Utilising Ontological Structure for Reasoning with Preferences

Gil Chamiel **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. On the one hand, 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 focusing query results and providing more meaningful and useful recommendations.

Keywords: Ontologies, Reasoning with Preferences

1 Introduction

We are faced with choices every day: which type of restaurant 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 modeling abstract notions of preference has provided a rich literature in logic and decision theory (Doyle & Thomason 1999, Fishburn 1999, Bradley et al. 2000) with applications such as modeling social choice in economics. 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 et al. 2006) building on the preference mechanisms added to SQL by (Kießling 2002, Kießling & Köstler 2002).*

Previous work in this area is limited. An ontology based similarity system has been presented by (Middleton et al. 2004) but provides for only basic features. (Schickel-Zuber & Faltings 2006) is much closer in spirit to this paper, introducing the notion of *ontology filtering*. However, they propose a score prorogation system within an hierarchical graph, where we focus on the structural properties of the ontology. (Schickel-Zuber & Faltings 2007) supplement this approach by trying to learn the ontologies themselves. (Kiefer et al. 2007) has applied similarity to semantic data mapping, ontology mapping and semantic web service matchmaking using techniques from linguistic analysis.

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) and (Kießling & Köstler 2002) extending SQL and it's partial implementation in SPARQL by (Siberski et al. 2006). This is followed by our proposal based on concept similarity, exploiting the structure in an OWL ontology. We end with a discussion and conclusions.

2 Background

Reasoning with preferences has been studied and developed in many disciplines from social choice, decision and game theory, logic and philosophy through to artificial intelligence and machine learning. In our work, we adopt these basic principles together with their application to preference querying in database systems. In order to provide more accurate and sensible preference querying, we have chosen to use description logic based ontologies as our data model since they provide a richer description of the underlying domain. In this section, we briefly discuss these concepts.

2.1 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 & 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.

2.2 Querying Ontological Information

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

2.2.1 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 & van Harmelen 2004) are a knowledge representation technique based on description logic (*DL*) (Baader et al. 2003) principles consisting of terminological entities, i.e. *Concepts* and *Properties* also referred to as *TBox* and *instances* (individuals) also referred 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 subconcepts, properties and sub-properties. We make no assumption about the nature of the ontology. For simplicity we concentrate on tree-like ontologies in this paper. However, the results can be straightforwardly extended to DAGs.

2.2.2 RDF and OWL

A very common methodology for creating ontological information is through the XML-based markup language OWL (*Web Ontology Language*) which allows the construction of the different entities of an ontology based on RDF 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 MusicAlbum: `<owl:Class rdf:ID="MusicAlbum">`. Here, the subject is a class definition, the predicate is the class id and the object is the string "MusicAlbum". Properties and Individuals can be created in the same manner. For example, the following tag constructs a new individual music album `<MusicAlbum rdf:ID="Album_1">`. Suppose we have a property definition of a album's release year, the following tag will assign the number 2005 as `Album_1`'s release year: `<Album_1 :releaseYear=2005>`.

2.2.3 SPARQL

The SPARQL query language over RDF graphs (Prud'hommeaux & 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.

3 Querying Ontological Information with Preferences

A natural progression is to extend SPARQL with preferential queries.

3.1 P-SPARQL

In a similar fashion to the way that Kießling extended SQL to enable database querying with preferences, (Siberski et al. 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.

4 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 terminological 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.

4.1 Motivating Example

4.1.1 The Music Record Shop Ontology

Consider an ontology which describes and stores information about music albums. This may include musical genre, performing artists, country where the album was produced, price and so on. Let's also consider a concept hierarchy composed by musical experts given in Figure 1. Note that this information is taken from domain experts while we do not assume the user has complete knowledge of the domain. Still this information can be readily exploited when reasoning with preferences (e.g. for a consumer who would like to search for items in a more sophisticated manner). In (Chamiel & Pagnucco 2008*a*) we identify a class of hierarchical systems referred to as *data inheritance systems*. This means that each concept in the hierarchy (not only a leaf) can be referred to not only as an abstract concept (probably through a set of properties this concept contains) but also as a data concept—a concept which can be associated with asserted instances (i.e., real objects). This is the standard behaviour of ontologies in the semantic web. This musical genre hierarchical system is of this kind: an album can be identified as *Progressive Rock* which is a leaf concept but at the same time an album can be identified with the concept *Rock* even though it serves as an abstract concept.

