Ontology Guided Dynamic Preference Elicitation

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

ABSTRACT

A challenge for preference based recommender systems is to elicit user preferences in an accurate and efficient manner. Eliciting preferences from the user in the form of a query that is then used to filter items from a database can result in a coarse recommendation 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 irrelevant alternatives) are returned. On the other hand, the advent of the world wide web has delivered an abundance of data at our fingertips. Information gathered from the web, reduced to structured ontologies, can prove useful in focussing preference elicitation mechanisms.

In this paper we present a preference elicitation process which allows users to communicate their preferences in a simple manner, through examples presented to them. The system then makes use of an ontology model, based on expert information and social web resources. It elicits the user's preferences guided by this ontology in an interactive and dynamic manner. We show that this leads to more effective recommendations.

We evaluate our work through empirical experiments and discuss the results in terms of preference elicitation coverage as well as the prediction accuracy of the preference model.

1. INTRODUCTION

Modelling user preferences and exploiting preferential information to assist users in searching for items has become an important issue in product recommendation. Eliciting user preferences in an accurate manner is a difficult and challenging task. In most cases the user lacks deeper, expert knowledge of the domain to allow for a more discriminating recommendation to be determined but they know what they like. Furthermore, even if they are aware of some of their preferences, it may be difficult for them to express these explicitly in a formal language.

In this paper we develop techniques for eliciting formal preferences from the user in a seamless fashion that hides the technical details. In our work a crucial desiderata is that the user does not need to know anything about the attributes that describe items. We focus on the problem of determining the most appropriate queries to present to the user in order to accurately elicit their preferences. We do so in an interactive manner which focuses on the user experience by utilising ontological information available on the World Wide Web through social web resources and expert libraries. By so doing we develop a complete system for personalisation that cushions the user from having to know the formal details of how preferences are represented and how preference queries over a database of items are formulated. We define four types of preference elicitation queries and show how the dynamics of these methods are designed to gather sufficient information from the user to quickly and accurately determine their preferences. This is the main contribution of the paper. We also show how these processes can be achieved more efficiently by clustering the item space. Finally, we briefly discuss how we use standard statistical methods together with these ideas in order to establish the user preference profile.

Research on personalisation has provided a rich literature in recommendation systems [1, 2, 3]. In more direct relation to preference elicitation, [4] provides a significant framework for formalising an elicitation process. However, this framework assumes user familiarity with the domain which contradicts the assumption of this research. [5, 6] model preference elicitation in terms of utility elicitation but these processes are not directly guided by a model of the data. Previous work in supplementing user preferences with ontological information is limited. Ontology based similarity systems have been presented in [7, 8] but provide for only basic features. [9] provide methods for augmenting collaborative product recommendation with information derived from taxonomies. However, none of these approaches have tackled the entire problem of eliciting formal preferences from a user and enhancing them with ontological information in an interactive manner. It is this problem that we address here in its entirety.

1.1 Motivating Example

One of the most obvious domains in which to evaluate the techniques developed here is that of user musical preferences. When it comes to music selection, people often express their preferences in terms of individuals, either via their favourite artists or simply by pointing to pieces of music which they prefer or do not prefer. However, people seldom possess a deep understanding about the features and characteristics behind the music and thus may find it difficult to search for new music based on their personal tastes. Fortunately, in the World Wide Web there exist digital libraries which contain information composed by experts in this domain or by users with special interests. One such major public library is MusicBrainz[1] which is essentially a social web service in the mu-

[1] `http://musicbrainz.org/`

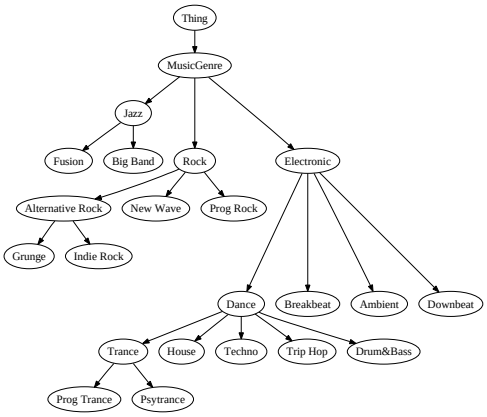

Figure 1: An Example Ontological Concept Hierarchy of Music Styles

sic domain. Through this service, users can input information in a structured manner from which it is relatively easy to infer an ontology.

For example, consider the concept hierarchy in Figure 1. This information is composed by musical experts and describes a hierarchy of musical styles. While information such as this is well understood and formalised by domain experts, we do not assume the user has extensive knowledge of the domain at all. Still this information can be readily exploited when reasoning with preferences. Note that in reality these structures tend to be far richer than this example and often describe an attribute in the domain in a comprehensive manner. In our work, we remedy this problem by eliciting complex user preferences via simple queries which the user may feel more comfortable answering. We illustrate our ideas through examples from the music domain and provide experimental results in music preference elicitation to evaluate our approach. However, it is important to note that the techniques we develop are general and can be applied to a variety of domains.

