Exploiting Ontological Information for Reasoning with Preferences

Gil Chamiel and **Maurice Pagnucco**
School of Computer Science and Engineering, The University of New South Wales,
NSW, Sydney 2052, Australia and NICTA, Sydney, Australia.
Email: {gilc|morri}@cse.unsw.edu.au

Abstract

The ability to model preferences and exploit preferential information to assist users in searching for items has become an important issue in knowledge representation. Accurately eliciting preferences from the user in the form of a query can result in a coarse recommendation mechanism with numerous results returned. The problem lies in the user's knowledge concerning the items among which they are searching. Unless the user is a domain expert, their preferences are likely to be expressed in a vague manner and so vague results (in the form of numerous alternatives) are returned.

In this paper we remedy this problem by exploiting ontological information regarding the domain at hand. This ontological information can be provided by a domain expert and need not concern the user. However, we show that it can prove useful in focussing query results, providing more meaningful and useful recommendations.

We are faced with choices every day: which type of restauratnt to go to; which radio station to listen to or TV channel to watch? In order to answer these questions we exercise our preferences. Preferences make effective reasoning possible since they encapsulate our everyday decisions. However, explicit preferences alone are only part of the story. In some situations our own understanding of the domain is poor and our preferences tend to reflect this. As a result, we are unable to effectively discriminate among the choices at hand. This paper addresses this problem by supplementing user preferences with ontological information so as to provide an effective user preference mechanism.

Research on modelling abstract notions of preference has provided a rich literature in logic and decision theory (Doyle and Thomason 1999; Fishburn 1999; Bradley, Rafter, and Smyth 2000) with applications such as modelling social choice in economonics. With the growth of the World Wide Web as a major platform for purchase and consumption, the need for personalised product recommendation systems has become evident, and their development has become an important issue in Knowledge Representation and Reasoning and Artificial Intelligence. One method that has gained popularity in the last few years is *social-based product recommendation*, e.g. collaborative filtering (Sarwar et al. 2001), where the selection history of other customers is used to recommend products. Such techniques tend to ignore deeper information of the domain in favour of following the herd mentality.

The aim of this paper is to exploit information that is available in the structure of an OWL ontology to supplement user preferences in order to provide an effective choice mechanism. We consider a number of methods based on different definitions of concept similarity *as measured in a RDF graph representation of an OWL ontology. Furthermore, we provide an implementation of our schemes in the SPARQL query language that extends previous work by (Siberski, Pan, and Thaden 2006) building on the preference mechanisms added to SQL by (Kießling 2002; Kießling and Köstler 2002).*

Previous work in this area is limited. An ontology based similarity system has been presented by (Middleton, Shadbolt, and Roure 2004) but provides for only basic features. (Schickel-Zuber and Faltings 2006) is much closer in spirit to this paper, introducing the notion of *ontology filtering*. However, they propose a score propogation system within an hierarchical graph, where we focus on the structural properties of the ontology. (Kiefer, Bernstein, and Stocker 2007) has applied similarity to semantic data mapping, ontology mapping and semantic web service matchmaking using techniques from linguistic analysis. Semantic similarity for linguistic analysis has been studied previously (e.g. (Li, Bandar, and McLean 2003)) and there are lessons here for conceptual similarity in ontologies.

The rest of this paper is arranged in the following way. In the next section we cover the necessary background material with a brief introduction to OWL and SPARQL. We then briefly look at the work by (Kießling 2002; Kießling and Köstler 2002) extending SQL and it's partial implementation in SPARQL by (Siberski, Pan, and Thaden 2006). This is followed by our proposal based on concept similarity, exploiting the structure in a RDF based OWL ontology. We end with a discussion and conclusions.

Background

In our work, we adopt principles from preferences in database systems and description logic based ontologies. In this section, we briefly discuss these concepts.

