2005.

[Mü00] Mülder, W.: Personalinformationssysteme - Entwicklungsstand, Funktionalität und Trends. *Wirtschaftsinformatik. Special Issue IT Personal*. 42:98–106. 2000.

[Mo03] Monster: Monster Deutschland and TMP Worldwide: Recruiting Trends 2004. In: *2. Fachsymposium für Personalverantwortliche*. Institut für Wirtschaftsinformatik der Johann Wolfgang Goethe-Universität Frankfurt am Main. 2003.

[MPB06] Mochol, M. und Paslaru Bontas, E.: Practical Guidelines for Building Semantic eRecruitment Applications. In: *International Conference on Knowledge Management, Special Track: Advanced Semantic Technologies (AST' 06)*. 2006.

[OB05] Oldakowski, R. und Bizer, C.: SemMF: A Framework for Calculating Semantic Similarity of Objects Represented as RDF Graphs. In: *Poster at the 4th International Semantic Web Conference (ISWC 2005)*. 2005.

[PBM05] Paslaru Bontas, E. und Mochol, M.: Towards a reuse-oriented methodology for ontology engineering. In: *Proc. of 7th International Conference on Terminology and Knowledge Engineering (TKE 2005)*. 2005.

[PC95] Poole, J. und Campbell, J.: A Novel Algorithm for Matching Conceptual and Related Graphs. *Conceptual Structures: Applications, Implementation and Theory*. 954:293–307. 1995.

[SBA03] Sowa, F., Bremen, A., und Apke, S. Entwicklung der Kompetenz-Ontologie für die Deutsche Montan Technologie GmbH. `http://www.kowien.uni-essen.de/workshop/DMT\_01102003.pdf`. 2003.