2. BACKGROUND

We wish to express preferences formally and in a way that allows us to query a database of items using these preferences. In this section we cover the necessary background to understand how we can query a database with preferences and, more specifically, how we can query an ontology-based database with preferences. In particular, we will see how we can utilise the structure of an ontology to more accurately reason with preferences so as to provide the user with a recommendation or an elicitation query.

2.1 Basic Preference Querying

In the context of database systems, [10] introduce the ability to query with preferences using an extension of SQL. 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 operators in terms of *likes*/*dislikes*(e.g. `=`, `<>`, IN) and numeric constraints (e.g. `<`, `>=`, `BETWEEN` and `AROUND`). In order to allow complex preference construction, two binary preference assembly operators are introduced, namely, the *Pareto* operator for considering two preference constructs as equally important, and the *Cascade* operator for prioritising one preference constructor over another.

The SPARQL query language is a W3C Recommendation and considered to be a vital tool for querying ontological information. [11] introduces an extended version of SPARQL (referred to as P-SPARQL) which follows the same ideas as [10] in introducing preferences as soft constraints into the language. It forms the basis of the implementation of our approach.

2.2 Utilising Ontological Structure for Querying with Preferences

In previous work [12] we extended P-SPARQL by exploiting the information in an expert supplied ontology to further refine the results of preference queries. This work solves the problem that plagues coarse preference queries, namely matches are not sufficiently distinguished. It also allows for the construction of similarity-based queries. We only present the basic ideas here and refer the interested reader to [12] for a more extensive coverage of these details.

In order to exploit the hierarchical structure of an ontology, we developed a method for computing categorical similarity between concepts in a $TBox$. We use this similarity method for performing preference querying over ontological information. We introduced a new Boolean operator $Sim(C_1, C_2)$ (is similar to) that tells whether one binding result (b_2) is preferred to another (b_1) w.r.t the user preference P:

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

where $C_0 \in Concepts$ is a user preference 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. In other words result binding b_2 is preferred to result binding b_1.

EXAMPLE 1. *Suppose we would like to query music albums while preferring albums of style similar to Alternative Rock music (as the highest priority) and then albums released around the year 2001. Our query has a* `PREFERRING` *section as follows:*

```
PREFERRING
  ?style ~= music:AlternativeRock
CASCADE
  ?year AROUND 2001
```

Where $\sim=$ *is the syntactic version of the above 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 the similarity between concepts in an ontology, each reflecting a different rationale. In [12] we develop a novel similarity method, based on [13], 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 which means that a concept will always be considered more similar to any of its subconcepts than other concepts.
3. Within a sub-graph and given a preference concept, it will consider a concept more similar to this preference concept according to the communicated level of specificity described by this preference concept.

Property 3 means that when the user specifies a certain preference concept, the sub-concepts below this concept will be ordered according to their distance to this preference concept (the 'closer' the 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 preference 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.

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

Where N_1, N_2 are the distances from the concepts C_0 and C_1 to their MRCA (most recent common ancestor) 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*), 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 preference concept to itself. This way we ensure that the similarity between the preference concept and itself will always be 1 (which accords with intuition).

EXAMPLE 2. *In our example (Figure 1), suppose we would like to calculate the similarity values of music styles in relation to the user preference concept Trance. The concept Ambient will have the similarity value* $Sim(Trance, Ambient) = \frac{2*2}{2+1+2*2+1.5} = 0.47$ *while the concept class Techno will have the similarity value* $Sim(Trance, Techno) = \frac{2*3}{1+1+2*3+1} = 0.67$. *Therefore, Techno will be considered more similar to Trance than Ambient. However, the similarity value between Trance and Techno will be smaller than the similarity value between Trance and any of its sub-classes, e.g.* 0.84 *for the similarity between Trance and ProgressiveTrance.*

2.3 Querying with Complex Preferences

This idea can be easily extended to the case where we have elicited the user preferences in terms of a partial pre-order rather than a single concept [14]. We would still like to be able to query our database of items with these preferences. The idea is to turn this partially specified preference ordering into a total pre-order by "filling out" (or completing) the user preferences with information from the ontology while utilising the notion of similarity. The position of every concept in the total pre-order will be determined by looking at their maximal similarity to any of the user ordered concepts. Concepts which are most similar to a user ordering concept will be then ordered according to their similarity to it. The intuition behind this is that we exploit the ordering given to us by the ontology (w.r.t a similarity measurement) without contradicting the preference ordering explicitly expressed by the user. Note that this also allows expressing indifference between two concepts.

