Kallol Naha

AUTOMATIC PREFERENCE LEARNING ON SEMANTIC ATTRIBUTES

Master on Computer Engineering: Computer Security and Intelligent Systems

Supervised by

Dr. Antonio Moreno

Department of

Computer Science and Mathematics

UNIVERSITAT ROVIRA I VIRGILI

September, 2016

Acknowledgements

This work has been supported by Universitat Rovira i Virgili. I would like to thank the advisor of this Master thesis, Prof. Dr. Antonio Moreno for his guidance and support during the elaboration of this work. Moreover, besides my advisor, I would like to thank Prof. Dr. Aïda Valls for her collaboration, as well as for creating an excellent work environment for me. Last but notleast, I would like to thank my family: my parent's late Sujit Kumar Naha and Joyonti Rani Kar for giving birth to me at the first place and supporting me spiritually throughout my life and in the stressful moments and my wife Nipa Rani Das for her continuous support.

Abstract

The huge volume of information and data available through Internet has led to an increase in the use of tools that can analyze them and support the users in their decision making processes. In particular, Recommender Systems are software systems that can take a large number of decisional options, analyze them according to the user's preferences and interests, filter those that are not appropriate and show to the user those that may fit better with his/her needs. In this kind of systems it is of paramount importance that the user preferences are as accurate as possible. These preferences may be either given directly by the user in an explicit way or may be learnt by the recommender system, by analyzing its interaction with the user. In this work we have focused on this kind of implicit learning approaches.

A common scenario in decision support systems is that each alternative is described with a set of attributes. When an alternative is evaluated, we must somehow find out how the value of each attribute fits with the user's preferences and put together the evaluation of each attribute to come up with the overall score of the alternative. In previous works in the ITAKA research group we had analyzed how we could learn the preferences of the user in an automatic way when numeric and categorical attributes were considered. In this work we have focused on a harder problem: preference learning on semantic attributes. Two scenarios have been considered, depending on whether each alternative has only one value or a list of values for each attribute. We present the learning algorithm and an analysis of its performance in a case study. The results are promising and open up the possibility to learn the semantic preferences in an automatic way.

Contents

List of figures

List of tables

Chapter 1
Introduction

The move from the World Wide Web (Web 1.0) to the Social Web (Web 2.0) has led to the appearance of the Knowledge Society, in which users have constant, direct, easy, cheap and flexible access to any kind of information about any topic. This availability of a huge quantity of information can be very helpful for users, but it can also be overwhelming for them. Thus, nowadays it is very necessary to have computational tools that may help users to acquire, analyze and manage these massive quantities of information. This fact is particularly important in our daily decision-making situations, in which we may have dozens or hundreds of options to choose from and it is not feasible for users to analyze all of them deeply to determine the one that fits better with their preferences.

One of the options provided in the last years from the Artificial Intelligence field is the use of Recommender Systems (RS) [Resnick & Valiant 97, Burke 07, Ricci et al. 11]. These systems should be able to analyze in an automatic fashion the characteristic of the options presented to the user (cars to buy, TV programmers to watch, movies to see, travel destinations to visit), taking into account the user's preferences, needs and context. This analysis should lead to a numerical or linguistic score of each option, which can be used to rank them, so that only the ones that seem to be more appropriate for the user are shown to him/her for the final decision. RS use a variety of recommendation techniques. The most common ones include content-based methods (alternatives are compared with a user profile, that stores his/her preferences), collaborative filtering (taking into account the ratings provided by all the users of the system) and demographic (making recommendations depending on the demographic characteristics of the user).

In this work we are focusing on content-based recommendation. The idea is that the RS has access to a user profile, which contains a representation of the user's preferences in the domain. Taking into account this knowledge, the system may compare the information about each alternative and decide the options that suit better the user's needs. Thus, the quality of the recommendations will depend heavily on how good is the knowledge of the RS about the interests of the user.

There are two basic ways in which the user profile may be built. One of them is that the user explicitly specifies his/her preferences. This option leads to a very accurate representation of the user's interests, but it requires an explicit effort by the user, which might involve the selection of some relevant values from long lists of possibilities. The second one, which is the one in which this work is going to focus, is that the RS learns automatically the user's preferences. This option does not require a conscious work of the user, but it can only be applied in situations in which the user is constantly faced with decisions, and the system may analyze the options selected by the user to infer his/her preferences. For example, it could be applied by a news recommender system, that analyses which are the news read by the user every morning and learns automatically the topics in which the user is (or not) interested.

In the field of Multi-Criteria Decision Making (MCDM) it is traditionally assumed that each alternative is described with a set of attributes (or criteria). Usually the input data of MCDM methods is an(alternative*attribute) matrix, in which each cell contains the value of an attribute for a given alternative. For example, if the alternatives are cars that could be bought by the user, the attributes could include the price of the car, the maximum speed, the fuel consumption, the amount of airbags, the luggage space, etc. The RS must analyze all this information to determine the car models that suit better the user's preferences. Of course, it is very different to recommend a car to a person that has to travel with three kids than to a person that travels alone and loves quick sports cars. If the RS knows the preferences of the user with respect to each attribute, it can analyze all the values of each alternative, give a score to each of them and aggregate somehow all of them to obtain an overall evaluation of the alternative. Having done it, it will be able to eliminate the alternatives with a low score, to rank all the alternatives or to show to the user only the best alternatives (relieving the user from the effort of manually analyzing hundreds or thousands of different options defined on dozens of alternatives).

In order for the recommendations to be accurate it is especially important that the knowledge of the user's preferences with respect to each criterion is as precise as possible. In our case, we are not being given the preferences by the user so we have to learn them automatically. In some previous works of the research group we studied how to learn automatically the preferences of the user with respect to numeric and categorical attributes [Marín et al. 13a, Marín et al. 13b, Marín et al. 14a]. In this work we have focused on the analysis of a new preference learning algorithm in the case of semantic attributes (attributes whose values are classes of a domain ontology, which is a structured representation of the main concepts and relationships in the domain).

Thus, the main objectives of the Master Thesis may be summarized as follows:

- Understand the general framework of implicit preference learning defined in the previous research work of the ITAKA research group.

- Define a new way to apply this preference learning framework in the case of semantic attributes. This new method should be applied both on uni-valued and multi-valued semantic attributes.

- Apply the semantic preference learning method in a case study and tune the values of its parameters to obtain the best performance.

The remainder of this document is structured as follows:

- Chapter 2 identifies the different kinds of attributes that may be used to describe a set of decisional alternatives, including the new semantic attributes. After that, it presents the general preference learning framework defined in previous works of the research group [Marín et al., 13a, Marín et al. 13b, Marín et al. 14a]. This chapter also explains how this preference learning framework was instantiated for the particular case of numerical and categorical attributes.

- Chapter 3 explains how the preference learning methodology has been adapted to the new case of dealing with semantic attributes. The idea is to take advantage of the structure of the underlying domain ontology to guide the learning process. This chapter distinguishes two cases, depending on whether the semantic attribute only has 1 value per alternative (uni-valued) or it has a list of values (multi-valued).

- Chapter 4 applies the novel semantic preference learning mechanism to a case study, with the objective of assessing whether it is indeed possible to tune the parameters of the algorithm in such a way that a good learning performance is obtained.

- Finally, the last chapter summarizes the work developed in this Master Thesis and comments briefly some possible lines of future work.

Chapter 2
A general MCDM framework for preference learning

This chapter introduces the appropriate background to understand the new contributions made in this work. First, we describe the basic types of attributes considered in a Multi-Criteria Decision Making setting, putting special emphasis on semantic attributes, which are the ones on which this work has focused. After that, we will introduce a general preference learning mechanism, developed in previous works of the ITAKA research group (Marín et al., 2013a, Marín et al. 2013b, Marín et al. 2014a). The idea of this general algorithm is that it is possible to learn automatically the user preferences if we consider settings in which the user is constantly faced with decisions (e.g. selecting TV programs every evening, reading news every morning, selecting the social networks updates to read every day, etc.). By analysing which are the alternatives selected by the user (and those that are discarded) it may be possible to detect patterns that lead to the knowledge of the user's preferences (e.g. if the user reads news about football every day, we can learn his/her interest on this topic; if the user never reads any news about tennis, we can infer that this topic is not interesting for him/her). Thus, over time the system should be able to learn automatically the user's positive and negative preferences and make up a complete and precise user profile with this information. We will explain in this chapter how this generic preference learning algorithm was already applied to learn preferences on categorical attributes [Marín et al., 2013b, Marín et al., 2014] and numerical attributes [Marín et al., 2013a]. The new contribution of this Master Thesis is the applications of this algorithm to the special case of semantic attributes, which have an underlying structure that, make them much more complex to manage than numeric or categorical criteria.

2.1 Definition of alternatives in MCDM systems

Multi-criteria Decision Making systems aim to support the user when he/she has to make a decision in a situation in which there is a large number of options or *alternatives* to choose from. For example, the user may choose among dozens of TV channels every evening, among dozens of possible travel destinations, among hundreds or thousands of social media updates to read, among millions of films or books to buy, etc.