Example Query Consider the following P-SPARQL query:

```
Prefix music:
  <http://example.com/music.owl#>
SELECT ?id ?genre ?country
FROM <http://example.com/music.owl>
WHERE {
  ?id music:hasGenre ?genre.
  ?id music:producedIn ?country.}
PREFERRING
  ?country = music:England
CASCADE
  ?genre = music:AlternativeRock
```

In this query, we prefer albums originating from England as our most important preference attribute before we prefer the alternative rock genre. In case no such album exists to perfectly answer our preference (i.e. no English Alternative Rock exists), we argue that it will be sensible to return an English album with genre similar to *Alternative Rock* over returning any arbitrary album which happens to be English. Running this query through P-SPARQL will have exactly this latter behaviour where any musical genre will be considered equal without examining its relationship to the target concept (English album). This will of course depend on what we mean by "similar to" which we explore next.

4.2 Querying over *IS-A* Relations

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 C_1 is also of type C_0 by inheriting its properties and functionality. 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 satisfying the transitivity property of the *IS-A* relation). In (Chamiel & Pagnucco 2008*a*) we introduce a new operator which allows the user to express preferences in terms of the *IS-A* relation.

Example 1. *In order to modify the previous example to prefer English albums and then albums of genre which* IS-A *Alternative Rock (i.e., Alternative Rock and all its descendants), we modify the* `PREFERRING` *section of our query to:*

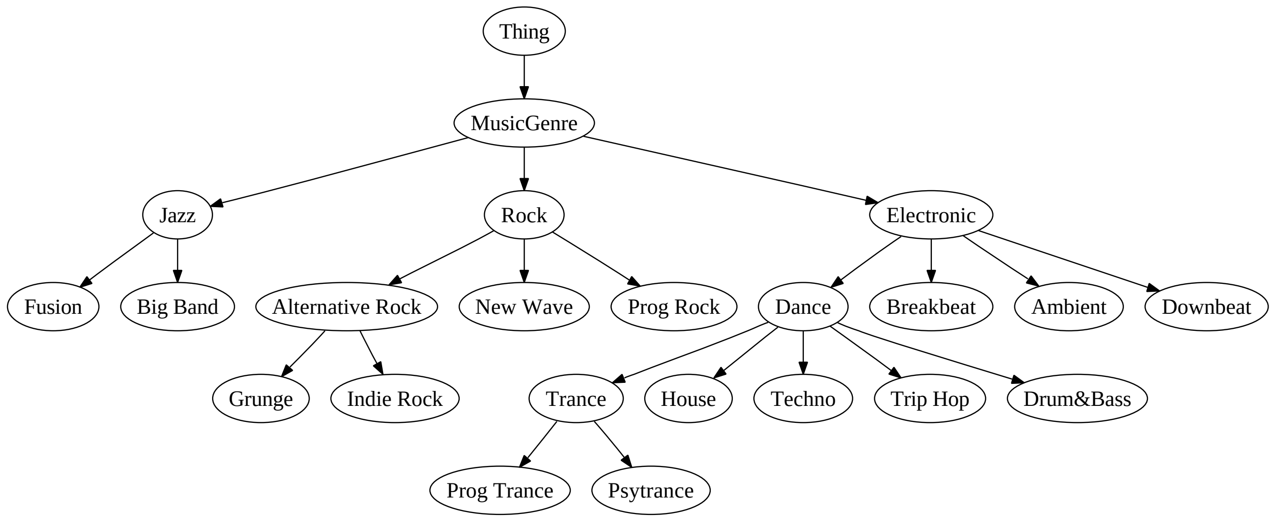


Figure 1: Ontological Concept Hierarchy of Music Genres

```
PREFERRING
  ?country = music:England
CASCADE
  ?genre ISA music:AlternativeRock
```

The results will thus be an albums made in England with genre in {AlternativeRock, Grunge, IndieRock}.