EXAMPLE 3. *The preference*

{AlternativeRock, ProgRock} `THEN` *{Electronic}*

means that the user prefers a song with style `AlternativeRock` *or* `ProgRock` *to one with style* `Electronic`. *Given a similarity method with the semantics mentioned above, and given the ontology of music styles above, the total pre-order created will be:*

{AlternativeRock,ProgRock}
{Grunge,IndieRock}
*{*other rock concepts*}*
{Electronic}
{Ambient,Breakbeat,Dance,Downbeat}
*{*other Electronic concepts*}*

AlternativeRock *and* ProgRock *perfectly match the first preference and appear at the top of the total pre-order. Concepts are then ordered according to their similarity to these preference concepts until we reach a concept more similar to the second preference: this is the concept* Electronic *(which perfectly matches the second preference). Concepts are then ordered according to their similarity to* Electronic *so as to complete the total pre-order.*

Another very essential enhancement this work provides is the ability to query the top-k elements in relation to the user preferences (in addition to querying the best match only) while preserving the qualitative nature of our reasoning. An implementation of these methods has been completed based on the *ARQ* SPARQL query engine (a *Jena* based query engine).

3. DYNAMIC PREFERENCE ELICITATION

The goal of our work is to provide users with personal recommendations that accurately reflect their preferences. In this paper, we concentrate on the *preference elicitation problem*, i.e. the problem of selecting *preference elicitation queries* to present the user in order to elicit their preferences. The resulting preferences are subsequently used to query a database of items as explained in the previous section. More accurately, we focus on the problem of selecting a series of such queries in an iterative and dynamic manner, i.e. with respect to both the user response to the queries presented to them as well as the system aim to cover certain possible preferences during this process. The attribute space of the domain, in our case, is defined via a domain ontology provided by a domain expert or social web resources which classify items in the domain according to multiple attributes. It is also important to point out that these items could be multiply classified to some attributes, i.e. have more than one value on certain attributes. This will influence the type of P-SPARQL queries we will choose in order to select and present items to the user (as opposed to more straightforward query relaxation techniques).

3.1 User Interaction

As described above, a basic assumption in our work is that users do not possess the expert knowledge which allows them to precisely and explicitly communicate their preferences in terms of the attributes in a domain. However, they may well have preferences which it is our job to elicit from (and for) them. Since we cannot query the users about those attributes directly, we will limit our preference elicitation queries to ranking individuals only and collect the information associated with them, building a preference model as we go. More specifically, the user is presented with individuals and asked to rate them as either *liked*, *disliked* or *neutral*. With this feedback (see top-left of Figure 2) the user's predicted preference order is modified by combining the statistical score and confidence interval for how much an attribute value is liked by the user to produce a partial pre-order representing their preferences. This process continues again with the elicitation method moving between *exploration* and *exploitation* phases in order to determine which item to present to the user next for their consideration and also to ensure that a sufficient cross-section of the item space is presented in order to obtain an accurate preference order.

We focus on the elicitation of user preferences by developing an elicitation technique that is itself guided by the expert supplied ontology. We will then show how this process can be made more efficient by clustering the items in the ontology. In the rest of the paper we will present this preference elicitation process and

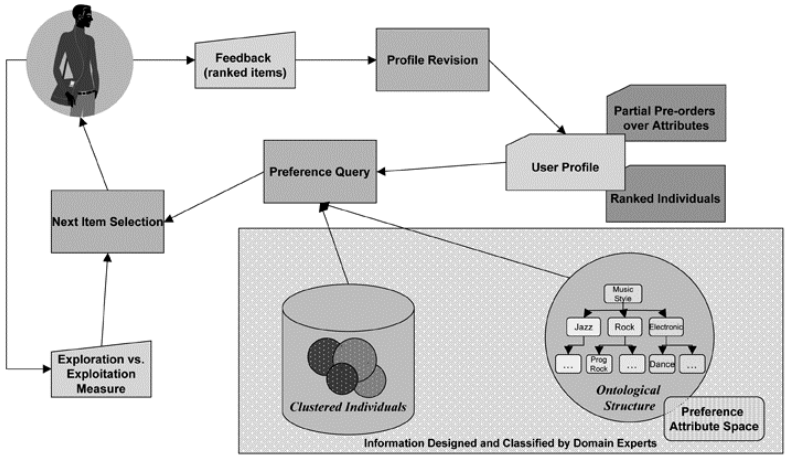

Figure 2: A High-Level Schema of the Preference Elicitation Process: users express their preferences over individuals presented to them ('Feedback' box in the figure). The system will then update their preference profile consisting of a partial pre-order over certain attributes as well as the user ranking history. The system will then consult an ontology in order to guide the next elicitation query (in terms of a new individual to be ranked – 'Next Item Selection' box). The system goal is to obtain confidence in previously elicited preferences (exploitation) as well as to cover new preferences (exploration).

its clustering-based extension. We also briefly discuss the building of a preference model on behalf of the user and its evaluation.

