Semantic Matchmaking using Ranked Instance Retrieval

Matthias Beck and Burkhard Freitag

Universität Passau, Fakultät für Mathematik und Informatik,
{Matthias.Beck, Burkhard.Freitag}@uni-passau.de

Abstract. Finding matching answers to a given request is hard, in particular if the objective is to find the most suitable partner. This especially holds in the context of the Semantic Web and its underlying specification language, i. e., description logics (DL)[1]. We present a clean, formal approach allowing users to devise preferences for queries on description logics knowledge bases. Thereby it is possible to use established DL-reasoners, particularly those tailored at instance retrieval like KAON2. Furthermore, our approach allows the formulation of soft constraints which is considered an important feature of semantic matchmaking.

1 Introduction

It is no news that matchmaking plays a crucial role in many areas of information systems. In the context of the Semantic Web, for example, matchmaking plays a central role in the discovery of appropriate semantic web services (cf. [1]) where appropriate means best complying with the user's request. So it is natural to provide the user with means to express his or her criteria for determining quality.

The contribution of this paper is a uniform method for the annotation of description logics instance retrieval queries with user preferences thereby allowing soft constraints resp. mere sorting. Since only the query is annotated and the knowledge base remains unchanged, it is possible to use standard reasoners like KAON2 [2], RACER [3] or, more generally, all reasoners supporting the so called DIG-Interface [4].

2 Motivating Example

Let us have a look at an example. Assume, you want to buy the latest "Harry Potter" DVD. You would probably look only for those web services that allow to order DVDs. Assume further that there is a registry which stores the service profiles of the particular web services in an appropriate formalism like OWL-S [5], which allows for more sophisticated semantic service descriptions as compared to standard formalisms like UDDI (cf. [6]). This easy query can be formulated as

[1] Other specification languages like the object-oriented Flora-2 are possible for the Semantic Web. Nonetheless, OWL is the current recommendation of the W3C.

$Q_1 := \mathit{OffersDVD}$. Retrieving results for Q_1 might be a quite tiresome experience since you would probably get tons of hits.

Of course you would provide more search criteria to narrow the search. For instance, if the DVD is meant to be a present and you are a little late, you would prefer shipping within 24 hours. But in any case shipping time should not exceed three days. This yields a new query

$$Q_2 := \mathit{OffersDVD} \sqcap (\mathit{24HoursShipping} \sqcup \mathit{3DaysShipping}).$$

That produces the correct result set[2], but without specifying user preferences, you get an unordered result that does not fit your needs. So the next step is to annotate the query with (numerical) user preferences. This results in the query

$$Q_3 := \mathit{OffersDVD}^1 \sqcap (\mathit{24HoursShipping}^2 \sqcup \mathit{3DaysShipping}^1).$$

The intuitive meaning is the following: in the result set individuals (e. g., web services) that provide 24 hours shipping[3] will get a higher rank than those offering three days shipping only. Note that membership in the result set is not affected by specifying preferences.

3 Preference Annotations

3.1 Ranking Tree

Due to the structure of queries and hence preferences a single numerical value is not sufficient to express rankings. The main reason is the existence of disjunctive knowledge in description logics. Consider the query $Q_4 := A^1 \sqcup (B^1 \sqcup C^2)^0$. Intuitively, since $B \sqcup C$ has a preference of 0, this expression should not contribute to the top level rank. On the other hand, if we get an equal top level rank for two individuals, an evaluation of $B^1 \sqcup C^2$ can be used to refine the ranking.

Definition 1 (Ranking Tree)

1. *For $r \in [0;1]$ (r) is a ranking tree.*
2. *Let $r \in [0;1]$ and $t_1, \dots, t_n$ ranking trees with $n \geq 1$, then $(r, t_1, \dots, t_n)$ is a ranking tree.*

$t_1 := (1, (1), (\frac{1}{3}, (1), (0)))$ is a sample ranking tree for query Q_4 (cf. section 3.2). Based on this definition we construct an ordering on ranking trees.

Definition 2 (Ordering $\trianglelefteq$ on ranking trees)
Let $a = (r_a, a_1, \dots, a_n)$ and $b = (r_b, b_1, \dots, b_n)$ with $n \in \mathbb{N}_0$ and $r_a, r_b \in \mathbb{R}$, a_i, b_i ranking trees with $i \leq n$. Then we define the relation $<$ by $a < b :\Leftrightarrow r_a < r_b$. Now we define that $a \trianglelefteq b$ holds if and only if

[2] The $\sqcup$-Operator is equivalent to a logical "or" while $\sqcap$ corresponds to a logical "and".
[3] In a more realsitic scenario you would possibly replace $\mathit{24HoursShipping}^2$ by $(\exists \mathit{shipsWithin}. \leq_{24})^2$ in query Q_3. The same holds for *3DaysShipping*. For the sake of brevity we stick to the shorter form.