Preference Querying in Database Systems

In the context of database systems, (Kießling 2002) presents a framework for dealing with preferences as soft constraints, named *Preference-SQL*. In this work, preferences are seen as strict partial orders over database tuples i.e. as transitive and irreflexive preference relations. It proposes an extension to the SQL query language (Kießling and Köstler 2002) which permits filtering the result set using soft constraints as opposed to classical SQL filtering which adopts hard constraints returning only those results that match the query *exactly*. A partial syntax of the extended query language is given below:

```
SELECT      <projection-list>
FROM        <table-reference>
WHERE       <hard-conditions>
PREFERRING  <soft-conditions>
ORDER BY    <attribute-list>
```

Using this syntax, the user can express their preferences as soft constraints and will receive tuples which *best match* those constraints. This approach is referred to as the BMO (*Best Match Only*) query model in which a tuple will find its way into the final result set if there does not exist any other tuple which *dominates* it, i.e. better satisfies the preference constraints. Preference constraints in this framework can be expressed through standard SQL operators in terms of likes/dislikes (e.g. `=`, `<>`, IN) and numeric constraints (e.g. `<`, `>=`, `BETWEEN`). This work also introduces a set of special operators which allow the user to express their preferences in terms of numerical approximations (through the operator `AROUND` which will favour values close to a given numerical target value) and in terms of minimization/maximization of numerical values (through the operators `LOWEST`/`HIGHEST` which will accept a lowest/highest value respectively over other values). In order to allow complex preference construction, two binary preference assembly operators are introduced:

The *Pareto accumulation* (`AND`) treats both constituent preference constructs as equal and is defined as:

$$x \prec_{P_1 \otimes P_2} y \Leftrightarrow (x \prec_{P_1} y \wedge (x \prec_{P_2} y \vee x \equiv y)) \vee (x \prec_{P_2} y \wedge (x \prec_{P_1} y \vee x \equiv y))$$

Where x, y are database tuples and P_1, P_2 are preference constructs. Intuitively, this definition says that y is strictly preferred to x whenever it is strictly preferred by at least one of the two constituent preferences and equally or more preferred by the other.

The *Prioritize accumulation* (`CASCADE`) is used to treat two preference constructs in order of importance and is defined:

$$x \prec_{P_1 \&\& P_2} y \Leftrightarrow x \prec_{P_1} y \vee (x \prec_{P_2} y \wedge (x \prec_{P_1} y \vee x \equiv y))$$

This definition says that x is strictly preferred to y whenever it is strictly preferred by the first preference in the cascade and otherwise it is strictly preferred by the second preference in the cascade and equally or more preferred by the first. Therefore, the first constituent preference in the cascade is given higher consideration.

Querying Ontological Information

Ontologies provide a standardised way of classifying terms related to a domain. They are central to the knowledge-based paradigm.

Description Logic Based Ontologies In recent years, with the emergence of the Semantic Web, the importance of knowledge representation and reasoning techniques over ontological information has gained added significance. Ontologies (Antoniou and van Harmelen 2004) are a knowledge representation technique based on description logic (*DL*) principles consisting of terminological entities, i.e. *Concepts* and *Properties* also referred to as $TBox$ and *instances* (individuals) also refered to as $ABox$. A very important feature of ontologies is the inherent ability to define terminological objects in a hierarchical manner, that is, concepts and sub-concepts, properties and sub-properties.

RDF and OWL A very common methodology for creating ontological information is through the XML-based markup langague OWL (*Web Ontology Language*) which allows the construction of the different entities of an ontology based on RDF (*Resource Description Framework*) graphs. In RDF, entities are represented using a triple pattern logic. Each element in the RDF graph has to be a literal or an RDF resource. For example, the following tag constructs an OWL class (concept) named Car: `<owl:Class rdf:ID="Car">`. Here, the subject is a class definition, the predicate is the class id and the object is the string "Car". Properties and Individuals can be created in the same manner. For example, the following tag constructs a new individual Car `<Car rdf:ID="Car_1">`. Suppose we have a property definition of a car's year, the following tag will assign the number 2005 as `Car_1`'s year: `<Car_1 :hasYear=2005>`.