3.2 Ontology Guided Preference Elicitation

As described above, the goal is to elicit implicit user preferences over some predefined attribute space through the rankings of instances that the system offers to the user. Once we have enough information about the user's rankings, we can start building the user preference profile. The dynamics of this process are designed so we can confidently elicit as many preferences as possible and at the same time allow for an interesting interaction between the user and the system. The idea here is to have a trade-off between wanting to learn more about the current hypothesis regarding the user's preferences and wanting to more widely explore the user preference hypothesis space as well as keeping the user happy and engaged. In the first case, we would like to become more confident about the user preferences in a particular area of the ontology while in the second case we would like to explore more areas in order to discover more preferences. From the user perspective we can view this trade-off in terms of *exploitation* of previously elicited preferences and *exploration* of new (as yet unrevealed) user preferences. Algorithm 1 shows the main interaction loop the system follows. At any given stage, we will look at a fixed size window of the user's last interactions (via the global variable $window$) in order to determine the next *preference elicitation query type*. We define four types of preference elicitation queries, namely: *similarity queries*, *queries with controlled dissimilarity*, *exploration queries* and *exploitation queries*. *Similarity queries* will search and offer the user an item similar to those last seen; *queries with controlled dissimilarity* will return an item which is less similar to these items (as determined by the ontology); *exploration queries* will return an item which is classified to sufficiently different attribute values to items last presented while *exploitation queries* will return an unseen item which holds a high value of information (see Section 3.2.3). At any stage the system has to make a decision in regard to the next preference elicitation query (see Algorithm 2), which determines the dynamics of the system. In each of these queries we call the function $getItemRelated()$ to return items at the top, middle or bottom of the ordered result set. We discuss the main ideas behind these decisions in the remainder of this section.

Algorithm 1 Elicit Preferences

$elicit()$
  $window \leftarrow \{\}$
  loop
    $query \leftarrow nextPreferenceQuery()$
    $response \leftarrow userResponse(query)$
    $window \leftarrow window \cup \{query, response\}$
    $updatePreferenceOrdering(query, response)$
  end loop

3.2.1 Similarity Queries

When receiving a positive response from the user, it is quite natural to form a hypothesis which entails that the information we are encountering at the current stage of elicitation is classified to attribute values which will be highly ranked in the user preference profile. Querying the user about similar information is desirable in order to either gain further confidence in this hypothesis or alternatively contradict it in which case we will conclude that these high rankings were due to noise. The way we achieve this is by executing similarity-based P-SPARQL queries which return the top-k individuals in relation to the current attribute values. Due to the nature of these selections, the resulting individuals will be classified not only to the exact same attribute values the user has ranked highly but also to values which are highly similar: the procedure $getItemsRelated(item, x, y)$ executes a P-SPARQL query and returns items x to y when ordered w.r.t the attributes $attr(1) \ldots attr(n)$ associated with a given $item$ where x and y specify an integer range of records. A simplified version of such a query is given below:

```
SELECT RECORD x TO y
PREFERRING
  attr(1) ∼= item.attr(1)
AND
```

Algorithm 2 Next Item Selection

```
nextPreferenceQuery()
  possibleQItems ← {}

  queryType ← establishQueryType()

  if queryType = Similar then
    possibleQItems ← getItemsRelated(lastQItem, 1, k)
  else if queryType = ControlledDissimilarity then
    possibleQItems ← getItemsRelated(lastQItem, k, l)
  else if queryType = Explore then
    possibleQItems ← getItemsRelated(lastQItem, l, m)
  else if queryType = ExploitPreferences then
    possibleQItems ← getItemsWithHVI()
  end if
  return rand(possibleQItems)
```

Algorithm 3 Establish Preference Query Type

```
establishQueryType()
 1: if |window| < winSize or negResponse(window) ≤
    negThr then
 2:   queryType ← Similar
 3: else if globalNegResponse() ≤ globalNegThr then
 4:   if previousQueryType = Similar then
 5:     queryType ← ControlledDissimilarity
 6:   else
 7:     queryType ← Explore
 8:   end if
 9:   window ← {}
10: else
11:   queryType ← ExploitPreferences
12:   window ← {}
13: end if
14: return queryType
```

```
  attr(2) ∼= item.attr(2)
...
AND
  attr(n) ∼= item.attr(n)
```

We keep gathering preferential information about these values until a certain threshold is met. This threshold will also be influenced by the number of negative responses we receive from the user where in case this number is high, the number of iterations we will spend on the current values (e.g. current branch in the ontology) will be reduced. Algorithm 3 shows the management of these parameters which will establish the type of preference elicitation query we will use next. We adopt this type of query when the window of user interactions has not yet reached its maximal size and the response we get from the user is not negative enough (line 1).