Each option/alternative will be defined with respect to a set of *attributes*, often called *criteria*. These attributes constitute the information that is relevant for the decision maker about each alternative. The number of attributes may be very diverse, ranging from a few ones to dozens or hundreds. Usually the problem relies on the fact that there will not be any alternative which is the best for the user for all the attributes. The aim of recommender systems is, given a certain

alternative, to evaluate how good the value of each attribute is for the user; after that, all these assessments have somehow to be put together in order to obtain a global score for the alternative, that gives its overall evaluation. That means that the system must have a good knowledge of the user preferences about every possible value of every attribute.

In decisional problems it is possible to find different kinds of attributes, depending on the values they can take. The most common kinds of attributes are the following:

- *Numerical* (quantitative) attributes: they can take numbers as values (naturals, integers, reals). Usually the acceptable numbers are restricted to a particular range.
- *Categorical* (qualitative) attributes: they can take as values a fixed finite set of different categories. Normally these categories are not ordered in any way.
- *Boolean* attributes: they can only take the values true /false. They might be considered as a special case of categorical criteria with only 2 categories.
- *Semantic* attributes: they can take as values the concepts of the domain.

In a real problem it is common to find combinations of all these kinds of attributes. For example, if we want to decide our holiday destination, each of the alternatives would be a city, which could be defined with numerical attributes (maximum temperature, altitude and population), categorical attributes (main official language, kind of climate), boolean attributes (presence of an international airport) and semantic attributes (main kind of tourist attraction).

In this Master Thesis we have focused our attention only on one type of criteria: the *semantic* ones. These attributes may take as values the main concepts of the domain. For example, in the case of the semantic attribute that describes the main kind of tourist attraction in a city, possible values could be Cathedral, Art Museum, Football Stadium or Amusement Park. Given a semantic criterion, it will be assumed that the domain concepts related to it are described in a knowledge structure known as an ontology. An *ontology* [Gruber, 93] provides an structured representation of the main concepts of the domain and their main (taxonomic and non-taxonomic) relationships. The concepts of the domain are represented as classes of the ontology, the taxonomic relationships between them are represented with class-subclass relationships, and the non-taxonomic relationships are represented with binary properties. Figure 1 [Sánchez, 07] shows an example of an ontology about cancer, where the concepts (classes) are types and subtypes of cancer (colon cancer, breast cancer, liver cancer, etc.), which are non-taxonomic related with other concepts (for example Chemotherapy reduces Breast Cancer).

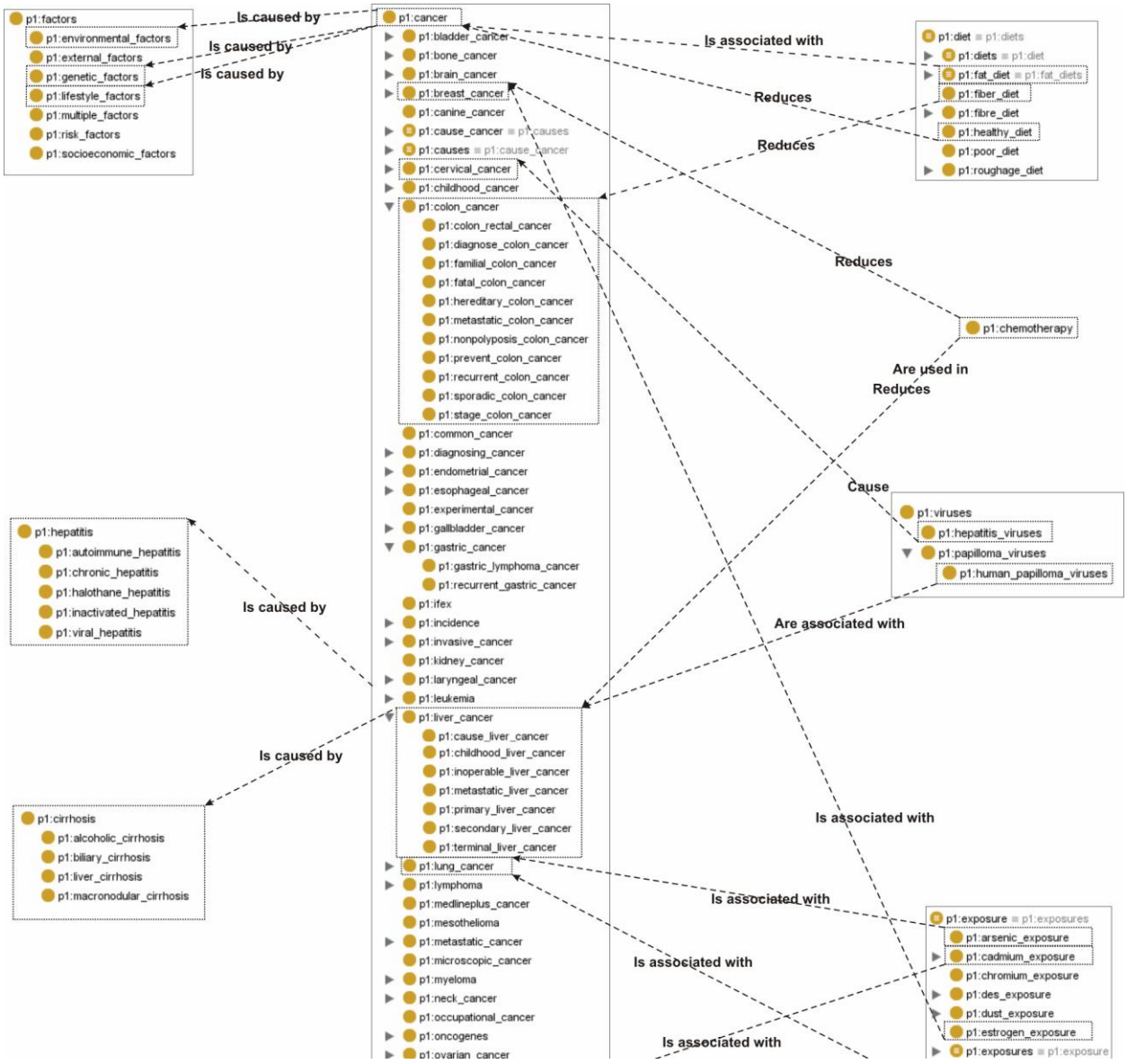

Figure 1: Example of a cancer ontology (Sánchez, 07).

Another example, more closely related to the case study that has been used in this work, is presented in Figure 2. This is a portion of an ontology that describes different kinds of tourist attractions. The full ontology was designed and developed at the Scientific and Technical Park of Tourism and Leisure (Vila-Seca, Tarragona), as part of the development of a personalised semantic recommender of touristic activities in the area of Costa Daurada and Terres de l'Ebre (Moreno et al.,13).

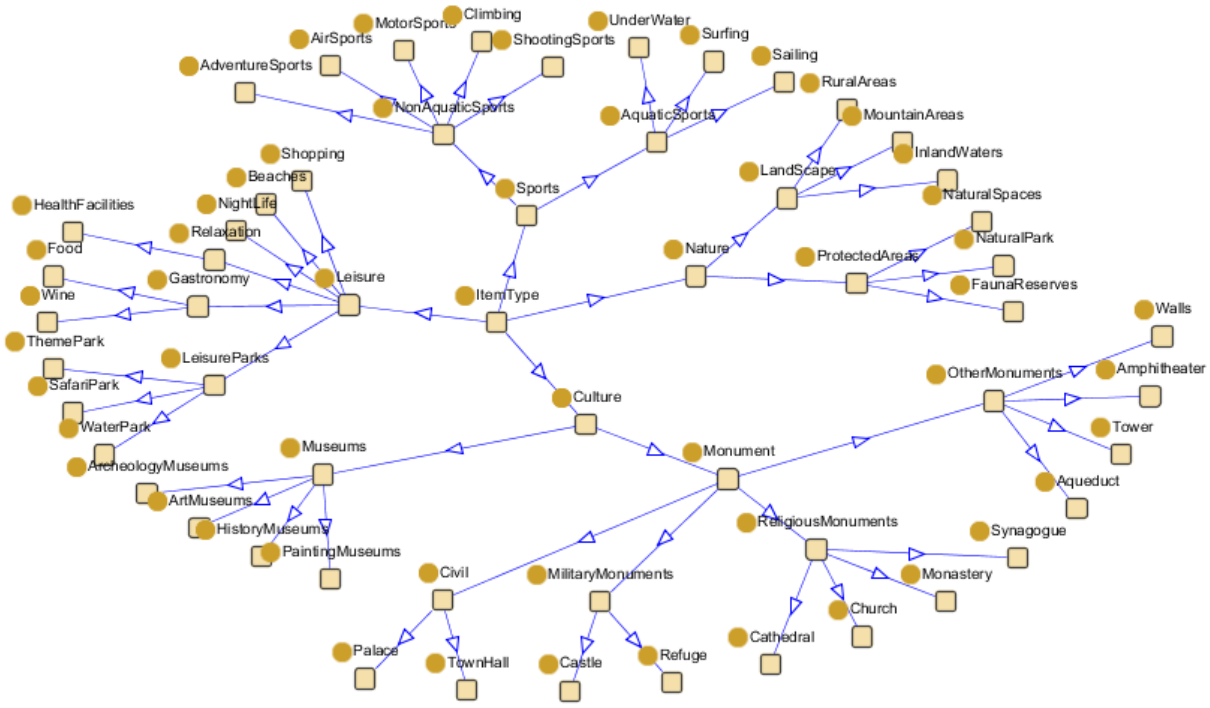

Figure 2: Portion of an ontology of tourist activities [Moreno et al., 13]

A portion of this ontology has been used in the case study described in chapter 4. We have only considered taxonomic relationships between concepts. In Figure 2 you can see that the main concepts include classes like Monuments, Museums, Leisure, Nature and Sports. These classes, on its turn, are decomposed into more specific classes (e.g. Military Monuments, Religious Monuments, Civil Monuments and Other Monuments). The hierarchical structure of classes and subclasses is analysed by the preference learning mechanism to discover the interests of the user, as will be described in the next chapter (e.g. if a user repeatedly selects holiday destinations that include Sailing and Surfing activities, the system could learn that he/she is interested in Aquatic Sports).