$$
\begin{aligned}
r_{KB}(C_1^{r_1} \diamond \ldots \diamond C_n^{r_n}, o) &= (d, r_{KB}(C_1, o), \ldots, r_{KB}(C_n, o)) \\
\text{with} \quad d &= \begin{cases} 0, & \text{if } r_i = 0, \text{ for } i \in \{1, \ldots, n\} \\ \frac{\sum_{i=1}^{n} c_i r_i}{\sum_{i=1}^{n} r_i}, & \text{else} \end{cases} \\
\text{and} \quad c_i &= \begin{cases} 1, \text{ if } KB \models C_i(o) \\ 0, \text{ else} \end{cases} , \diamond \in \{\sqcup, \sqcap\} \\
r^{\neg}_{KB}(C_1^{r_1} \diamond \ldots \diamond C_n^{r_n}, o) &= (d^{\neg}, r^{\neg}_{KB}(C_1, o), \ldots, r^{\neg}_{KB}(C_n, o)) \\
\text{with} \quad d^{\neg} &= \begin{cases} 0, & \text{if } r_i = 0, \text{ for } i \in \{1, \ldots, n\} \\ \frac{\sum_{i=1}^{n} c_i^{\neg} r_i}{\sum_{i=1}^{n} r_i}, & \text{else} \end{cases} \\
\text{and} \quad c_i^{\neg} &= \begin{cases} 1, \text{ if } KB \models \neg C_i(o) \\ 0, \text{ else} \end{cases} , \diamond \in \{\sqcup, \sqcap\} \\
r_{KB}(\neg C, o) &= r^{\neg}_{KB}(C, o) \\
r^{\neg}_{KB}(\neg C, o) &= r_{KB}(C, o) \\
r_{KB}(\mathcal{X} R.C, o) &= -1, \mathcal{X} \in \{\exists, \forall, \leq n, \geq n\}, n \in \mathbb{N} \\
r^{\neg}_{KB}(\mathcal{X} R.C, o) &= -1, \mathcal{X} \in \{\exists, \forall, \leq n, \geq n\}, n \in \mathbb{N} \\
r_{KB}(A, o) &= \begin{cases} (1), & \text{if } KB \models A(o) \\ (0), & \text{else} \end{cases} \\
r^{\neg}_{KB}(A, o) &= \begin{cases} (1), & \text{if } KB \models \neg A(o) \\ (0), & \text{else} \end{cases}
\end{aligned}
$$

KB: knowledge base; $C, C_1 \ldots, C_n$: arbitrary concept expressions
A: atomic concept; R: role symbol; o: individual

Fig. 1. Ranking mapping

1. *$a < b$ or*
2. *$r_a = r_b$ and $\exists i : a_i < b_i$ and $\forall 1 \leq j \leq n : b_j \not< a_j$ or*
3. *$r_a = r_b \wedge \forall 1 \leq i \leq n : a_i \trianglelefteq b_i$*

Giving this definition a closer look, we can see that $a \trianglelefteq b$ if the top-level rank of a is smaller. If the top-level ranks are equal, there are two possibilities: one child top-level rank is smaller (and other child ranks are not greater) or all child trees are smaller w. r. t. $\trianglelefteq$. For example, if we have a second ranking tree $t_2 := (1, (1), (0, (0), (0)))$, then $t_2 \trianglelefteq t_1$ holds. Note that $\trianglelefteq$ is a partial order relation on the set of ranking trees (having the same structure).

3.2 Ranking Mapping

Given an annotated query and an individual, we must evaluate the rank (ranking tree) of that individual with respect to the query. This is done by the mapping shown in figure 1. Note that the result of the ranking mapping is a ranking tree. It is only meaningful to compare ($\trianglelefteq$) ranking trees if they were assembled using the same query. We do not further elaborate on the definition since it is rather technical. For background information on description logics we refer to [7]. Nonetheless we present some remarks on the mapping.

Central part of the definition is the mapping for (n-ary) conjunctions and disjunctions. The definition of the ranking function is the same for both. The ranking tree is created by computing a top-level rank d and afterwards computing ranking trees for the operands. The top-level rank is the weighted average[4] of the preferences. Negation is treated by "saving" it inside the $r_{KB}^{\neg}$-function. The negation is then re-applied while computing the $c_i^{\neg}$-parameter. This scheme is passed on to the operands.

Preferences inside role expressions are cut off by setting the rank of the expression to -1. This is justified as follows: consider the query $\exists \mathit{hasBeach}(\mathit{Rocky}^1 \sqcup \mathit{Sandy}^2)$. We must build a ranking for locations that have beaches by evaluation of certain characteristics of these beaches, in this case *Rocky* and *Sandy*. If more than one beach is associated to a single location, there are several possible semantics. A possibility is trying to maximize the ranking. In this example that would mean trying to find an associated beach that is rocky and sandy. It is also possible to evaluate the average ranking over all associated beaches. There are other reasonable semantics, too.