EXAMPLE 4. *Suppose the user has ranked an item classified to music styles* House *and* Techno *and with the release year 2002. The P-SPARQL query with respect to this item will then be:*

```
SELECT RECORD 1 TO k
PREFERRING
  ?style ∼= :House
AND
  ?style ∼= :Techno
AND
  ?releaseYear ∼= 2002
```

Algorithm 4 Select Items with High Value of Information

```
getItemsWithHVI()
  partialPreorder ← collectPreferences()
  totalPreorder ← computePreferences(partialPreorder)
  value ← null
  i = 0
  while value = null do
    if totalPreorder[i] ∉ partialPreorder then
      value ← totalPreorder[i]
    end if
    i++
  end while
  return getItems(value)
```

We will then randomly select an item from this result set and present it to the user as the next elicitation query. We can assume that this elicitation query item will have similar attribute values to the previous one. For example, music styles House *and* Trip Hop *and with the release year 2003.*

3.2.2 Querying Diverse Items

Let us consider now the case where we have elicited a sufficient amount of information about a certain branch in the ontology. We would like to change our preference elicitation queries so that we can elicit user preferences about other parts of the ontology. If the reaction of the user was sufficiently positive in the previous phase, in order to make the transition between items smoother and offer a better user experience we select items within a limited distance from the previous items. In this case $establishQueryType$ will return query type $ControlledDissimilarity$ which will modify the call to $getItemsRelated$ by selecting elements from the middle of the result set (controlled by the parameters k and l predetermined as a function of the size of the item-space).

EXAMPLE 5. *Consider the item ranked in Example 4 and suppose we have now reached the stage where we would like to query items with certain diversity to this item. We will execute the same P-SPARQL query as before with the parameters k and l. We can assume that the next elicitation query item will have attribute values which appear within a certain limited distance to the previous one. For example, music styles* Dance *and* Ambient *and the release year 2000.*

However, if the user's response to the previous phase was not positive, then eliciting preferences from that particular area in the ontology may no longer be desirable and we can try moving further afield. Since we are dealing with hierarchical structures, it is relatively easy to control the selection of preference elicitation queries and increase the level of dissimilarity our P-SPARQL query returns. In that case we will execute an *exploration query* which will select items from the bottom of the similarity query's result set and will return an item with greater distance than the last window. Note that once we have selected an item via these query types, we will go back to querying with similarity (which means we will reset the window) until the threshold criteria will entail querying with certain diversity again.

3.2.3 High Value of Information

In many domains, users may have preferences over more than one type of individual. In ontological terminologies, they may hold preferences over different branches in the hierarchy. Therefore, once we have gathered enough confidence in the preference elicitation for a particular part of the hierarchy, it is important to be able to

explore different areas as well. In our work, preferential information with a *high value of information* are those preferences the user holds which we have not yet revealed. In order to discover these preferences we need to be able to explore individuals classified to attribute values the system is uncertain about. Since we are dealing with hierarchical structures, here again we can control the selection of preference elicitation queries. We will choose to select new attribute values about which we are uncertain of the user preferences and which are highly similar to known preferences. The way we achieve this is by computing the preferences we have obtained from the user up to this point as a partial pre-order and then searching for an attribute value which does not appear in this pre-order but is similar to an attribute value which is highly preferred in the order. The obtaining of a partial pre-order over a certain attribute from ranked individuals is briefly discussed in Section 3.4. Once we obtain the partial pre-order, we expand it to a total pre-order (as described in Section 2.3) and select an attribute value which appears highly ranked in the total pre-order but does not appear in the partial pre-order. Since this value did not appear in the partial pre-order we can deduce that the user has not yet ranked sufficiently enough items of this type. And since it appears highly ranked in the ordering, it satisfies our High Value of Information criterion. Algorithm 4 shows a simplified version of these ideas where the $getItems(value)$ procedure is assumed to execute a P-SPARQL query selecting items with a certain attribute value which equals a given $value$. In terms of user interaction, we will use this query type in order to re-start the elicitation process when repeatedly receiving negative feedback from the user (line 3 in Algorithm 3). Note that this preference query type is also used at the beginning of every user session.

EXAMPLE 6. *Suppose we have calculated the user preferences as a partial pre-order in terms of music style and the resulting ordering is:*

{AlternativeRock, ProgRock} THEN *{Electronic}*

Calculating the total pre-order, the music style IndieRock *will appear in a high position in the total pre-order and will be used to select the next preference query item (see Figure 1).*

3.2.4 Complexity and Performance Issues

The main difficulty when it comes to database querying with preferences is the complexity of queries. Even though the basic ideas behind the Pareto and Cascade operators are simple, the nature of comparison queries makes them quadratic in the number of items. Since we are dealing with large item spaces and since the execution of such queries could be very frequent, it is crucially important to make these queries more efficient. Furthermore, in our case, when we wish to explore diverse items w.r.t a ranked item, it is unnecessarily expensive to compare all particular individuals to each other in order to get this desired effect. In order to avoid that, it would be better to focus our attention on part of the item-space. We do so by grouping our individuals into some high-level clusters and reason over these clusters before we dive into the actual selection of particular preference query items. We discuss this enhancement in the next section.