To finish the discussion on the attribute-based description of alternatives in decision-support systems, it is useful to remark the difference between uni-valued and multi-valued criteria. The former can only take a single value for each alternative, whereas the latter can take a list of values in a single alternative. In the case of semantic attributes it is more common to find multi-valued criteria than with numerical or categorical attributes. For example, a holiday destination could be described with a semantic attribute Tourist Attractions that could take as value a list of its main kinds of attractions. Thus, a single alternative (city) could have in this attribute a list of values that would be concepts of a domain ontology, like for instance [Cathedral, Castle, Amphitheater, Theme Park]. Both uni-valued and multi-valued attributes have been considered in this work, as will be shown in the next chapter.

2.2Implicit preference learning framework

The main aim of this work is to extend the implicit preference learning framework developed in the ITAKA research group in the last years so that it can manage semantic attributes (up to now, as described in the introductory chapter, the framework was limited to the case of numerical and categorical attributes). The preference learning method is fully implicit, because the user does not have to provide any explicit information about his/her preferences. However, it may only be used in domains in which the user is constantly confronted with decisions, and the system has access to the options selected by the user. The intuitive idea is that by analysing these chosen alternatives (and also the ones that are *not* chosen by the user) it might be possible to learn, after a modest number of interactions, which are the preferences of the user with respect to the criteria that describe the alternatives (i.e. the system should be able to learn, for each value of each attribute, whether that value is interesting or relevant to the user or not). The information about the preferences of the user, which is stored in the user profile, will be updated after each decision that he/she makes. Figure 3 shows the main modules of the preference learning framework [Marín et al. 13a].

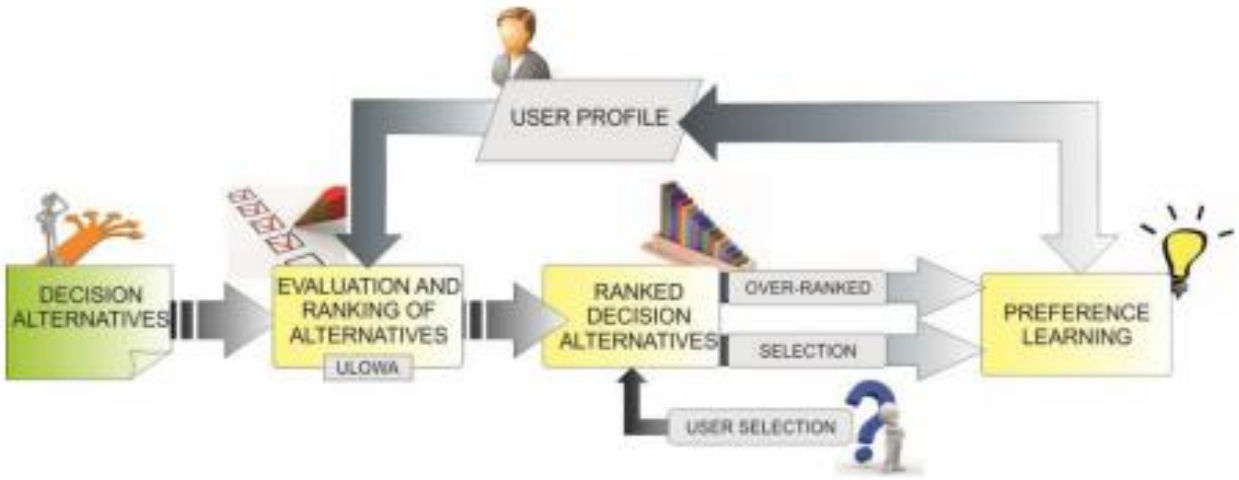

Figure 3: Implicit preference learning mechanism [Marín et al., 13a]

The preference learning algorithm shown in Figure 3 is iterative and progressive. It analyzes each of the decisions that are made by the user during a certain period of time, and it keeps updating and refining its knowledge about the user preferences after each iteration. Assume that we have some initial knowledge in the user profile about the user preferences on the values of each attribute. The actions made in each iteration (i.e. in each decisional situation of the user) are the following:

- The user is given a set of alternatives, described with a certain number of criteria.

- The system analyzes the attribute values of the alternatives, scores them and ranks them, taking into account the information about the user's preferences in the user profile. This step requires the separate analysis of each attribute and an aggregation of the assessments of each attribute into a single overall score. As shown in Figure 3, the aggregation method developed in our previous is called ULOWA (Unbalanced Linguistic Ordered Weighted Aggregation, [Marín et al., 14b]).

- The output of the previous step is a list of alternatives, ordered according to how well they fit with the user's preferences (the first alternative in the list is, in theory, the one that should be selected by the user). The user is shown this list of alternatives and it selects one of them. From this action we can now not only which is the option selected by the user, but also which are the options that were above this one in the list ordered by the system. These options are called over-ranked alternatives.

- The preference learning module receives three inputs: the current preferences of the user, the selected alternative and the set of over-ranked alternatives. Its mission is to analyse the values of the attributes in the selected and the over-ranked options and to update the information about the user's preferences accordingly.

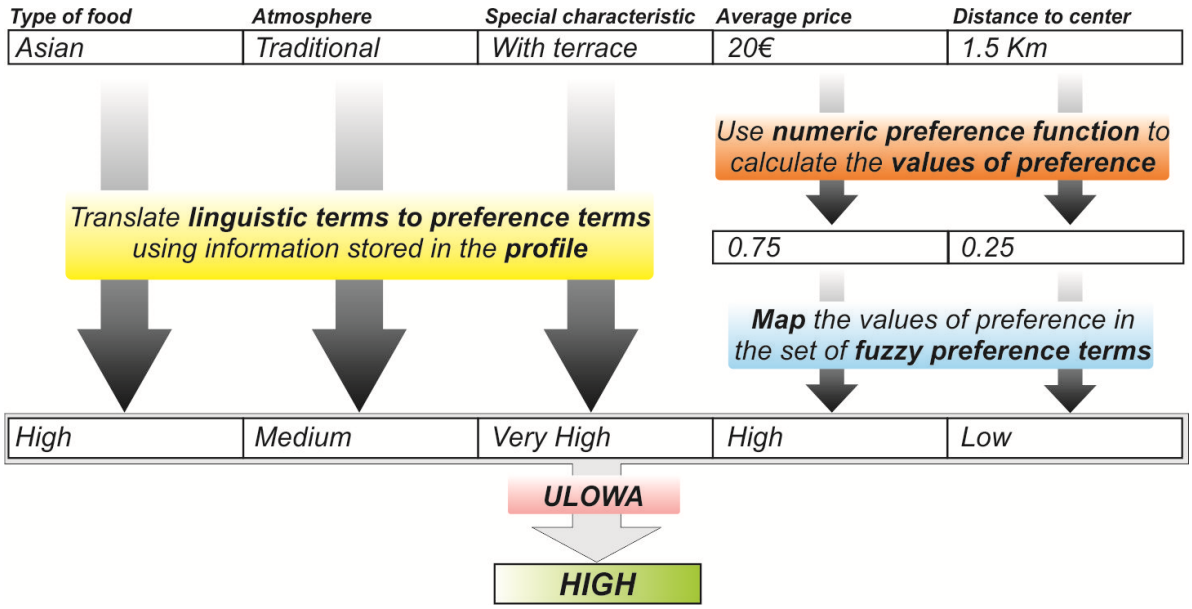

Figure 4: Evaluation of an alternative [Marín et al., 14a]

The two main underlying ideas about the preference learning algorithm are the following:

- The user is especially interested in the option he/she has chosen. Therefore, we should increase the preference degree of the values of the attributes of this option.

- According to the current preferences, the over-ranked alternatives seemed to be better for the user than the one he/she has finally selected. Therefore, this fact is showing that probably the information about the user's preferences in the user profile is not correct, and the system is over-rating values that the user does not like (and/or under-rating values that the user actually likes). Thus, the strategy followed by the preference learning algorithm is to decrease the preference degree associated to the values of the attributes in the over-ranked alternatives.

In summary, the main idea of the preference learning algorithm is the following. Let us assume that 10 options are considered in each iteration. Those 10 options are scored, ranked according to the current user preferences and shown to the user. If the user actually selects option number 5, that means that the options 1 to 4 are over-ranked (and probably option 5 was under-valued). Thus, the system will increase the preference on the attribute values of option 5 and decrease the preference on the attribute values of options 1 to 4. The hypothesis of the learning algorithm is that, by repeating this process a moderate but sufficiently high number of times, at the end the information about the user preferences in the user profile will be correct enough to guide a fully autonomous recommendation process.