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.

4.3 Similarity-based Querying

In order to exploit 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):

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

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, suppose we would like now to query music albums while preferring English albums (as the first priority) and then albums of genre *similar to* Alternative Rock music. We modify the `PREFERRING` section of our query to:

```
PREFERRING
  ?country = music:England
CASCADE
  ?genre ∼= music:AlternativeRock
```

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.

4.3.1 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:

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

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

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

where $N_i = len(C_i, MRCA(C_1, C_0))$ 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, the music genre *Alternative Rock* will be considered more similar to the target concept *Rock* than the genre *Dance*. 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. We can see this directly in Figure 3(a):

Example 2. *Consider the user's target concept to be* Dance. *Concepts within the same distance from this target concept (e.g. Trance, Techno, Electronic) are all considered at the same level regardless of their IS-A relation with* Dance.

This naïve approach, being less intuitive, can however serve well as a benchmark method for comparing other methods.

4.3.2 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 hierarchy-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 music ontology: the user has asked for a music album of the genre Trance (possibly among other criteria). There is no album of this genre or a sub-concept of Trance which satisfy the request. We do consider an album of genre Electronic and an album of genre Techno. Up until now, these two concepts will be considered as equally similar to Trance (both with distance 3 from Trance). The problem here is that it may well be argued that Techno is more similar to Trance than Electronic since Trance and Techno *share more specific information*. Both Trance and Techno inherit from Dance which means that they share the special properties of Dance that distinguish it from other types of Electronic music. (Wu & Palmer 1994) presented a similarity measurement (in the context of linguistics) which takes the level of shared information into account:

$$Sim(C_1, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3} \qquad (4)$$

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*).

Example 3. *In our example,* $Sim(Trance, Electronic) = \frac{2*2}{2+0+2*2} = 0.67$ *while* $Sim(Trance, Techno) = \frac{2*3}{1+1+2*3} = 0.75$ *and thus Techno will be considered more similar to Trance than Electronic and will dominate it in the context of music genres.*

As for (2), a major drawback in using this method to compute similarities in ontology hierarchies is that it does not respect the *IS-A* relation axiom, as can be seen directly in Figure 3(b):

Example 4. *Let us consider again the user's target concept to be* Dance*. The concept* Electronic *has a similarity measurement of* 0.8 *to the target concept while the concept* ProgressiveTrance*, being a sub-concept of* Dance *has a smaller similarity measurement of* 0.75 *and thus considered less similar to* Dance *than* Electronic.

It may be argued that even though Progressive-Trance, being a very specific type of Dance music, is indeed Dance, due to its specific characteristics it may have gone far enough from this target concept to be considered less similar to it compared to a concept outside that target concept. We find no justification for making such an argument mainly due to the fundamental 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.

4.3.3 Most Specific Shared Information w.r.t Ontology Depth

In this paper we 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 (4) by reducing the similarity measurement in relation to the depth of the compared concepts and the maximal depth of the relevant part of the ontology (further details are in Section 5.1). We propose a novel similarity method, based on (Wu & Palmer 1994), which has three interesting properties:

1. It considers two concepts more similar if they share more specific information.
2. It respects the *IS-A* relation axiom as described above.
3. Within a sub-graph, it will consider a concept more similar to a target concept according to the communicated level of specificity described by this target concept.

Property 3 means that when the user specifies a certain target (most preferred) concept, the sub-concepts below this target concept will be ordered according to their distance to the target concept (the 'closer' distance, the more similar they are). The intuition behind this is to respect the user's communicated level of specificity (given by the depth of this target concept in the ontology) considering concepts which are 'closer' to this level of specificity to be more similar. This is measured by the following similarity metric which determines the similarity of concepts C_0 and C_1.

$$Sim(C_1, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3 + AVG} \qquad (5)$$

Where N_1, N_2 and N_3 are defined as in (4), 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. Note that in practice we then normalise the similarity measurement to be between 0 and 1 by dividing it by the similarity of the target concept to itself. This way we ensure that the similarity between the target concept and itself will always be 1 (which accords with intuition).