SPARQL The SPARQL query language over RDF graphs (Prud'hommeaux and Seaborne 2006) is a W3C Recommendation and considered to be a vital tool for dealing with ontological information in the semantic web. SPARQL allows queries over RDF Graphs using *Triple Pattern Matching* by introducing variables and binding the appropriate RDF resources to the variables. A partial syntax for SPARQL is given below:

```
SELECT    <projection-var-list>
FROM      <ontology-reference>
WHERE     <var-bindings>
FILTER    <hard-conditions>
ORDER BY  <var-list>
```

As opposed to SQL, the projected entities are variables where the equivalent entity to a database tuple is a set of variables also referred to as a *solution binding* (or a *result binding*). The `WHERE` clause in the query is typically used to bind variables to RDF resources while hard constraints are given through the `FILTER` clause (although it is possible to perform certain filter operations through the `WHERE` clause itself). The `ORDER BY` clause acts as a solution modifier, i.e. a solution list transformation function, which sorts the result bindings according to the variables in a given list.

Other solution modifiers offered by SPARQL are `LIMIT` to limit the number of results returned and `OFFSET` to instruct SPARQL to start the result set after a given number of bindings are found.

Querying Ontological Information with Preferences

A natural progression is to extend SPARQL with preferential queries.

P-SPARQL

In a similar fashion to the way that Kießling extended SQL to enable database querying with preferences, (Siberski, Pan, and Thaden 2006) presents an extension to SPARQL to query ontological information with preferences. The fundamental idea here is similar; a new query element is introduced to allow the construction of preferences as soft constraints. The extended syntax of SPARQL (in the rest of this paper, we refer to this extension as P-SPARQL) is given below:

```
SELECT      <projection-var-list>
FROM        <ontology-reference>
WHERE       <var-bindings>
FILTER      <hard-conditions>
ORDER BY    <var-list>
PREFERRING  <soft-conditions>
```

In the preferring section, every filter operator supported by SPARQL can be used as well as two scoring operators, `HIGHEST/LOWEST` with similar semantics as Preference-SQL. Also, similarly to Preference-SQL, two complex preference assembly methods are implemented, i.e. the Pareto operator for treating two preference operators as equally important and the Cascade operator to prioritize one preference operator over the other. Finally, in this work the BMO (Best Match Only) query model was adopted where a solution binding is a best match if there is no other solution binding dominating it (i.e., strictly preferred). Each solution binding competes against every other solution binding where a solution binding will find its way into the final result set if it is a best match under this definition.

Exploiting Hierarchical Structure for Reasoning with Preferences

One of the most distinctive properties in representing knowledge using ontologies is the inherent ability to define the terminologies in the ontology in an hierarchical manner. In analogue to terminologies in database systems, i.e. the database schema, in this work we consider the terminoligical component of an ontology ($TBox$) to be the part which stores information created by experts on the domain at hand. This assumption will then enable us to supply (possibly recommend) to the user information about individuals ($ABox$) according to their preferences without having to assume a very high level of domain knowledge on behalf of the user, and exploit the level of knowledge the user does have in order to perform more accurate preference querying. This is important because in many product recommendation settings, the user possesses little if any domain knowledge.

Motivating Example

The Beer Ontology Consider an ontology which describes and stores information about beer (Figure 1). This may include the type of beer, country of origin, alcohol level etc. Drawn from domain-specific knowledge, beer types are typically viewed in an hierarchical manner. It is common to consider categorizations of beer types by the differences in yeast used in the fermentation process. Thus, the main beer categories are *lagers* (made of bottom-fermenting yeasts), *ales* (top-fermenting yeasts) and *lambics* (spontaneous fermentation using wild yeasts). Under these categories we can find sub-types of lagers, ales and lambics and so on. A partial beer type hierarchy graph is given in Figure 1. Note that this information was taken from domain specific expert data and it is not neccesary for users (beer consumers in this case) to be aware of the different types of beer or the hierarchical relations between them. Still this information can be readily exploited when reasoning with preferences. This domain example was chosen in the context of this work due to the fact that each hierarchical level defines not only a level of abstraction but can store real objects as well. For example, the type Dark Ale is an abstract concept (possibly defined as a dark top-fermented beer) which other beer types (e.g. Stout and Porter) inherit from but it is also a way of describing real objects (some beers are categorized simply as Dark Ales). In other words, individuals (beer instances) will be associated not only with leaf concepts but with concepts at any level. This is a crucial point for the work presented here.