Just for the sake of completeness, the following two subsections make a brief description of how this general algorithm was applied to learn preferences on numerical and categorical criteria on our previous ITAKA works [Marín et al. 13a, Marín et al. 13b, Marín et al. 14a]. Each type of attribute requires a different instantiation of the generic learning algorithm, and the adaptation to semantic attributes (described in the next chapter) is very different from the previous ones to numerical and categorical criteria. Another new contribution of this work is the consideration of both the uni-valued and the more complex multi-valued case, as will also be seen in the rest of this document.

2.2.1Preference learning on categorical attributes

The mechanism for preference learning in categorical attributes is described in detail in [Marín et al., 13b].The user profile adaptation is conducted by two processes. The first one—called *on-line* adaptation—is executed every time the user asks the system for a recommendation and evaluates the information that can be extracted from the current set of ranked alternatives. The second one—called *off-line* adaptation—is triggered after the recommender system has been

used a certain number of times. It considers the information given by the history of the previous rankings of alternatives and the selections made by the user in each case. In this subsection we will only provide a flavor of the on-line adaptation mechanism.

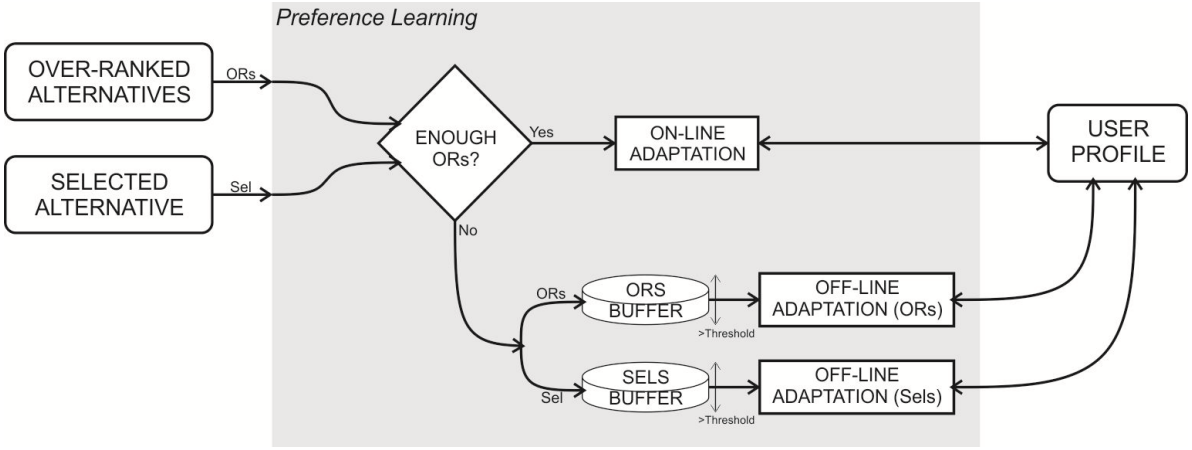

Figure 5: Preference learning on categorical attributes [Marín et al., 14a]

The adaptation processes modify the user profile by increasing or decreasing the level of preference of certain values of the criteria. The algorithm accumulates evidence to update all values, decides which values should be updated, and finally changes them by increasing/decreasing the preference score given the order of a predefined set of linguistic terms. For instance, a value with the preference *High* can be increased to *Very-High* or decreased to *Medium*.

The on-line profile adaptation process tries to keep the user profile updated by evaluating each of the user's selections, without taking into account the previous usage of the system. The main goals of this stage are to *decrease* the preference of the attribute values that are causing non-desired alternatives to be given high scores and to *increase* the preference of the attribute values that are important for the user but are not well judged on the basis of the current user profile.

For each recommendation made by the system, two items are evaluated: the selected alternative, which is the choice made by the user, and the alternatives that were ranked above it. Many conclusions can be derived from this information by extracting *characteristics* from the available data. A *characteristic* is a tuple <*String*, V_i, $\Re$> consisting of the name of one of the attributes, a value of the attribute and the number of times it is repeated among a concrete set of alternatives. A minimum number of alternatives are required, because if characteristics are extracted from a reduced number of alternatives they do not give useful information.

The features extracted from the set of alternatives that were ranked above the user's final selection are referred to as *over-ranked characteristics*, and they contain elements that were not

selected by the user. These characteristics are detected by observing the repetitions of the values in this set, but only when it has enough elements. Over-ranked characteristics are used to *decrease* the level of preference of the attribute values in the over-ranked set. The intensity of the decrease is regulated by the number of repetitions: if a characteristic is repeated many times, there is more reason to decrease the preference of the attribute value.

The features extracted from the user's final selection that do not appear in the set of over-ranked alternatives more than a given number of times are called *selection characteristics*. The adaptation process will only consider those selection characteristics that appear among the over ranked alternatives in a low percentage. In a similar procedure, selection characteristics are used to *increase* the level of preference of the attribute values indicated by the characteristics. The less the value appears among the over ranked characteristics, the greater the intensity of the increase.

2.2.2 Preference learning on numerical attributes

The way in which the general preference learning mechanism was applied to numerical attributes is explained in [Marín et al., 13a] and [Marín et al., 14a]. A brief summary is provided in this subsection.

The numeric adaptation of the user profile is inspired by Coulomb's Law: "*the magnitude of the electrostatics force of interaction between two point charges is directly proportional to the scalar multiplication of the magnitudes of charges and inversely proportional to the square of the distances between them*". The main idea is to consider the value stored in the profile (current preference) as a charge with the same polarity as the values of the same criterion on the over ranked alternatives, and with opposite polarity to the value of that criterion in the selected alternative. Thus, the value of the profile is pushed away by the values in the over ranked alternatives and pulled back by the value in the selected alternative. As in the case of categorical attributes, two stages were considered in the adaptation algorithm. The first one, called *on-line adaptation* process, is performed each time the user asks for a recommendation and there are enough over ranked alternatives. The other stage, called *off-line* process, is performed after a certain amount of interactions with the user.

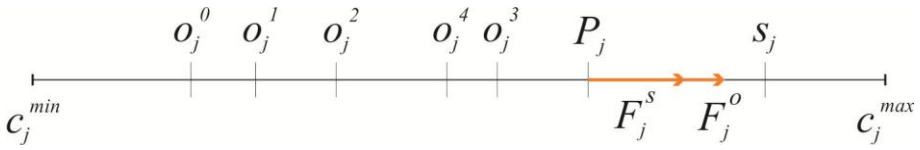

Figure 6: Attraction and repulsion forces [Marín et al., 14a]

For the on-line stage, the information available in each iteration is the user selection and the set of over ranked alternatives. In order to calculate the change of the value of preference in the user profile for each criterion it is necessary to study the attraction force done by the selected alternative and the repulsion forces done by the over ranked ones in each criterion, as represented in the example in Figure 6, in which the *j*-th value of the five over ranked alternatives o^0, o^1, o^2, o^3, and o^4 causes a repulsion force F^o_j, and the value for the same criterion of the selected alternative, s_j, causes an attraction *force* F^s_j. Both forces are applied on the *j*-th value of the profile, P_j.

The attraction force F^s done by the selected alternative for each attribute *j* is defined as

$$F^s(P,s,j,\alpha)=\begin{cases}\Delta_j\left(\dfrac{1}{\left|s_j-Pj\right|^{\alpha}}\cdot\dfrac{s_j-P_j}{\left|s_j-P_j\right|}\right) & if\, s_j\neq P_j\\ 0 & if\, s_j=P_j\end{cases} \qquad (1)$$

In this equation, Δ_j is the range of the criterion *j*, s_j is the value of the criterion *j* in the selected alternative and P_j is the value of the same criterion in the stored profile *P*. The parameter α adjusts the strength of the force in order to have a balanced adaptation process. The repulsion force exerted by the over ranked alternatives for each criterion *j* is defined as a generalization of Eq.(1) as follows:

$$F^o\left(P,\left\{o^1,...,o^{no}\right\},j,\alpha\right)=\sum_{i=1}^{no}\frac{1}{\left|P_j-o^i_j\right|^{\alpha}}\cdot\frac{P_j-o^i_j}{\left|P_j-o^i_j\right|} \qquad (2)$$

In this expression $o^i$is the *i*-th over-ranked alternative, o^i_j is the value of attribute *j* for o^i, and *no* is the number of over ranked alternatives. Finally, both forces are summed up and the resulting force is calculated.

The techniques designed for the on-line stage fail at detecting user trends over time since they only have information of a single selection. The *off-line adaptation* process gathers information from several user interactions. This technique allows considering changes in the profile that have a higher reliability than those proposed by the on-line adaptation process, because they are supported by a larger set of data.

The off-line adaptation process can be triggered in two ways: the first one evaluates the user choices, while the second one analyses the over ranked alternatives discarded by the user in several iterations. The possibility of running the off-line process (in any of its two possible forms) is checked after each recommendation. In the first case, the system has collected some alternatives selected by the user in several recommendation steps, and it calculates the attraction

forces (F'_s) exerted by each of the stored selected alternatives over the values stored in the profile, using an adaptation of Eq. (2), that has as inputs the profile P, the past selections $\{s^1,\ldots,s^{rs}\}$, the criterion to evaluate j, and the strength-adjusting parameter α:

$$F_s'\left(P,\left\{s^1,...,s^{rs}\right\},j,\alpha\right)=\sum_{i=1}^{rs}\frac{1}{\left|s_j^i-P_j\right|^{\alpha}}\cdot\frac{s_j^i-P_j}{\left|s_j^i-P_j\right|} \qquad (3)$$

The second kind of off-line adaptation process evaluates the set of over ranked alternatives that have been collected through several iterations and which were not used in the on-line adaptation process (because it did not have enough over ranked alternatives in a single iteration). When the stored over ranked alternatives reach a certain number, the off-line adaptation process calculates with Eq. (2) the repulsion forces over the profile values exerted by those alternatives (F^o).