Example 5. *In our running example,* $Sim(Trance, Electronic) = \frac{2*2}{2+0+2*2+2} = 0.5$ *while* $Sim(Trance, Techno) = \frac{2*3}{1+1+2*3+1} = 0.67$ *and thus, as for the previous method, Techno will be considered more similar to Trance than Electronic. But now, as opposed to (4), this methods will still give a smaller similarity measurement (*0.53*) between Dance and Electronic and a greater similarity measurement between Dance and any sub-concept of Dance, e.g.* 0.67 *for the similarity between Dance and ProgressiveTrance.*

We can prove that (5) 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 (5). 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* $b_2 \prec_{P(C_0)} b_1$*).*

It can be observed that Equation 5 induces a total preorder over concepts satisfying asymmetry and transitivity. The asymmetry property comes directly from point 2 above. The power of this method is that it encapsulates all the discussed intuitions. It satisfies the *IS-A* relation axiom, always considering sub-concepts of a target concept more similar than non-sub concepts w.r.t this target concept. It also considers concepts that share more specific information more similar unless the previous statement does not hold. We discuss this method further in the analysis (6).

4.4 Exploiting Property Structure for Querying with Preferences

Similarly to the way concepts in an ontology can be organized in an hierarchical manner, an hierarchical structure can be applied to properties (roles) as well. RDF provides us with the ability to define sub properties through the $subPropertyOf$ operator. Typically, these properties will have the same domain and range. Consider the property structure given in Figure 2 defining a (partial) structure over roles in music album production. All properties which inherit from the property *participates* will have the signature: $participates(MusicPerson) \rightarrow MusicAlbum$. It is a fact that some of the best known musicians, as well as creating their own music, also participate in the production of other albums (e.g. Brian Eno, Robert Fripp etc). By nature, the work engineered by some musicians, for example, will be significantly different to the music they compose or perform. On the other hand, it is safe to say that a certain individual's participation in an album as an electric guitar performer will have more similar contribution to an album where the same individual played the bass guitar than an album where they participated as a record producer. In our preference reasoning, we exploit this structure in order to provide more accurate results in a similar manner to the way we exploited conceptual structure. Similarly to structural-based operators available for preferences over concepts, the syntax for constructing preferences over properties in the ontology is available through the *Sim* and the *IS-A* operators:

$$\texttt{?prop} \sim= C_0$$

Where `?prop` is a variable which has to appear as the predicate in some triple block in the `WHERE` clause of the SPARQL query. The same syntax can be applied for using *IS-A*. For example:

```
Prefix music:
    <http://example.com/music.owl#>
SELECT ?id ?genre ?property
FROM <http://example.com/music.owl>
WHERE {
  ?id music:hasGenre ?genre.
  ?id ?property :MyMusicPerson.
  ?id rdf:type :MusicAlbum. }
PREFERRING
  ?property ~= music:keyboards_in
```

The semantics of the preference reasoner will be similar to the semantics for reasoning with concepts. From the technical point of view, SPARQL allows us to place variables as properties. These will be bound through the execution of the query to anything that will match the domain and range. In our example, the variable `?property` will be bound to any function from *MusicAlbum*[1] to *MusicPerson.*[2] Once the variable is bound we use the *RDFS* property *subPropertyOf* to analyze the property structure.

5 Implementation—The *SPOQE* System

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 et al. 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). A demo of the system – *SPOQE* (Similarity-based Preferences over Ontologies Query Engine) is available online. [3] On top of the described similarity measurement methods, we supplemented this implementation by adding some further numerical preference attributes mentioned in (Kießling & Köstler 2002) such as `AROUND` and `BETWEEN`.

[1]We explicitly ensure this in the query by binding the variable *?id* to elements of type *MusicAlbum* although this is not mandatory.

[2]Assuming *MyMusicPerson* is of type *MusicPerson*.