Example Query Consider the following P-SPARQL query:

```
Prefix beer:
  <http://example.com/beer.owl#>
SELECT ?id ?type ?country
FROM <http://example.com/beer.owl>
WHERE {
  ?id rdf:type ?type.
  ?id beer:hasCountry ?country.}
PREFERRING
  ?country = beer:Australia
CASCADE
  ?type = beer:DarkAle
```

In this query, we prefer Australian beers as our most important preference attribute before we prefer dark ales. In case no such beer exists so it perfectly answers our preference (i.e. no Australian Dark Ale exists), we argue that it will be sensible to return an Australian beer with similar type to *Dark Ale* over returning any arbitrary beer type which happens to be Australian. Running this query through P-SPARQL will have exactly the behaviour where any type of beer will be considered equal without examining its relationship with the target concept.

Equality as an *IS-A* Relation

In Description Logic (Baader et al. 2003) concept inheritance can be viewed as an *IS-A* relation $C_1 \sqsubseteq C_0$ where a sub-concept (or property) C_1 is also of type C_0 by inheriting its properties and functionality. In our case, *Stout*

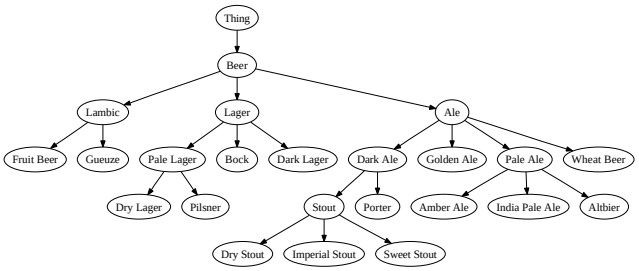

Figure 1: Ontological Concept Hierarchy of Beer Types

will be considered a Dark Ale and thus (assuming the existence of an Australian Stout) should dominate other types which are not a sub-concept of Dark Ale. Even though OWL is based on description logic concepts, this behaviour is not inherited in SPARQL. Filtering using the equal operator will return only those objects which have that particular target concept. Furthermore, the filtering option `?type rdfs:subClassOf C` (for querying individuals which have a sub-concept of some target concept C), will result only in those that are a direct sub-class of C omitting individuals which have the type C itself (and thus not satisfying the axiom $C \sqsubseteq C$) as well as individuals which have a type which is not directly inherited from C (and thus not satisfing the transitivity property of the *IS-A* relation). The effect of the *IS-A* relation can be obtained by changing the semantics of the equality operator so that $C_1 = C_0 \Leftrightarrow C_1 \sqsubseteq C_0$. Note that querying with the *IS-A* relation can affect not only preference querying but also standard hard-constraint querying. Note also that treating the equality operator as an *IS-A* relation will result in it being asymmetric.

However, there are still a few problems with this method: what if there is no result under the target concept (or any of its sub-concepts)? Should we consider all other concepts outside the target concept as equal? Furthermore, is there any relation between different levels within a sub-hierarchy? Do we want to distinguish between general and specific concepts? We discuss these issues in the next section.

Similarity-based Querying

In order to expolit the hierarchical structure of an ontology, we present here a series of methods for computing categorical similarity between concepts in a $TBox$. We use these similarity methods for performing preference querying over ontological information.

We introduce a new Boolean operator $Sim(C_1, C_2)$ (is similar to):

$$b_1 \prec_{P(C_0)} b_2 \Leftrightarrow Sim(C(b_1), C_0) < Sim(C(b_2), C_0) \quad (1)$$

Where $C_0 \in Concepts$ is the target concept, $b_1, b_2 \in ResultBindings$ and $C(b_i)$ is the value bound to the relevant variable in the result binding b_i w.r.t C.

For example, in order to change the previous example to prefer Australian beers and then beers **similar to** the type Dark Ale, we change the `PREFERRING` section of our query to:

```
PREFERRING
  ?country = beer:Australia
CASCADE
  ?type ~= beer:DarkAle
```

Where $\sim=$ is the syntactic version of the similarity operator $Sim(C_1, C_2)$.

Note that by introducing a new Boolean operator we do not change the notion of domination querying. We still have the ability to compare two result bindings to obtain the preference domination relationship between them.