3.3 Making Preference Elicitation Faster with Clustered Ontologies

There are many ways to cluster categorical and numerical information. In our work, we make use of a particular structure, namely *explicit semantic relations*, which is knowledge described in the ontology by way of direct semantic associations between individuals. This is usually done via *roles* which describe a direct semantic relation between individuals of the same type. For example, $similarTo : Artist \mapsto Artist$ to describe that one artist is similar to another. These structures can be viewed as weighted or unweighted graphs in which elements of the item-space are nodes of the graph and the semantic relation determines the edges (e.g. $similarTo$). We find that in many domains there exist classifications which form this kind of structure and induce a semantic network and in some cases a similarity network. In our example, there exists a similarity network between artists, available in music libraries etc., which can be viewed as a graph (in our case, an unweighted graph) and allows us to generate a high level hierarchical clustering structure.

3.3.1 Clustering Semantic Relatedness

The main idea behind clustering similarity graphs is to look for highly connected sub-graphs. The way this is done is through methods which are based on *minimum cut trees* within the graph. Given a graph $G(V, E)$, a cut is a set of edges whose removal will disconnect the graph. A minimum cut is a cut with a minimal number of edges. [15] presents a clustering algorithm which computes minimal cuts iteratively and looks for sub-graphs with a high level of connectivity. At the end of this process, the graph will be partitioned into j clusters where j is the number of clusters and is unknown at the beginning of the process. In our work we adopt this technique since it has a simple generalisation into a hierarchical clustering method. This is a very desirable effect since we are dealing with ontologies which treat hierarchical structures straightforwardly. The resulting cluster structure gives a high-level classification of individuals with some similar characteristics.

3.3.2 Querying the Clustered Item Space

The basic idea behind our enhanced preference elicitation technique is similar to what we have seen before. The main difference, now that we have our items clustered, is that on every elicitation query, we will first determine which cluster we would like to select from and then execute the P-SPARQL query limiting the search to that cluster (and thus limiting the complexity to the size of the cluster). Algorithm 5 (which now replaces Algorithm 2) shows the revised preference query item selection procedure with clustering. The selection of the particular cluster will depend on the preference query type and the cluster of the item previously ranked according to the same principles we discussed in 3.2.

3.4 Building the Preference Model

During the preference elicitation process, users encounter a variety of different items whose attributes are described in terms of concept classes. These concept classes are used to represent their preferences. We now build a partial pre-order preference relation over those classes on behalf of the user. The way we approach this is through standard statistical reasoning where we look at the mean score of individuals classified to each class taking into account the confidence of this score. For each concept class, we compute its score and confidence. The score of a class is measured in terms of the probability that if we draw an individual classified to this class, it will be ranked positively, negatively or neutrally. A requirement for a class to be included in the partial pre-order is that its confidence measure is at most equal to some predetermined constant which determines how sparsely ranked a concept might be to be included in the partial pre-order. We will order the classes in the partial pre-order according to their score and confidence where two classes will be ordered at the same level if they do not have a significant statistical difference. This is computed in relation to their overlapping confidence interval ratio. This partial pre-order pref-

Algorithm 5 Next Item Selection (revised)

$nextPreferenceQuery()$
  $possibleQItems \leftarrow \{\}$

  $queryType \leftarrow establishQueryType()$

  if $queryType = Similar$ **then**
    $c \leftarrow lastQItem.cluster$
    $possibleQItems \leftarrow getItemsRelated(lastQItem, c)$
  else if $queryType = ControlledDissimilarity$ **then**
    $c \leftarrow getSimilarCluster(lastQItem.cluster)$
    $possibleQItems \leftarrow getItemsRelated(lastQItem, c)$
  else if $queryType = Explore$ **then**
    $c \leftarrow getDiverseCluster(lastQItem.cluster)$
    $possibleQItems \leftarrow getItemsRelated(lastQItem, c)$
  else if $queryType = ExploitPreferences$ **then**
    $c \leftarrow getClusterWithHVI(lastQItem.cluster)$
    $possibleQItems \leftarrow getItemsWithHVI(preference, c)$
  end if
  return $rand(possibleQItems)$

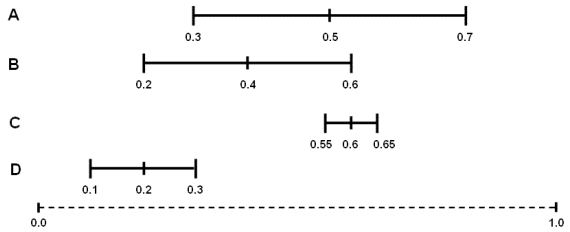

Figure 3: Overlapping Confidence Intervals Example

erence model will be then expanded to a total pre-ordering using (Section 2.3). Due to lack of space, we do not present the mathematical details of this method. However, this can be easily derived using standard statistical inference.