5.1 Reasoning in Context—Conceptual Views

When a large ontology is available, we may wish to limit our search over similar concepts in the ontology purely because some parts of the ontology are not relevant to our query. For example, if part of our ontology relates to music and another part to sports, when looking for music recommendations there is no need to consider concepts in the sports part of the ontology. In such cases, it makes sense to limit the search for similar concepts and, in fact, denote that sections of the ontology are irrelevant to other sections and need not be searched over. (Balke & Wagner 2004) refers to these sections in an ontology as *Conceptual Views* and models these as the concepts which inherit directly from the top concept *Thing*. In this paper, we allow the ontology designer to be able to make the decision in regards to what constitutes a conceptual view. To effect this we adopt a unary function $isConceptView(C)$ that takes a concept as argument. This denotes that only concepts $D \sqsubseteq C$ are to be considered similar to one another (according to the metric adopted). Other concepts are considered to be "irrelevant" with $Sim(D, E) = 0$ for concepts $D \sqsubseteq C$ but $E \not\sqsubseteq C$. In other words, this feature serves to restrict searches to relevant portions of the ontology, i.e. Conceptual Views. This feature can also significantly limit the search space when performing our structure-based reasoning.

5.2 Computing Top-k Elements

Our system offers two query models: the user can either search for the elements which best match their preferences (BMO—Best Match Only query model) or, retrieve the top-k elements. Using top-k is useful not only in order to experiment with preference orderings but also since it may be the case that a possible result binding is dominated by one (or a few) other result bindings but still very similar to the optimal solution and may simply be 'good enough' (especially when dealing with similarity as the basis for comparison). In BMO, a 'competition loop' takes place where once a result binding (or a database tuple) has been dominated it is no longer considered a best match and thus will not be returned to the user. In our system, top-k selection is made without losing the qualitative nature of our reasoning, i.e. without using direct scoring of individuals: while performing the competition loop we give a score to an individual not according to its content but by examining how many 'competitions' it wins, how many it loses and how many resulted in a draw (neither individuals dominated the other). This is a round robin competition loop since we do not stop when a result binding was found to be dominated. For example, a result binding b_1 competing against another result binding b_2 will receive:

- 2 points if it dominates b_2, i.e. $b_2 \prec_P b_1$.
- 1 point if it does not dominate b_2 and b_2 does not dominate it i.e. $b_2 \equiv_P b_1$.

[3]http://spoqe.web.cse.unsw.edu.au/

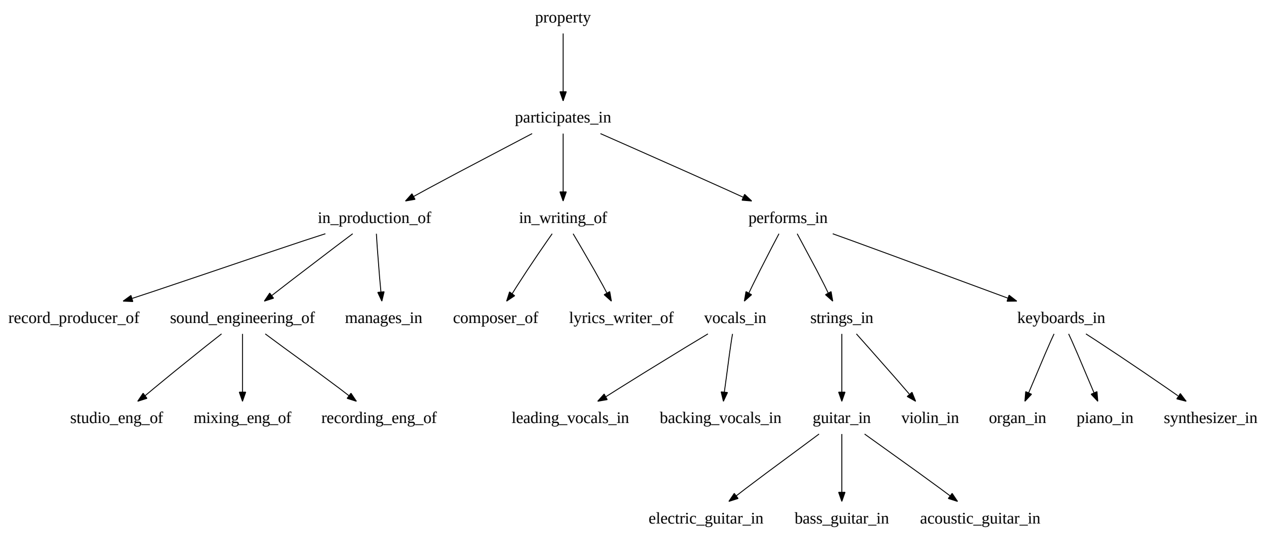


Figure 2: Ontological Property Hierarchy of Music Album Production

- 0 points if it is dominated by b_2, i.e. $b_1 \prec_P b_2$.