There are many ways to compute similarity between concepts in an ontology, each reflecting a different rationale. In the rest of this section we discuss various methods and their rationales for computing this kind of similarity measurement.

Similarity via Direct Graph Distance A very simple method for measuring similarity in an ontology graph is by looking at the direct distance between a candidate concept and the target concept:

$$Sim(C_1, C_0) = \frac{1}{Dist(C_1, C_0)} \quad (2)$$

Where distance is defined as the distance of each concept to the *most recent common ancestor* of the two concepts:

$$Dist(C_1, C_0) = N_1 + N_0 + 1 \quad (3)$$

$N_i = len(C_i, MRCA(C_1, C_0))$ and $MRCA(C_1, C_0)$ is the most recent common ancestor of C_1 and C_0 (i.e., the *meet* in lattice theoretic terms). Note that the distance between a concept and itself here is equal to 1 to avoid division by zero. The rationale behind this simple method is to consider concepts which are "closer" to the target concept more similar. For example, a plain *Ale* or a *Stout* will be considered more similar to the target concept *Dark Ale* then a *Pale Lager*. The main problem with this approach is the fact that it is not consistent with the *IS-A* relation assumption. It may consider concepts which are not a sub-concept of the target concept more similar than concepts under the target concept. This naïve approach, being less intuitive, can however serve well as a benchmark method for comparing other methods.

Distance and the *IS-A* relation Combining the two ideas presented so far, i.e. treating a sub-concept as the target concept (the *IS-A* relation) and measuring similarity via graph distance solves the above problem. (4) is a similarity measurement that will differentiate between sub-concepts and non-sub-concepts of the target concept. Sub-concepts of the target concept will all be equally similar to the target concept and will always be more similar to the target concept than those concepts "outside" the target concept. The similarity of concepts outside the target concept will be according to their closeness to the target concept.

$$Sim(C_1, C_0) = \begin{cases} 1 & \text{if } C_1 \sqsubseteq C_0, \\ \frac{1}{Dist(C_1, C_0)} & \text{if } C_1 \not\sqsubseteq C_0. \end{cases} \quad (4)$$

Communicated level of specificity The previous method considers all concepts below a target concept as equal. This

may well be the case for many users or in many domains (we discuss the relevance of the different methods in the discussion section) but in some cases, the following argument can be made: suppose a user asks for an Ale. As discussed above, we cannot assume that the user is familiar with the complete sub-hierarchy under Ale. In other words, we cannot assume that the user knows exactly which types of Ale exist (we only assume that the ontology engineer and the people in charge of data entry are experts in the field and can be trusted). What we can assume is that our user knows what Ale means (as opposed to other types of beer). Furthermore, if a different user asks for a very specific type of Ale, we can now assume that he is familiar with that type and probably looking specifically for it. If the user asking for an Ale wanted that very specific type of Ale, he would probably ask for it directly. In other words, we may consider sub-concepts "closer" (and thus more general) to the target concept because they are closer to the **level of specificity** the user has **communicated in their preference**. The following measure will create this effect:

$$Sim(C_1, C_0) = \begin{cases} \frac{1}{Dist(C_1, C_0)} & \text{if } C_1 \sqsubseteq C_0, \\ -Dist(C_1, C_0) & \text{if } C_1 \not\sqsubseteq C_0. \end{cases} \quad (5)$$

Most Specific Shared Information So far we have concentrated on the similarity between sub-concepts under the target concept. We saw different ways of treating this information. Another very important issue when dealing with hierarchical-based similarity methods is the similarity between the different concepts not inherited from the target concept and that target concept. Consider the following query over our beer ontology: the user has asked for a Stout (possibly among other criteria). There is no Stout or a sub-concept of a Stout which satisfy the request. We do consider an Ale and a Porter. Up until now, these two concepts will be considered as equally similar to Stout (both with distance 3 from Stout). The problem here is that it may well be argued that Porter is more similar to Stout than a plain Ale since Stouts and Porters **share more specific information**. Both Stout and Porter inherit from Dark Ale which means that they share the special properties of a Dark Ale that distinguish it from a plain or any other type of Ale. (Wu and Palmer 1994) presented a similarity measurment (in the context of linguistics) which takes the level of shared information into account:

$$Sim(C_1, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3} \quad (6)$$

Where N_1, N_2 are the distances from the concepts C_0 and C_1 to their MRCA respectively and N_3 is the distance from this MRCA and the root of the ontology (assuming the most general concept is the OWL concept *Thing*). In our example, $Sim(Stout, Ale) = \frac{2*2}{2+0+2*2} = 0.67$ while $Sim(Stout, Porter) = \frac{2*3}{1+1+2*3} = 0.75$ and thus Porter will be considered more similar to Stout than Ale and will dominate it in the context of beer type.