2.3 Conclusion

This chapter has introduced the main types of attributes that are used to describe the available alternatives in a decisional process. This work has focused only on semantic attributes, which are those that can take as values the concepts of the domain. This chapter has also presented the general implicit preference learning framework developed in the last years in the ITAKA research group. It is an iterative process in which, in each step, the alternative selected by the user and the ones that had been ranked over it are analyzed in order to update the information about the user's preferences. We have provided a quick overview of how the preferences on categorical and numerical attributes are updated. In the following chapter we present the new contribution of this work, which is the adaptation of the preference learning mechanism to the case of semantic attributes.

Chapter 3
Preference learning on semantic attributes

This chapter describes how the general multi-criteria preference learning framework shown in the previous chapter has been adapted to deal with semantic attributes. First, the representation of preferences and the basic idea of using the information provided by the user in his/her continuous selections is summarised, and the need of using some ontology-based similarity measures is introduced. After that, it is explained how preferences can be learned when there is a single value for each alternative in each semantic criterion. Finally, it is shown how the learning algorithm has been changed to deal with the multi-valued case, in which an alternative may have a list of values in a semantic criterion.

3.1 Representation of preferences

An important decision to be taken in a recommender system is how to store the information about the user preferences. In the categorical case, there was a linguistic label (e.g. *Very High, High, Medium, Low, Very Low*) associated to each possible value of the attribute [Marín et al., 13b]. If the algorithm wanted to increase/decrease the preference on a particular value, it had to change the current label by the next/previous one. In the numerical case, there was a triangular preference function that assigned a value between 0 and 1 to each value in the domain of the attribute [Marín et al., 14a]. In order to increase/decrease the preferred value, the system could increase/decrease the position of the central point of the triangle.

In order to deal with preferences on the values of semantic attributes the situation is more complex, since the values that a semantic criterion may take have a certain taxonomical structure, and it is not obvious how to store the preferences associated to those values. In this work two important decisions concerning the representation of preferences in the user profile were taken:

- The system will only store preferences about the leaves of the ontological tree associated to each semantic attribute. Thus, it will know the interest of the user in the most specific concepts of the ontology. For example, if we consider the tourist ontology show in Figure 2, the system would store the preference on concepts like *Cathedral*, *Church*, *Monastery* and *Synagogue*, but it will not have any preference concerning its common father *Religious Monuments*.
- Unlike most preference representation mechanisms, which only store positive preferences, in our system we will store both positive and negative preferences. In that way the system will have more knowledge to guide the ranking of alternatives in each of the iterations of the learning process. More concretely, the system will store a numerical value between -1 and 1 for each specific concept of the ontology underlying each semantic attribute.

3.2 Preference learning framework

The preference learning framework, as commented in the previous chapter, is based on the fact that users are constantly confronted with decisions to make, and we can use the outcome of those decisions to guide the preference learning process, so that we can gradually discover the preferences of the user in a particular domain. In this iterative process, in each step the system scores and orders a list of alternatives, taking into account the current knowledge about the user's preferences. After that, the user makes a particular *selection*, in a certain position (*i*) of this list. Given this choice of the user, the system acquires both positive information (the values of the attributes of the selected option) and negative information (the values of the attributes of the options that had higher scores, i.e. the options that were located between the first and the *i-1*th position of the list, which are called *over-ranked*).

In a nut shell, the basic idea of the preference learning algorithm is the following:

- Increase the preference on the values that appear in the attributes of the selected alternative. We don't know exactly why the user has chosen a particular option, because an option is defined by all the values that it has in all the criteria defined in the domain, but it is safe to assume that he/she will like most, if not all, of the attribute values associated to that alternative.
- Decrease the preference on the values that appear in the attributes of the over-ranked alternatives. We will assume that, as the user chose an option that was less ranked than the over-ranked ones, that means that the values appearing in these alternatives have a preference that is too high and it should be lowered.
- It may be the case that an attribute value appears both in the selected alternative but also in some of the over-ranked ones. In this case, we have decided to give preference to the positive information of the user's actual selection; thus, we will only increase the preference on that value.

How will preferences be actually increased or decreased in the learning algorithm? In the case of the preferences on the values of the selected option, they will be increased by a given fixed factor, which will be called *X*. The value of this *increasing factor* will be a parameter of the learning algorithm. In the following chapter it will be studied how the performance of the algorithm changes with different values of this parameter. If *X* is very high, the system will change the preference values in the user profile very quickly, and it will tend to prefer those values that appear in the selected options, even if they do not appear very often. If *X* is lower, then the system will not have a high preference associated to a given value unless that value has appeared frequently in the user's selections; therefore, the learning process will be smoother and slower. When the system adds *X* to the current preference degree on a value it must also make sure that the new preference does not exceed 1.

In the case of the preferences on the values that appear only in the over-ranked alternatives, they have to be decreased. It is at this point where the learning system can take advantage of the information provided by the ontology associated to the semantic attribute in order to decide how much the preference on a particular value should be decreased. The presence of a taxonomical structure on the attribute values permits to define some kind of similarity measure between two values. There are many ontology-based measures available in the literature [Sánchez et al., 12]. In this work we have considered one of the simplest ones, which is the *path-length*: the distance *d* between two values is the length of the path that links those two values in the taxonomical tree.If this distance is considered, the following facts hold (the examples are taken from the tourist ontology shown in Figure 2):

- The distance between a concept and itself is 0 (*d*(*Cathedral*, *Cathedral*) = 0).

- The distance between a concept and its immediate father in the hierarchy is 1 (*d*(*Cathedral*, *Religious Monument*) = 1).

- The distance between two brother concepts is 2 (*d*(*Cathedral*, *Church*) = 2). Notice that this is the minimum similarity between two different leaves of the tree.

- If the height of the taxonomical tree is H, then the theoretical maximum distance between two concepts is 2*H. This maximum distance is obtained when the *Least Common Subsumer* of the two concepts (the most specific concept that is a super class of both concepts) is the root of the ontology. In the example shown in Figure 2 the root of the ontology is the node labeled *Item Type*, shown in the centre of the figure. Some branches below *Monument* have length 4, and all the other branches in the tree have length 3, so the maximum distance between two concepts is 7, e.g. *d*(*Cathedral*, *Sailing*) = 7.

- The distance between two concepts might be normalised to a value between 0 and 1 by dividing it by the maximum possible distance in the ontology. In the tourism example of Figure 2, distances could be normalised by dividing the path length by 7.

The intuitive idea of the preference learning algorithm is that the preference degree on the attributes values should be decreased. The magnitude of this decrease should depend on the distance between the attribute value that appears in the selected alternative (*v*) and the attribute value that appears in the over-ranked alternative (*w*). If the distance between them is high, then there should be a strong decrease in the preference of *w*, because the system is giving a high score to a value that is far away from the one that seems to be preferred by the user; however, if the distance between them is small, then the preference decrease should also be smaller. In particular, we have decided that if *v* and *w* are brother concepts (i.e. they have the same father), the preference will not be decreased. For example, if the user has chosen a *Cathedral*, and there was a *Church* in the over-ranked alternatives, the preference on *Church* will not be decreased; however, if there was a *Sailing* activity in the list of

over-ranked options, then its preference will be highly decreased (because *Sailing* is very different from visiting a *Cathedral*). Moreover, the system will also have a parameter, *Y*, which will modulate the amount of decrease to make in each of the iterations of the learning process. The following equation shows how the preferences of the attribute value *w* of an over-ranked alternative will be changed, depending on its distance to the attribute value *v* of the selected option:

$$\text{Pref}(w) = \text{Pref}(w) - (Y * (d(w,v) - 2)) \quad (1)$$

Note that, as commented above, if *v* and *w* are brothers the distance between them is 2, so the preference level of *w* would not be changed. When applying this formula the system has to make sure that the new preference is never below -1. In the next chapter we will study in a case study how the preference learning accuracy changes for different values of the *Y* parameter. If this value is high, the attribute values in the over-ranked alternatives will be heavily penalised; if it is low, the change in the preferences will be smoother, more gradual and slower. The relationship between the values of the two parameters, *X* and *Y*, will also be important. This relationship will determine the relative weigth between the increases in the values of the selected alternatives and the decreases in the values of the over-ranked alternatives.