At the end of this process, we sort the individuals according to their accumulated score. As in BMO, the best match result bindings will appear at the top of the preference ordering but now just below them the result bindings which are closer to the preference query (but not perfectly matching it) will appear.

6 Analysis

We evaluate our similarity method by looking at some intuitive examples and examining the behaviour of the resulting total orders. We base our analysis on the concept similarity method defined in (5). Let us have a closer look at this operator. In order to preserve the properties we listed in Section 4.3.3, this operator was designed so it will have different behaviours when comparing a target concept with a descendant to when comparing a target concept to an ancestor. It can be deduced that if $C_1 \sqsubseteq C_0$ and $C_2 \sqsubseteq C_1$ then

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

$$Sim(C_2, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3 + AVG + \frac{1}{2}\Delta}$$

Where Δ is the the direct distance between C_1 and C_2. And in the case where $C_0 \sqsubseteq C_1$ and $C_1 \sqsubseteq C_2$:

$$Sim(C_2, C_0) = \frac{2 * N_3 + 2\Delta}{N_1 + N_2 + 2 * N_3 + AVG - 1\frac{1}{2}\Delta}$$

Example 6. *When assigning the target concept to be* Dance *(Figure 3(c)) we see that the similarity measurement linearly decreases for descendant concepts w.r.t* Dance *while every 'climb' up to an ancestor reduces the similarity measurement significantly (and non-linearly). Intuitively, you can say that the difference in similarity when comparing elements 'in the area of' Dance is bigger than the difference in similarity between elements which are quite 'far' from this target concept and thus the ordering according to it is less relevant. Note that for clarity we normalise the similarity measurement to be between 0 and 1 by dividing it by the similarity of the target concept to itself.*

7 Discussion

While the method that we provide here for exploiting ontological structure to enhance preferential querying leads to a much more discriminating recommendation system, there are some limitations that can be further explored. We consider two such issues here. One has to do with the expressiveness of user preference queries while the other has to do with the structure of the ontology that is used.