We are currently examining preferences inside role expressions and plan for an annotation method allowing the user to specify the particular aggregation function to be used. For the time being, preferences inside role expressions are not considered to avoid ambiguity. The structure of the resulting ranking tree very much resembles the query structure. The ranking tree is more compact because negation nodes are left out and nodes resulting from role expressions are cut off.

Recall query $Q_4 := A^1 \sqcup (B^1 \sqcup C^2)^0$ and ranking tree $t_1 := (1, (1), (\frac{1}{3}, (1), (0)))$ from section 3.1. If there is an individual o that is an instance of A and B but not C, then t_1 is the ranking tree for o w. r. t. Q_4.

3.3 Soft Constraints and Sorting

Definitions of the ranking tree and the ordering $\trianglelefteq$ allow for formulation of soft constraints using our formalism. So another way of refining query Q_1 from section 2 is to specify some soft constraints that you prefer being satisfied without enforcing them. For example you could prefer paying by credit card. Specifying this is now easy using preferences: $Q_5 := \mathit{OffersDVD}^1 \sqcap (\mathit{CreditCardPayment}^1 \sqcup \top^0)$. Here $\top$ denotes the top concept, i. e., every individual is an instance of $\top$. The result set consists of all individuals being instances of *OffersDVD* but those also providing credit card payment get a higher rank. Since we allow for n-ary operators, it is also possible to combine soft constraints: $Q_6 := \mathit{OffersDVD}^1 \sqcap$

[4] Aside from the weighted average, the rank could be computed by an arbitrary function.

```
<owl:Class rdf:about="http://www.im.uni-passau.de/pref#Query">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="http://www.im.uni-passau.de/matchmaking#UPS">
      <pref:Pref>1</pref:Pref>
    </owl:Class>
    <owl:Class rdf:about="http://www.im.uni-passau.de/matchmaking#FedEx">
      <pref:Pref>2</pref:Pref>
    </owl:Class>
  </owl:unionOf>
</owl:Class>
```

Fig. 2. Preference annotated query in OWL ($\mathit{UPS}^1 \sqcup \mathit{FedEx}^2$)

($\mathit{CreditCardPayment}^1 \sqcup \mathit{DeliveryAsGift}^2 \sqcup \top^0$). Moreover nesting is possible[5]:

$$Q_7 := \mathit{OffersDVD}^1 \sqcap (\mathit{CreditCardPayment}^2 \sqcup (\mathit{FedEx}^1 \sqcup \mathit{UPS}^2)^1 \sqcup \top^0)$$

It is worth noting here that it is possible to rank uncertain (disjunctive) knowledge. If the knowledge base entails the fact that an individual o is an instance of $\mathit{FedEx} \sqcup \mathit{UPS}$, this knowledge is ranked using preference 1 in Q_7. Since description logics allow for disjunctive knowledge, this is possible even without explicitly knowing whether o is an instance of *FedEx* or *UPS*. On the other hand, if we knew that o is also an instance of *UPS* for example, this knowledge would increase the rank on a lower ranking level. In addition to soft constraints it is even possible to formulate mere sorting conditions:

$$Q_8 := \mathit{OffersDVD}^1 \sqcup (\mathit{CCardPaym}^2 \sqcap \mathit{24HoursDelivery}^1 \sqcap \mathit{FedEx}^3 \sqcap \bot^0)$$

Since no individual is an instance of $\bot$, the second part of the formula is not satisfiable. So all results to query Q_8 must be instances of *OffersDVD* and therefore *OffersDVD* cannot affect the ranking. Hence, the second part is alone responsible for the ranking. This is due to the fact that an individual i can never be instance of $\mathit{CCardPaym} \sqcap \mathit{24HoursDelivery} \sqcap \mathit{FedEx} \sqcap \bot$. However, i can be an instance of *CCardPaym*, *24HoursDelivery* or *FedEx*. This information is used to build the ranking.

4 Prototype

Our ranking algorithm is composed of three parts that essentially represent three consecutive phases. In phase one the preference annotated query given in OWL is (pre)processed, preferences are extracted and removed from the query. For an example query see fig. 2. Results to the unannotated query are retrieved via the reasoner in phase two. At the moment we build upon the KAON2 reasoner [2][8]

[5] Note that *FedEx* (the same holds for *UPS*) is short for *offersShippingByFedEx* or $\exists \mathit{offersShippingBy}.\mathit{FedEx}$ in Query Q_7.

since it is tailored to instance retrieval. We plan to support other reasoners like RACER [3] and other DIG [4] compatible reasoners in the future.