3.3 Preference learning algorithm for uni-valued semantic attributes

We will start the analysis by considering the case in which each of the alternatives is defined by a single value in each semantic attribute (this was the usual situation in the categorical and numerical attributes studied in previous works). The user profile must be initialised in some way with information about the user's preferences with respect to each leaf of the ontology that represents the structure underlying the possible values of a semantic attribute. It is certainly not feasible to request the user to provide a preference value for each particular concept, since we might have hundreds of possible values for each semantic attribute. Some possible ways of initialising the user profile are the following:

- The preference of the user with respect to each concept could be initialised randomly, with a value between -1 and 1.
- We could assign a neutral preference value (0) to all the concepts.
- We could ask the user to give a general preference value for a small number of high-level concepts, and initialise all the leaves belonging to each of them with the same value. For example, the user could give a general preference between -1 and 1 to *Sports*, and we could assign this value to all specific instances of this concept. This could be a good initial approximation with a small effort from the user. This idea was employed for example in the SigTur [Moreno et al., 13] tourism recommender system, in which the user initially filled a small form with general preference information on different motivations for the travel, like *Nature*, *Shopping*, *Gastronomy*, etc. (see Figure 7).

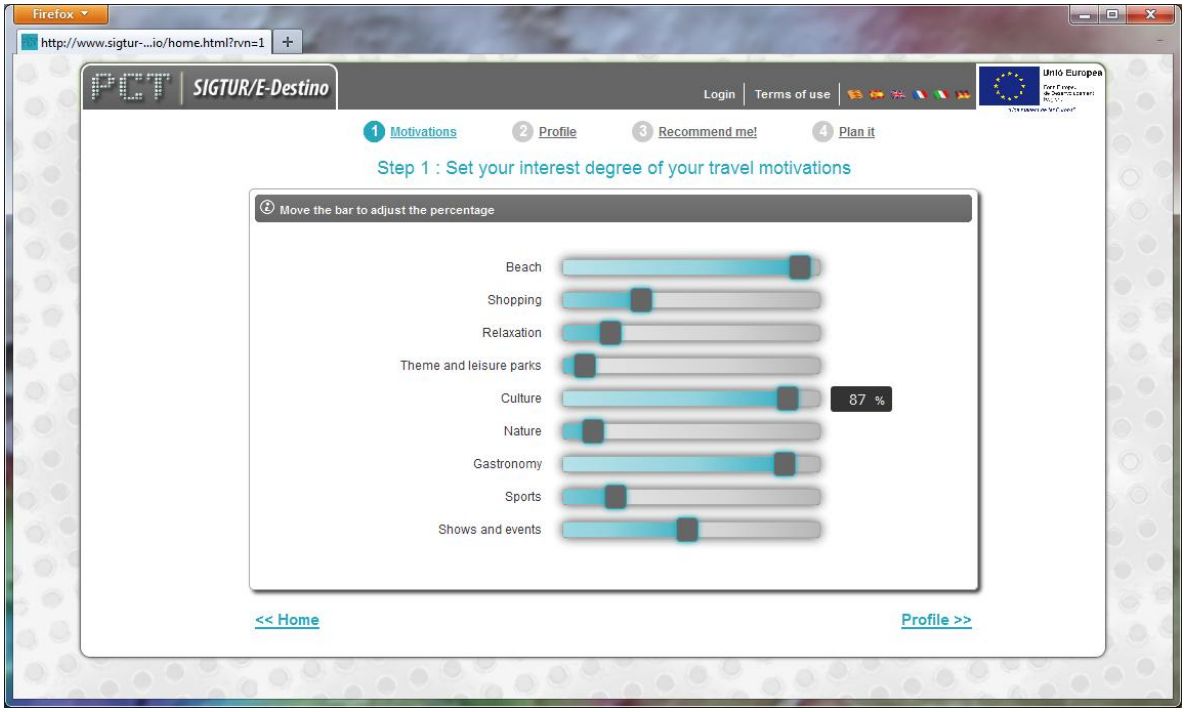

Figure 7: Expression of general travel motivations in SigTur [Moreno et al., 13]

In this work we have chosen the first option, so preference values will be randomly initialised. In the case study shown in the following chapter several runs of the preference learning system with different initialisations will be made and average results will be presented, so the effect of the random selection of initial preferences will be minimised.

The input elements of the semantic preference learning process are a set of alternatives, defined on some semantic attributes, and an ontology (taxonomy) for each of those attributes, which represents how the possible values of the attribute are structured. The general preference learning algorithm is the following:

1. ***Initialisation***. Initialise the user profile with a random preference [-1,1] for each specific concept of the ontologies.
2. ***Iterative Preference Learning***. Start an iterative process, in which the following steps are repeated in each iteration:
    a. ***Random selection of options.*** Select randomly a small number of alternatives (e.g. 10) and remove them from the set of alternatives.
    b. ***Ranking***. Give a score to each of these alternatives, according to the current information in the user profile, and rank them in a list *L*. The score of each alternative is the addition of the current preference of each of its attribute values.

c. ***User selection.*** The user selects his/her preferred alternative from the ordered list *L*. The position of this selected alternative will be called *p*. This selection defined two elements: the selected alternative *S*, and the list of over-ranked alternatives *OL* (the alternatives located from position 1 to position *p*-1 in the list *L*). Notice that the list *OL* may be empty, if the user selects the first option in *L*.
d. ***User profile adaptation.*** The information in the user profile is updated after the analysis of the attribute values in the alternative *S* and the over-ranked options in *OL*.

In the next subsections we will comment in more depth the two last steps of this learning process: user selection and user profile adaptation.

3.3.1 User selection

The learning process requires that the user makes a decision in each of the iterations, choosing the alternative that fits better with his/her preferences. In a real-world setting this would be an actual choice of the user (for example, if the system was learning the kinds of news that the user is interested in, the selection would be composed by those news the user actually reads). However, in our learning platform we want to make tests of the preference learning algorithm in an automatic way, without having a real human user making hundreds of selections; thus, we have had to *simulate* the actual selection process.

In order to make the simulation, we have added to the system the information of the real preferences of the user, which are stored in an *Ideal User Profile*. These are the preferences that we want to learn in an automatic way. In our platform they are used to simulate the real selection of a human user. The idea is that, in step 2c of the algorithm shown above, the system will score each of the alternatives using this Ideal User Profile (this scoring process will be commented in section 4.2 when the case study is described). The alternative with the highest score is the one that fits better the real user preferences that the system wants to learn; therefore, we can assume that the user would choose that alternative, so that is the selected alternative *S* in step 2c.

Notice that the real preferences in the Ideal User Profile are *not* used in any way in the profile adaptation process in the following step (in fact, they are indeed the preferences we aim to learn, so the system obviously ignores them). They are only used in the learning platform to simulate which would be the manual selections of a real human user with some specific preferences.

3.3.2 User profile adaptation

This step is the core of the preference learning system. At this point the system must analyse, following the general guidelines described in the previous section, the attribute values that appear in the selected alternative S and the attribute values of the (possibly empty) over-ranked list of alternatives OL.

The degree of preference on those values, stored in the user profile, is changed as follows:

- For those attribute values *w* that appear in *S*, their preference is updated as follows:

$$\text{Pref}(w) = \text{minimum}\ (1, \text{Pref}(w) + X) \qquad (2)$$

  X is a parameter of the learning algorithm that determines how fast the preference for a value is increased when this value appears in the selected alternative. In this formula it is made sure that the maximum preference for a concept is 1.

- For those attribute values *w* that appear in an alternative in *OL* and do *not* appear in the selection *S*, the following formula is applied in each appearance (*v* is the value of the same attribute in the selected option *S*, and d(*w*,*v*) is the path-length distance between *w* and *v* in the ontology associated to the attribute):

$$\text{Pref}(w) = \text{maximum}\ (-1, \text{Pref}(w) - (Y * (d(w,v) - 2))) \qquad (3)$$

  Y is another parameter of the preference learning algorithm, which basically represents how much an attribute value is penalised for appearing in an alternative that was not chosen by the user but was highly valued than it by the recommender system. The decrease in the preference level grows with the distance to the value in the selected option. The system also takes care that the preference on a concept is never below -1.

3.3.3 Illustrative example

Let us comment a brief example to illustrate the preference learning procedure. Table 1 shows five alternatives, O1 to O5, defined on three attributes, A1 to A3. The options have been ranked in step 2b in the order shown in the table (O1-O2-O3-O4-O5). Thus, according to the current preferences in the user profile, the best option for the user is O1. However, imagine that in step 2c the option selected by the user is actually O4. Thus, O4 is the selected alternative S, and the options above it (O1, O2 and O3) are the over-ranked ones in OL. The information in O5 will not be used by the preference learning algorithm.

	A1	A2	A3
O1	a	e	g
O2	c	e	l
O3	b	f	m
O4	c	f	j
O5	a	e	n

Table 1. Example of preference learning

The preferences in the user profile would be changed as follows:

- The degree of preference on the values *c* (A1), *f* (A2) and *j* (A3) would be increased in X, because these are the values of the attributes in the selected alternative.

- The degree of preference on the values *a* and *b* (A1), *e* (A2), and *g*, *l* and *m* (A3) would be decreased, because these are the values that appear in the over-ranked alternatives and do not appear in the selection. In each case the degree of decrease would depend on the distance to the attribute value in the selected alternative. For example, the decrease in preference for the value *a* would be Y*(d(a,c) -2); thus, if the length of the path between *a* and *c* in the ontology associated to attribute A1 were 5, the decrease would be 3*Y. If the concepts *a* and *c* were brothers (children of the same father), the distance between them would be 2, and the preference value on *a* would not be decreased. In the example shown in Table 1 the value *e* (A2) appears in two over-ranked alternatives, O1 and O2; therefore, its preference would be decreased twice.