A major drawback in using this method to compute similarities in ontology hierarchies is that it does not respect the *IS-A* relation axiom on ontologies with depth > 4 when comparing the similarity between a general concept and a sub-concept on depth ≥ 4 from it. For example, this method will give a similarity measurement of 0.5 between *Ale* and *Lager* and a smaller similarity measurement of 0.44 between *Ale* and a sub-concept of *Ale* at depth 6 (that is, a very specific type of Ale). It may be argued that even though this very specific type of Ale is indeed an Ale, due to its specific charateristics it may have gone far enough from the base concept to be considered less similar to it compared to a concept outside that base concept. We find no justification for making such an argument mainly due to the fundumental significance of the *IS-A* relation in description logic conceptual inheritance.

We propose here a method for measuring similarities between concepts while considering specific shared information more similar and preserving the *IS-A* DL axiom.

Most Specific Shared Information w.r.t Tree Depth In order to measure similarity between concepts while considering concepts that share more specific information to be more similar and preserve the DL semantics of the *IS-A* relation, we modify (6) by reducing the similarity measurement in relation to the depth of the compared concepts and the maximal depth of the tree. In order to keep the method symmetric, we look at the average distance of the two compared concepts to the depth of the tree:

$$Sim(C_1, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3 + AVG} \quad (7)$$

Where N_1, N_2 and N_3 are defined as in (6), AVG is the average distance of MAX to the depth of the concepts C_0 and C_1 and MAX is the length of the longest path from the root of the ontology to any of its leaf concepts. For example, $Sim(Stout, Ale) = \frac{2*2}{2+0+2*2+2} = 0.5$ while $Sim(Stout, Porter) = \frac{2*3}{1+1+2*3+1} = 0.67$ and thus, as for the previous method, Porter will be considered more similar to Stout than Ale. But now, as opposed to (6), this will still give a smaller similarity measurement (0.24) between *Ale* and *Lager* and a greater similarity measurement between *Ale* and any sub-concept of *Ale*, e.g. 0.4 for a sub-concept of *Ale* at depth 6. We can prove that (7) guarantees that sub-concepts are always preferred to non-sub-concepts w.r.t a target concept.

Theorm 1. *Let $\prec$ be defined by (1) with Sim defined as in (7). Let C_0, C_1, C_2 be concepts in an ontology such that $C_1 \sqsubseteq C_0$ and $C_2 \not\sqsubseteq C_0$. For all result bindings b_1, b_2 such that $C(b_1) \in C_1$ and $C(b_2) \in C_2$ it holds that $Sim(C_0, C_1) > Sim(C_0, C_2)$ (hence $b_2 \prec_{P(C_0)} b_1$).*

The powerfulness of this method is that it encapsulates both discussed intuitions while preserving the symmetric nature of (6). It stands in the *IS-A* relation axiom, always considering sub-concepts of a target concept more similar than non-sub concepts w.r.t a target concept. It also considers concepts that share more specific information more similar unless the previous statement does not hold.

Implementation

An implementation of the methods proposed here have been completed based on the *ARQ* SPARQL query engine (Seaborne 2005) (a *Jena* based query engine) and building on the implementation of (Siberski, Pan, and Thaden 2006) where the iterative processing of preference querying is performed as a *solution modifier* (similarly to the way the classical sorting functionality `ORDER BY` is done). On top of the described similarity measurement methods, we complemented this implementation by adding some further numerical preference attributes mentioned in (Kießling and Köstler 2002) such as `AROUND` and `BETWEEN`. In order to give the user the flexibility in choosing their preference query semantics w.r.t similarity, the system receives two parameters: equality operator semantics (strong equivalence or *IS-A*) and a similarity measurement method.