EXAMPLE 7. *Consider the scoring statistics of four classes as gathered by the system according to user responses in Figure 3 given in terms of confidence intervals* $[0, 1]$. *Computing the overlapping confidence interval ratio of classes* A *and* B*, we can infer that* A *and* B *should be considered at the same preference level. On the other hand, class* C *will be strictly preferred over class* A*. Similarly, we can compute the overlapping confidence interval ratio of classes* B *and* C*. The resulting partial preference pre-order in this example will be* $C \succeq \{A, B\} \succeq D$.

4. EXPERIMENTAL RESULTS

We evaluate our preference elicitation methods through a series of experiments in the domain of music. We have implemented a music preference elicitation system we call *The Music Preference Explorer* which selects and suggests music to users according to the preference elicitation methods described above. We ran an experiment with the system and several users.

4.1 Data Model

The data model we use is an ontology we collected from information available on the web. The ontology consists of 134 artists and around 1200 tracks from 338 albums of popular rock, pop and electronic music. We created a hierarchy of 137 styles which is arranged as a DAG and classifies each album to at least one style (and around four on average). We also collected similarity information between artists and clustered our data set into 26 different clusters arranged hierarchically according to the method described in 3.3. The cornerstone for this dataset is the *MusicBrainz* library. However, since this is a new service, some classifications needed to be collected manually from other resources on the web.

4.2 User Trials

When a user logs into the system, they are presented with a piece of popular music and can specify their ranking as one of the following options: they can either say they like the music they are listening to, they do not like it or that they are not sure about their preferences. The system then collects those rankings and gradually builds a preference model on behalf of the user in terms of the music style classified. The system executes the preference elicitation cycle we described in 3.2. If the user responds positively to certain types of music, the system will suggest music which is similar according to its model. However, it will gradually select music with a different style and at some point explore styles which are quite different (and notify the user that it has done so). If a user gives a negative response to a certain type of music, this process will occur faster. We ran the trial on 22 users and present the results in the following section.

4.3 Results

We measure the performance of our methods in terms of the prediction accuracy of the elicited preference model as well as the coverage of different preference levels.

4.4 Preference Model Accuracy

Our first test measures the accuracy of the elicited model by trying to predict the user response on random tracks according to the created preference model. Figure 4 shows the accuracy of correctly predicting whether a user will say they like a piece of music and when they say they will dislike it. We compute the mean rank of a track (according to its classified information) and compute a threshold (depending on the number of preference levels the user holds) to differentiate between predicting whether the track will be liked or otherwise. In order to measure the growth in accuracy, we recreated the user ranking cycles by first building their preference model using the first 20 tracks (in order) and gradually adding tracks until we used all the tracks they had ranked. We have clearly shown a growth in prediction accuracy and reached over 80 percent accuracy on average from around 160 examples. Note that these results were achieved by basically looking at a single (probably very central) attribute and performing well on average.

4.5 Preference Coverage

Finally, since we are dealing with a dynamic process where our methods are designed to cover different user preferences over time by allowing the user to explore new preferences, we measure preference coverage in terms of preference ordering and clusters covered. Figure 5 shows the growth in the number of different preference ordering levels we elicit over time. This includes the partial pre-order computed through the statistical analysis described in 3.4 as well as the number of total pre-orders realised after augmenting this pre-order with further information from the ontology. This gives some idea about the dynamics of the system and shows the growth in elicited information over time. In addition to the number of preference levels, we also measure the number of clusters the user has visited since this is a very central part of the elicitation

cycle. Note that when reaching over 180 ranked items, the number of preference levels starts to drop. This is due to the overlapping confidence interval matching we described in Section 3.4: some ordering levels are merged when gathering more statistical confidence. Note also that when reaching this number of rankings, our model prediction accuracy reaches around 85% (Figure 4). This shows that the preference ordering matching was indeed justified and better reflected the user's true preference.