3.4 Preference learning algorithm for multi-valued semantic attributes

The analysis is more complex when alternatives can have a list of values (and not just a single value) in each of the semantic attributes, because there is a higher uncertainty on why a user has selected a particular alternative or why a highly valued alternative has not been chosen by the user. This multi-valued case is much more common in semantic attributes than in numerical or categorical attributes. A previous study in the group already considered the case of multi-valued categorical attributes [Marín et al., 14a], although it did not apply any technique similar to the one developed in this thesis and described in this section. In the multi-valued case it is also going to be assumed that the user profile is initialised with random positive and negative preferences about the specific concepts of the ontologies associated to the semantic attributes.

The preference learning algorithm for the multi-valued case follows the same general idea than the uni-valued scenario described in the previous section. Thus, it is an iterative approach in which, in each of the iterations, the system shows a ranked list of alternatives to the user (sorted according to the current knowledge about the user's preferences) and the user selects one of them. This action of the user defines, implicitly, the set of over-ranked alternatives. The intuitive idea is that the system should increase the level of preference on the attribute values that appear in the selected option, and decrease the degree of preference with respect to the attribute values present in the over-ranked alternatives. Thus, the same general preference learning algorithm will be used, although some important changes have to be made to accommodate the use of a list of values for each attribute in each alternative.

The input elements of the semantic preference learning process are a set of alternatives, defined on some semantic attributes, and an ontology for each of those attributes, which represents how the possible values of the attribute are structured. The preference learning algorithm for the multi-valued case is the following:

1. ***Initialisation***. Initialise the user profile with a random preference [-1,1] for each specific concept of the ontologies.
2. ***Iterative Preference Learning***. Start an iterative process, in which the following steps are repeated in each iteration:
    a. ***Random selection of options.*** Select randomly a small number of alternatives (e.g. 10) and remove them from the set of alternatives.
    b. ***Ranking***. Give a score to each of these alternatives, according to the current information in the user profile, and rank them in a list *L*. Scoring an alternative in the uni-valued case was quite simple; however, the move to the multi-valued scenario requires some thought and making some decisions on how to do it. Some reflections on how to score an alternative defined on multi-valued attributes are given in subsection 3.4.1.
    c. ***User selection.*** The user selects his/her preferred alternative from the ordered list *L*. The position of this selected alternative will be called *p*. This selection defined two elements: the selected alternative *S*, and the list of over-ranked alternatives *OL* (the alternatives located from position 1 to position *p*-1 in the list *L*). As commented in the previous section, if the user selects the first option in *L* then the list *OL* is empty. Some comments on this step are also made in the following subsection, because it also requires scoring the options available in each of the iterations.
    d. ***User profile adaptation.*** The information in the user profile has to be updated after the analysis of the attribute values in the alternative *S* and in the over-ranked options in *OL*. In this step, described in more detail in section 3.4.2, it has to be considered that now there is a list of values for each attribute and alternative, so the adaptation process is more complex.

Thus, in the next subsections the main changes to be made in the different steps of the learning process (ranking of alternatives, user selection and user profile adaptation) when considering multi-valued semantic attributes are described.

3.4.1 Ranking of alternatives and user selection

There are two steps of the algorithm in which it is necessary to give a score to each of the alternatives. In the *Ranking* phase, the options have to be ordered according to the information in the user profile (the current knowledge of the system about the preferences of the user, which changes dynamically). In the *User Selection* step, as explained in the previous section, in order to simulate the manual selection of the user, the alternatives have to be ordered according to the Ideal User Profile (which is a fixed representation of the true preferences of the user).

Thus, in both cases it is necessary to assess the adequacy of an alternative with respect to a profile, which is an assignment of a value between -1 and -1 to each of the basic concepts (leaves) of the ontologies that provide a taxonomic structure to the values of the semantic criteria. Two things must be decided: how to give a score to the list of values assigned to a semantic attribute in a given option, and how to aggregate the scores of each attribute to obtain an overall score for the alternative.

To illustrate the multi-valued scenario faced at this point, let us consider a simple example. Imagine that objects are defined on 3 semantic attributes (*A1, A2, A3*). A specific object *o* has the values (*a, b, c*) in A1, (*d*) in A2 and (*e,f*) in A3. In the current user profile, the preference values assigned to the attribute values (*a, b, c, d, e, f*) are 0.8, -0.6, 0.7, -0.2, -0.7, -0.8, respectively. The system has to decide how good this alternative is for the user taking into account each of the attributes separately, and then somehow merge these three results to obtain *o*'s final score. In this example the evaluation with respect to *A2* should be slightly negative (-0.2) and the evaluation with respect to *A3* should be highly negative (*e* and *f* have very negative preference values, -0.7 and -0.8). However, the evaluation with respect to criterion *A1* is not very clear, since the user seems to like two of the three values of the attribute (*a* and *c*, with preferences 0.8 and 0.7) and dislike the other value (*b*, -0.6).

The basic options to consider when a multi-valued attribute is being evaluated with respect to a user profile are the following:

- *Add* the preference scores associated to each of the values (in this case, the value for *A1* would be 0.8 – 0.6 + 0.7 = 0.9). Note that with this option the final result is not bounded between -1 and 1. However, we can see if the result is positive or negative to have a general assessment of the adequacy of the list of attribute values to the user.

- Compute the *average* score of the preferences of all the values (in the example, the value for A1 would be 0.9/3 = 0.3). With this option the final value is normalised between -1 and 1, but we can't distinguish between a situation like the one shown in the example (0.8, -0.6 and 0.7) and the situation (0.3, 0.3, 0.3).

- Take the *maximum* preference of all the values (in the case of the example, the final preference for the attribute *A1* would be 0.8, the maximum of 0.8, -0.6 and 0.7). This option also preserves a normalised value between -1 and 1. It is an *optimistic* evaluation, in which the alternative is considered as good for a certain attribute if the user likes at least one of its values. For example, if we are evaluating a restaurant and the semantic attribute describes the types of food that it serves, we could think that it is correct to recommend the restaurant if the user likes very much one of those types of food (even if he/she dislikes the other kinds of food offered in the restaurant).

- Another (more conservative) option is to consider that an alternative should have a high score for a given attribute only if the user likes very much all the attribute values. This option may be implemented by taking the *minimum* preference of all the values (in the running example, the evaluation for A1 in this case would be -0.6, the minimum of 0.8, -0.6 and 0.7). Thus, the system would recommend a specific restaurant only if the user likes all the types of food that it serves.

- In general, it could be possible to apply any *Ordered Weighted Averaging (OWA)* operator [Yager, 88] to aggregate the preference values assigned to the list of values of an attribute. The three previous options are just specific instantiations of the OWA operator with different weighting vectors ((0.3, 0.3, 0.3) for the average, (1.0, 0.0, 0.0) for the maximum and (0.0, 0.0 and 1.0) for the minimum).

In this work we have performed some experiments with different options, and we decided to use the maximum preference, as will be shown in the case study in the next chapter.

The scores obtained for each attribute must also be somehow aggregated to obtain the final score of the recommendation alternative. In this step we decided to keep the same strategy than in the uni-valued case; therefore, the system simply adds the score obtained for each attribute. Other more complex aggregation choices, as the ones discussed in the case of the evaluation of a single alternative, could certainly have been considered.

3.4.2 User profile adaptation

In this step, which is made in each of the iterations, the system needs to update the information that it has about the user's preferences, taking into account the current preferences, the option *S*chosen by the user and the (possibly empty) list *OL* of over-ranked alternatives. The intuitive idea is the same than the one in the uni-valued case: the system should increase the preference on the attribute values that appear in the selected alternative, and decrease the preference on those that appear in the over-ranked alternatives. The scenario is now more complex because each alternative has a list of values for each criterion, instead of a single-value.

The changes to be made in the current preferences of the user are the following:

- For those attribute values *w* that appear in *S*, their preference is updated as follows:

$$\text{Pref}(w) = \text{minimum}\,(1, \text{Pref}(w) + X) \qquad (4)$$

  In the case of the values that appear in the attributes of the selected option S, the update is the same than in the uni-valued case (a fixed increase X, bounded by +1, is applied, as was shown in eq. 2). As the system does not have any kind of information about why the user

has chosen that option, it assumes that the user must probably like all (or most of) its attribute values.

- For those attribute values that appear in an alternative in *OL* and do *not* appear in the selection *S*, the preference updating formula is more complex. Given a certain attribute *A*, if *w* is one of the attribute values in a member of the *OL* list, $L=(l_1, l_2, \ldots, l_k)$ is the list of values of the attribute *A* for the selected option *S* and *d* is the path-length distance between two values in the ontology associated to the attribute *A*, then the preference on *w* is modified with the expression

$$\text{Pref}(w) = \text{maximum}\,(-1, \text{Pref}(w) - (Y * (d(w,l_j) - 2))) ,$$
$$\text{Where } l_j \text{ is the element of L that minimises the distance } d \text{ with w} \qquad (5)$$

The intuitive idea is that the preference on an attribute value that appears in the over-ranked alternatives will be decreased heavily only if it is very different from all the attribute values in the selected alternative. Thus, the system is following a very cautious approach when decreasing preferences. If the value *w* is similar to one of the attribute values in *S*, then the preference decrease will be small.