Discussion

We have described here various similarity meaurement techniques to be used in the context of preference querying. How should one choose a method in order to utilize this for their own purpose? Furthermore, who should bear the responsibility of deciding on the different parameters in this system? We briefly discuss these issues here.

In terms of the similarity method applied, apart from the different rationale encapsulated in the different methods, it is important to distinguish between two different inheritance systems: *Abstract Inheritance Systems* and *Data Inheritance Systems*. Abstract inheritance systems define an hierarchical system in which data objects (individuals) are attached only to leaf nodes. The intermediate levels in the hierarchy serve as abstract concepts (and thus are not used for instantiation). For example, suppose we have an hierarchy over the different programming languages according to some features of those languages. *Java* may inherit from the concept *Object Oriented Language* but this concept cannot be used for instantiation of a specific program (a program will eventually be written in a specific language). This abstract concept can be used for reasoning purposes. In data inheritance systems, non-leaf nodes also often define an abstraction over their child concepts but instances can be associated with any concept in the hierarchy. An example of a data inheritance system is our beer ontology. We argue that this distinction should be taken into consideration when selecting a similarity measurement method. For example, pure distance based methods (2) and (5) will not perform well on abstract inheritance systems while *IS-A* relation based methods will perform better on those systems.

Conclusions

In this paper we have enhanced preference querying by supplementing the user's preferences with ontological information; in particular, by exploiting the structural properties of the ontology describing the domain at hand. We claim that this is significant for a vast class of ontologies that we refer to as *data inheritance systems*. By introducing a notion of similarity based on distance metrics, we free the user from having to possess a deep understanding of the underlying domain. In particular, we adapt one similarity measure (6) and introduce another (7). Ontologies are developed by domain experts who have intimate knowledge of their subject area and are exploited by naive users who can specify weaker preferences without the fear of being overwhelmed by the results. Moreover, the results are not biased by the whims of previous user consumption but by exploiting domain expertise. Our proposals are implemented on ARQ and enhance the SPARQL standard.

References

Antoniou, G., and van Harmelen, F. 2004. *A Semantic Web Primer (Cooperative Information Systems)*. MIT Press.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The description logic handbook: Theory, implementation, and applications*. New York, NY, USA: Cambridge University Press.

Bradley, K.; Rafter, R.; and Smyth, B. 2000. Case-based user profiling for content personalisation. *Lecture Notes in Computer Science* 1892:62–72.

Doyle, J., and Thomason, R. H. 1999. Background to qualitative decision theory. *AI Magazine* 20(2):55–68.

Fishburn, P. 1999. Preference structures and their numerical representations. *Theor. Comput. Sci.* 217(2):359–383.

Kiefer, C.; Bernstein, A.; and Stocker, M. 2007. The fundamentals of iSPARQL – A virtual triple approach for similarity-based semantic web tasks. In *ISWC '07*.

Kießling, W., and Köstler, G. 2002. Preference SQL: design, implementation, experiences. In *VLDB*, 990–1001.

Kießling, W. 2002. Foundations of preferences in database systems. In *VLDB*, 311–322.

Li, Y.; Bandar, Z. A.; and McLean, D. 2003. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering* 15(4):871–882.

Middleton, S. E.; Shadbolt, N. R.; and Roure, D. C. D. 2004. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.* 22(1):54–88.

Prud'hommeaux, E., and Seaborne, A. 2006. SPARQL query language for RDF. Technical report, W3C Candidate Recommendation.

Sarwar, B. M.; Karypis, G.; Konstan, J. A.; and Reidl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, 285–295.

Schickel-Zuber, V., and Faltings, B. 2006. Inferring User's Preferences using Ontologies. In *AAAI 2006*, 1413–1418.

Seaborne, A. 2005. ARQ - A SPARQL processor for Jena. http://jena.sourceforge.net/arq.

Siberski, W.; Pan, J. Z.; and Thaden, U. 2006. Querying the semantic web with preferences. In *International Semantic Web Conference*, 612–624.

Wu, Z., and Palmer, M. 1994. Verb semantics and lexical selection. In *32nd Annual Meeting of the Association for Computational Linguistics*, 133 –138.