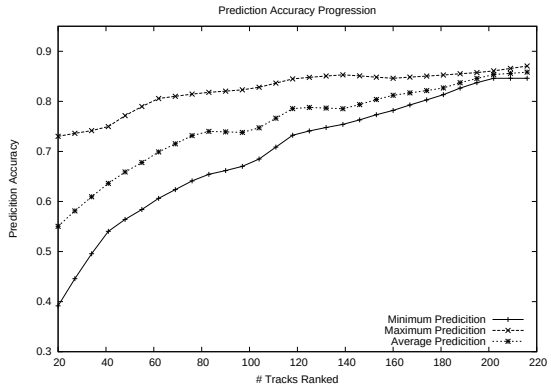

Figure 4: Prediction accuracy progression: shows the accuracy of predicting whether the user will like/dislike a track according to the elicited preference ordering

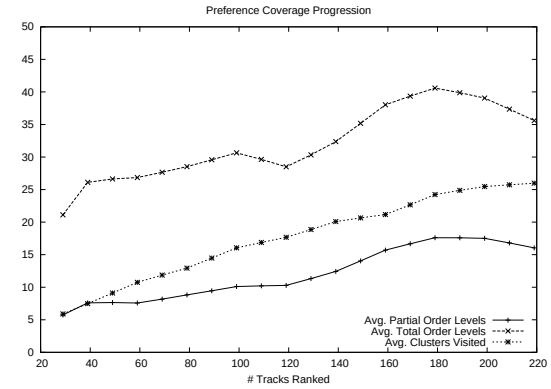

Figure 5: Preference coverage progression: shows the progression of preference ordering levels (partial and total) as well as the number of clusters visited. Note that when reaching over 180 ranked items, the number of preference levels starts to drop due to overlapping confidence interval matching as described above, as the confidence grows

5. CONCLUSIONS

In this paper we have developed a preference elicitation method which caters for a large class of problems in preference modelling. In particular, we would like to elicit a preference ordering that can be used as the basis of a formal preference query to a database of items. However, on the one hand user's are not adept at making explicit their preferences and on the other, even those preferences that could be made explicit, may be difficult for the non-expert user to specify in a formal query language. The technique we developed here provides a formal solution to this problem while at the same time hiding the technical detail from the user. In our method, we rely on a complex model of the domain at hand; namely a domain ontology created by experts. We presented an interactive elicitation process which is dynamic and is able to elicit many preferences by covering large portions of the ontology. During this process, users are presented with examples which they are asked to rank and a system collects this information and builds a preference model on their behalf. The dynamics of the process relies mainly on the interaction between exploiting previously elicited preferences thus suggesting similar examples and exploring new preferences. We evaluated our methods through experiments run with several users in the domain of music and show significant results in terms of the preference model prediction accuracy as well as the coverage of different preferences elicited during this process. For future work, we intend to investigate the elicitation of more complex preference models, i.e., dependency structures over multiple attributes as well as a more advanced use of ontologies and semantic web features. Another avenue for future work would be in the area of collaborative filtering. We can aggregate the preference profiles of similar users to provide a more accurate 'collaborative' preference.

6. REFERENCES

[1] Anand, S.S., Kearney, P., Shapcott, M.: Generating semantically enriched user profiles for web personalization. ACM Trans. Inter. Tech. **7**(4) (October 2007)

[2] Rashid, A., Albert, I., Cosley, D., Lam, S., Mcnee, S., Konstan, J., Riedl, J.: Getting to know you: learning new user preferences in recommender systems. In: IUI, New York, NY, USA (2002) 127–134

[3] Bradley, K., Rafter, R., Smyth, B.: Case-based user profiling for content personalisation. LNCS **1892** (2000) 62–72

[4] Boutilier, C., Brafman, R.I., Hoos, H.H., Poole, D.: CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. JAIR **21** (2003)

[5] Boutilier, C.: A POMDP formulation of preference elicitation problems. In: AAAI. (2002) 239–246

[6] U. Chajewska, D.K., Parr, R.: Making rational decisions using adaptive utility elicitation. In: AAAI. (2000) 363–369

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

[8] Schickel-Zuber, V., Faltings, B.: Inferring User's Preferences using Ontologies. In: AAAI 2006. (2006) 1413–1418

[9] Ziegler, C.N., Lausen, G., Schmidt-Thieme, L.: Taxonomy-driven computation of product recommendations. In: CIKM '04. (2004) 406–415

[10] Kießling, W.: Foundations of preferences in database systems. In: VLDB. (2002) 311–322

[11] Siberski, W., Pan, J.Z., Thaden, U.: Querying the semantic web with preferences. In: ISWC. (2006) 612–624

[12] Chamiel, G., Pagnucco, M.: Exploiting ontological information for reasoning with preferences. In: Multidisciplinary Workshop on Advances in Preference Handling. (2008)

[13] Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: ACL. (1994) 133–138

[14] Chamiel, G., Pagnucco, M.: Exploiting ontological structure for complex preference assembly. In: Australian Joint Conference on Artificial Intelligence. (2008)

[15] Flake, G.W., Tarjan, R.E., Tsioutsiouliklis, K.: Graph clustering and minimum cut trees. Internet Mathematics **1**(4) (2004) 385–408