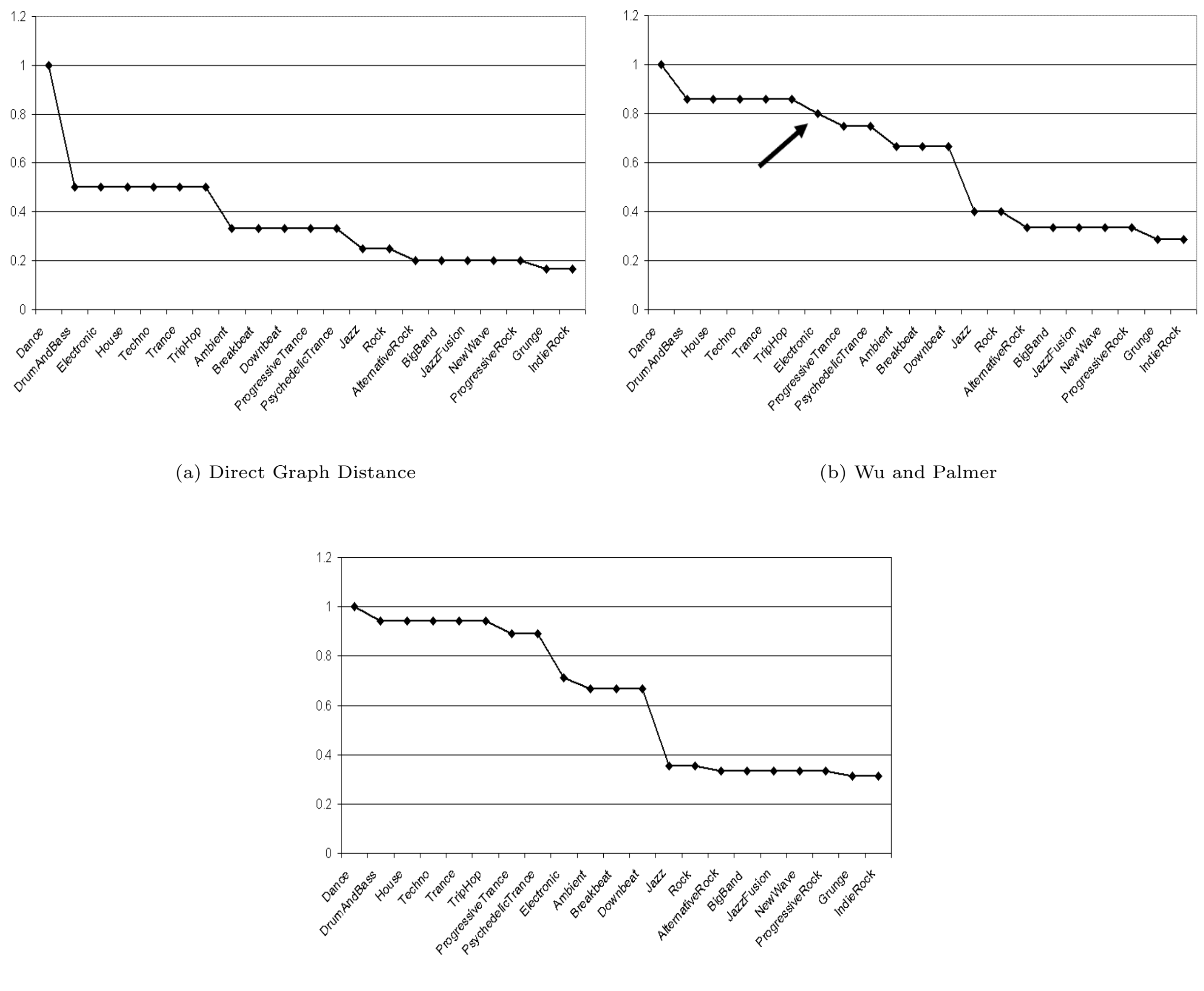
7.1 Receiving Partial Preorder from the User

In this paper the user is limited in the way they can express preferences. It is not possible to express that two classifications are equally preferred by the user for instance. Furthermore, `CASCADE` only allows for limited levels of preferences. In (Chamiel & Pagnucco 2008*b*) we propose a preference assembly method where the user can specify an ordering over concept classes which allows for classes to be equally preferred. That is, to specify a preference as a partial pre-order

$$\{C_{1,1}, C_{1,2}, \cdots, C_{1,i}\} > \cdots > \{C_{n,1}, C_{n,2}, \cdots, C_{n,k}\}$$

Each set $\{C_{l,1}, C_{l,2}, \cdots, C_{l,n}\}$ represents concepts that the user equally prefers. We only require that preferences be consistent; i.e., transitivity and asymmetry is preserved. Furthermore, we have an ontology for the domain at hand in OWL format. We can turn this partial pre-order into a total order by "filling out" (or completing) user preferences with information from the ontology by utilising notions of similarity such as that described here. We end up with a total order $C_1 > \cdots > C_m$ over all concepts in the ontology with those corresponding to concepts specified in the user preference maintaining the order imposed by the user. Concepts not explicitly ordered by the user are arranged in a consistent way so that they occur after the concept in the user ordering to which they are most similar. In this way we end up with a fine-grained recommendation mechanism like the one developed in this paper but allowing the user to express more of their preferences and hence exert greater influence over the final orderings (and therefore the final recommendation). Syntactically, this could be expressed as follows (and we could introduce a similar syntax for the IS-A relation):

$$\texttt{?var} \sim= (C_{1,1}...C_{1,i}) \ \texttt{THEN} \ (C_{2,1}...C_{2,j}) \ \texttt{THEN} \ \cdots \ \texttt{THEN} \ (C_{n,1}...C_{n,k})$$



(a) Direct Graph Distance

(b) Wu and Palmer

(c) Method Proposed in (5)

Figure 3: Structure-based similarity methods comparison w.r.t the target concept *Dance*. The x-axis represents concept classes in the ontology while the y-axis represents the similarity of these concepts to the target concept *Dance*.