Phase three performs the actual ranking. This is done by computing the ranking tree for each result retrieved in phase two. The results of several test runs are encouraging. We are using an indexed cache to speed up the computation. In our test runs we observed that the time for computation of the ranked result set t_{rank} depends linearly on the time for un-ranked retrieval t_{ret}, i. e., $t_{rank} \approx n \cdot t_{ret}$. The factor n is determined by the complexity of the query.

5 Discussion and Summary

Many approaches to ranking and preferences have been proposed. Castillo et al. [9] studied matchmaking of services by means of description logics. They allow for neither ranking nor user preferences. A different approach is due to Colucci et al. [10]. They allow for both negotiable and strict requirements in a description and introduce two novel description logics inference services, concept abduction and concept contraction, to deal with them.

Kießling et al. [11] developed PreferenceSQL, a powerful SQL extension. The objective is similar to our approach. While PreferenceSQL is aimed at relational data, our target are description logics knowledge bases. So in PreferenceSQL preferences are set on the attribute level while we set preferences on class membership. Another difference is that we can handle disjunctive knowledge which is common in description logics but not in databases. Besides, we allow for preferences on arbitrary constraints while this is only allowed for soft constraints in PreferenceSQL.

Brewka [12] describes a rank based description language with qualitative preferences. He follows a slightly different approach: central component is the ranking of different *models* of the knowledge base, not a ranking of consequences with respect to a query as in our approach. To accomplish that, he deeply integrates preferences into the logical language. Consequently, standard reasoners cannot be used. Another difference lies in the fact that his approach admittedly does not work well with numerical (quantitative) preferences.

A Google-like approach to knowledge ranking is presented by Ding et al. in [13]. They do not allow for user-defined preferences.

A recent paper [14] on the combination of ontologies and preferences is due to Lukasiewicz et al. Their notion of preference differs from ours. Particularly, they allow for qualitative preferences only.

As a natural limitation of our approach, annotations of user preferences must follow the query (term) structure. We argue that this is not a serious restriction since query and preference structure often match. Otherwise it is possible to specify a sorting (cf. section 3.3).

We have introduced an approach to semantic matchmaking by ranked instance retrieval in knowledge bases represented in description logics. Ranking is solely based on the query which is annotated with preferences. One of the advantages of our ranking approach lies in the fact that each user gets exactly the ranking he or she specified. Therefore, ranking is personalized and self-explanatory.

A prototype implementation has been presented that shows promising behavior. Of course several optimizations of our algorithm are possible and already on the way. For example, in certain cases a complete computation of the ranking tree is not necessary since the ranking can be evaluated directly for some concepts without looking at subconcepts. Also a more intensive use of index structures will be investigated.

6 Acknowledgement

We wish to thank the anonymous reviewers for valuable comments.

References

1. Burstein, M.H., Bussler, C., Zaremba, M., Finin, T.W., Huhns, M.N., Paolucci, M., Sheth, A.P., Williams, S.K.: A Semantic Web Services Architecture. IEEE Internet Computing **9**(5) (2005) 72–81
2. Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB), Universität Karlsruhe (TH): KAON2 - Homepage. http://kaon2.semanticweb.org (2006) last visited May 2006.
3. Haarslev, V., Möller, R.: Description of the RACER System and its Applications. In: Proceedings International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August. (2001) 131–141
4. Bechhofer, S., Möller, R., Crowther, P.: The DIG Description Logic Interface. In: Description Logics. (2003)
5. Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic Markup for Web Services. Website (2004)
6. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.P.: Importing the Semantic Web in UDDI. In: CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web, London, UK, Springer-Verlag (2002) 225–236
7. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press (2003)
8. Hustadt, U., Motik, B., Sattler, U.: Reducing SHIQ-Description Logic to Disjunctive Datalog Programs. In: KR. (2004) 152–162
9. González-Castillo, J., Trastour, D., Bartolini, C.: Description Logics for Matchmaking of Services. In Görz, G., Haarslev, V., Lutz, C., Möller, R., eds.: Proceedings of the KI-2001 Workshop on Applications of Description Logics. (2001)
10. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. Electronic Commerce Research and Applications **4**(4) (2005) 345–361
11. Kießling, W., Köstler, G.: Preference SQL - Design, Implementation, Experiences. In: VLDB. (2002) 990–1001

12. Brewka, G.: A Rank Based Description Language for Qualitative Preferences. In de Mántaras, R.L., Saitta, L., eds.: ECAI, IOS Press (2004) 303–307
13. Ding, L., Pan, R., Finin, T.W., Joshi, A., Peng, Y., Kolari, P.: Finding and Ranking Knowledge on the Semantic Web. In: International Semantic Web Conference. (2005) 156–170
14. Lukasiewicz, T., Schellhase, J.: Variable-Strength Conditional Preferences for Matchmaking in Description Logics. In: KR. (2006) 164–174