7.2 Weighted Edges in an Ontological Graph

The methods described in this paper utilize graph distance to perform reasoning and enhance preferential selection. However, so far all edges of the graph (i.e., ontology hierarchy) are considered to have the same distance (or weight). This relies on the underlying assumption that all sections of the ontology hierarchy have been developed to the same 'level of specificity'. In reality some edges might describe a 'closer' relation than others. For example, in our music example we may argue that not all concept classifications at the same level of the hierarchy represent the same level of specificity in their categorisation of music. This situation can result for a number of reasons. In some cases it represents uncertainty about the domain and the classifications made in it; there may be more information regarding certain parts of the ontology while less information for other parts and this is reflected in the depth of the hierarchy. In other cases, the ontology hierarchy cannot be uniformly developed throughout. The ontology engineer has no means of defining this (besides creating bogus hierarchical levels) and it may be very difficult for them to make the different distinctions and weight the edges manually. One way of dealing with this issue is to allow the ontology engineer to attach weights to the edges in the OWL ontology graph. This however is quite laborious and tedious task. Another way is to assign graph edge weights automatically by examining the depth of a sub-graph where an edge which has a deeper (and thus richer) sub-graph under it will receive a smaller weight to demonstrate its smaller distance from the parent concept. This assumes that all parts of the ontology have been developed to the same level of specificity and may lead to better recommendations when adopting the technique described here.

8 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 (4) to introduce another (5). 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. & van Harmelen, F. (2004), *A Semantic Web Primer (Cooperative Information Systems)*, MIT Press.

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

Balke, W.-T. & Wagner, M. (2004), Through different eyes: assessing multiple conceptual views for querying web services, *in* 'WWW '04: Proceedings of the 13th international World Wide Web', ACM, New York, NY, USA, pp. 196–205.

Bradley, K., Rafter, R. & Smyth, B. (2000), 'Case-based user profiling for content personalisation', *Lecture Notes in Computer Science* **1892**, 62–72.

Chamiel, G. & Pagnucco, M. (2008*a*), Exploiting ontological information for reasoning with preferences, *in* 'Proceedings of the Fourth Multidisciplinary Workshop on Advances in Preference Handling'.

Chamiel, G. & Pagnucco, M. (2008*b*), Querying the semantic web by assembling complex user preferences and ontological structure, Technical Report UNSW-CSE-TR-0817, School of Computer Science and Engineering, University of New South Wales.

Doyle, J. & 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. & Stocker, M. (2007), The fundamentals of iSPARQL – A virtual triple approach for similarity-based semantic web tasks, *in* 'ISWC '07'.

Kießling, W. (2002), Foundations of preferences in database systems, *in* 'VLDB', pp. 311–322.

Kießling, W. & Köstler, G. (2002), Preference SQL: design, implementation, experiences, *in* 'VLDB', pp. 990–1001.

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

Prud'hommeaux, E. & Seaborne, A. (2006), SPARQL query language for RDF, Technical report, W3C Candidate Recommendation.
URL: *http://www.w3.org/TR/rdf-sparql-query/*

Sarwar, B. M., Karypis, G., Konstan, J. A. & Reidl, J. (2001), Item-based collaborative filtering recommendation algorithms, *in* 'World Wide Web', pp. 285–295.

Schickel-Zuber, V. & Faltings, B. (2006), Inferring User's Preferences using Ontologies, *in* 'AAAI 2006', pp. 1413–1418.

Schickel-Zuber, V. & Faltings, B. (2007), Using hierarchical clustering for learning the ontologies used in recommendation systems, *in* 'KDD'07', pp. 599–608.

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

Siberski, W., Pan, J. Z. & Thaden, U. (2006), Querying the semantic web with preferences, *in* 'International Semantic Web Conference', pp. 612–624.

Wu, Z. & Palmer, M. (1994), Verb semantics and lexical selection, *in* '32nd Annual Meeting of the Association for Computational Linguistics', New Mexico State University, Las Cruces, New Mexico, pp. 133 –138.