3.4.3 Illustrative example

Let us see in a small example how the preferences would be changed in the multi-valued scenario. Table 1 shows four options, O1 to O4, defined on a single semantic attribute A. O4 is the option chosen by the user (*S*), so the over-ranked list *OL* is (O1, O2, O3).

	A
O1	(a,b,c)
O2	(b,g)
O3	(e,f)
O4	(a,f)

Table 2. Example of preference learning in the multi-valued case

The preferences in the user profile would be changed as follows:

- The degree of preference on the values *a* and *f* would be increased in X, because these are the values of the attribute in the selected alternative.

- The degree of preference on the values *b* and *c* (object O1), *g* (object O2), and *e* (object O3) would be decreased, because these are the values that appear in the over-ranked alternatives and do not appear in the selection. Note that the preference on *b* would actually be decreased twice, because it appears in two of the over-ranked alternatives (O1 and O2).

  For example, to calculate how much the preference on *g* must be decreased, the system has to calculate the minimum distance between *g* and all the attribute values in S (*a* and*f*). If *d*(*g*,*a*)=5 and *d*(*g*,*f*)=8, then the minimum would be 5 and the preference on *g* would be changed, applying equation (5), with the expression

$$\text{Pref}(g) = \text{maximum}\ (-1, \text{Pref}(g) - (Y * (5 - 2)))$$

  Thus, the preference on *g* would be decreased by 3 times Y (always checking that it does not go below the minimum preference -1).

3.5 Conclusions

This chapter has described the core contribution of this work, which is the adaptation of the general implicit preference learning algorithm (initially defined only on numerical and categorical attributes) to the case of dealing with *semantic* attributes, defined as those that can take as values the most specific concepts of a domain ontology. The intuitive idea is still to have an iterative selection procedure in which, in each step, we increase the preference on the values of the selected option and we decrease the preference on the values of the over-ranked alternatives. In the case of this work, the structure of the ontology is basically used to decide how much the preference on an attribute value of the over-ranked alternatives should be decreased, depending on its semantic distance to the attribute value(s) in the selected option. Moreover, two different scenarios have been considered, depending on whether the semantic attributes can take only one value or if they can be multi-valued. The next chapter will describe the application of this general preference learning algorithm to a specific case study.

Chapter 4
Case study: tourist destinations

In this chapter the new preference learning algorithm on semantic attributes is applied to a particular scenario to test its performance under different circumstances. In particular, we wanted to check how successful was the algorithm for different values of its parameters X and Y. In this small case study the alternatives to be considered are tourist destinations, defined on some semantic attributes. A particular ideal profile to be learnt is defined, and it is analysed if the algorithm may learn these ideal preferences after analysing some selections of the user (which is simulated with the help of the ideal profile, as described in the previous section).

In the following subsection the specific data used in this scenario is described. After that, we apply the learning algorithm in the two cases considered in this work: uni-valued and multi-valued semantic attributes.

4.1 Data

In a previous work in the ITAKA research group some new ontology-based mechanisms for Information Extraction were defined [Vicient et al., 13]. One of the domains in which they were applied was Tourism. In particular, the main features of 150 top tourist destinations were identified in an automatic way from their Wikipedia pages. This initial work contained different kinds of attributes; however, in this case study we will consider only 3 semantic attributes: sports, buildings, landmarks. These attributes contain information about the main sports that can be seen in the city, the main kinds of buildings and the most relevant landmarks. The ontology-based information extraction procedure identified a possible list of values (features) associated to each attribute; therefore, all of them were multi-valued.

Destination	Building	Landmark	Sport
Mecca	Mosque	Abraj Al-Bait	Football
Prague	Public_University	River	Football
Moscow	Monastery	Urban_Park	Tennis
Beijing	Cathedral	Botanical_Garden	Basketball
Vienna	House	Zoo	Ice_Hockey
Taipei	Public_University	River	Football
Saint_Petersburg	House	Canal	Swimming
Macau	Cathedral	Hill	Formula_One
Venice	House	River	Basketball
Warsaw	Business_School	Zoo	Golf

Table 3. Example of destination data for the uni-valued case

For the case study we have used two simplified versions of these data: a uni-valued one (in which one of the attribute values was randomly chosen for each attribute) and a multi-valued one (in which each city had between 1 and 5 values for each attribute). A small excerpt of the uni-valued version is shown in Table 3.

The ontology used to automatically extract these values from Wikipedia was developed in the *SigTur* project, in which a personalised recommender of touristic activities was developed [Moreno et al., 13]. The sections of the ontology that describe buildings, sports and landmarks are shown in the annex of this document. For example, it may be seen that the class Building is divided in several subclasses, covering different kinds of buildings (industrial, commercial, cultural, religious, etc.). The values of the attributes correspond to the most specific classes of these ontologies (the leaves of the hierarchical tree structure).

In all the tests performed in this case study the aim was to learn the preferences associated to a particular individual. These preferences were represented in the *Ideal User Profile*, described in section 3.3.1. In this case study it was assumed that the user preferred the attribute values associated to the subclasses *Religious Buildings*, *Water Landmarks* and *Aquatic Sports*. Thus, the aim of the preference learning mechanism is to learn a preference +1 on these values and a preference -1 on the rest of the values. The initial user profile is built by putting a random value between -1 and 1 for each basic concept of the ontology.

4.2 Preference learning results (uni-valued case)

The preference learning framework works in an iterative fashion, as described in the previous chapter. In the tests we will consider 100 iterations. The main steps taken in each of the iterations are the following:

- Choose randomly 10 of the 150 destinations.
- Score and rank the 10 destinations considering the current user profile (list *L*). The score of each destination is the addition of the preference value of each of the attribute values. For instance, the value of Macau (see Table 3) would be the addition of the preferences on *Cathedral*, *Hill* and *Formula_One*.
- Score and rank the 10 destinations using the Ideal User Profile (list *P*). The system has to check if the attribute values of the destination are semantically similar to the preferred classes in the Ideal User Profile. If *v* is the attribute value (one of the most specific classes of the ontology *o*) and *c* is the preferred class for that attribute, then the score for that attribute will be calculated as

$$\text{score}(v) = 1 - (d(v,c) / (2 * \text{height}(o))) \qquad (6)$$

In this expression *d* is the path-length distance described in the previous chapter, which is normalized by dividing it by its maximum possible value. If the value *v* is one of the subclasses of the preferred class *c*, then a score of 1 is automatically given. The overall score of the city is the addition of the scores of the three attribute values.

For example, the score for the value *Formula_One* of the attribute *Sports* in Macau (Table 3) would be 1 - (*d*(*Formula_One*, *Aquatic_Sports*) / (2*height(Sports Ontology))) = 1 - 5/8 = 3/8. The score for the value *Cathedral* of the attribute *Building* of the same city would be 1, because *Cathedral* is a subclass of the preferred class *Religious_Building*.

- The first city in list *P* is the selected option *S*.
- The system checks the position of *S* in list *L* (position *p*). The sub list of *L* from position 1 to position *p*-1 is the list of over-ranked alternatives, *OL* (if *p*=1, then this list is empty).
- The current user profile is modified according to the attribute values in the selected option *S* and the over-ranked alternatives in *OL*, as described in section 3.3 (*User Profile Adaptation*). Recall that this adaptation depends on the value of two parameters, *X* and *Y*. *X* controls how much the preference of a value increases when it belongs to the selected alternative *S*. *Y* modulates the decrease of the preferences of the values appearing in the over-ranked alternatives.

In all the tests shown in this chapter we have made 10 different runs of the system, starting with random initial profiles. The results are the average of the 10 executions.

There are two aspects that have been measured in the performed tests to evaluate the performance of the preference learning algorithm:

- In order to check if we are learning the ideal user preferences, we have defined a *profile distance* between the dynamic current user profile *CP* and the fixed Ideal User Profile *IUP* as follows:

  ProfileDistance(*CP*,*IUP*) =
  (addition, for all specific concepts *c* of the ontology,
  abs(preference(*c*,*CP*) – preference(*c*,*IUP*))) /
  (2*number of specific concepts in the ontology) (7)

  Thus, this distance is the average value of the preference differences between the current profile and the ideal profile for all the possible values of the attributes. In the ideal user profile the preference is +1 for all the values that belong to the preferred classes (*Religious_Buildings*, *Water_Landmarks* and *Aquatic_Sports*, in this case study) and -1 for all the other values (e.g. all the buildings that are not religious). Note that this distance

ranges from 0 (if the two profiles coincide) to 1 (if the two profiles only contain +1 and -1 values and they do not coincide in any case). If the preference learning algorithm works well, this difference should keep decreasing and move towards 0.

- We also check the evolution of the *position of the selected alternatives* in each of the iterations. If preferences are learnt correctly, the system should give a very high score to the best alternative, so it should appear in one of the initial positions of the list *L* (in step 2b of the algorithm shown in section 2.2). In other words, the length of the list of over-ranked alternatives should keep decreasing. If the user ideal preferences were perfectly learnt, the best candidate would always have the higher score and the list of over-ranked alternatives would be empty.

We start by checking the influence of the parameter X in the algorithm (the preference increase on the preferences of the attribute values of the alternative selected in each iteration). Y is fixed to 0.075 in these tests. The following figure (average of 10 runs starting with random initial profiles) shows how the distance between the current user profile and the ideal user profile changes depending on the value of the X parameter:

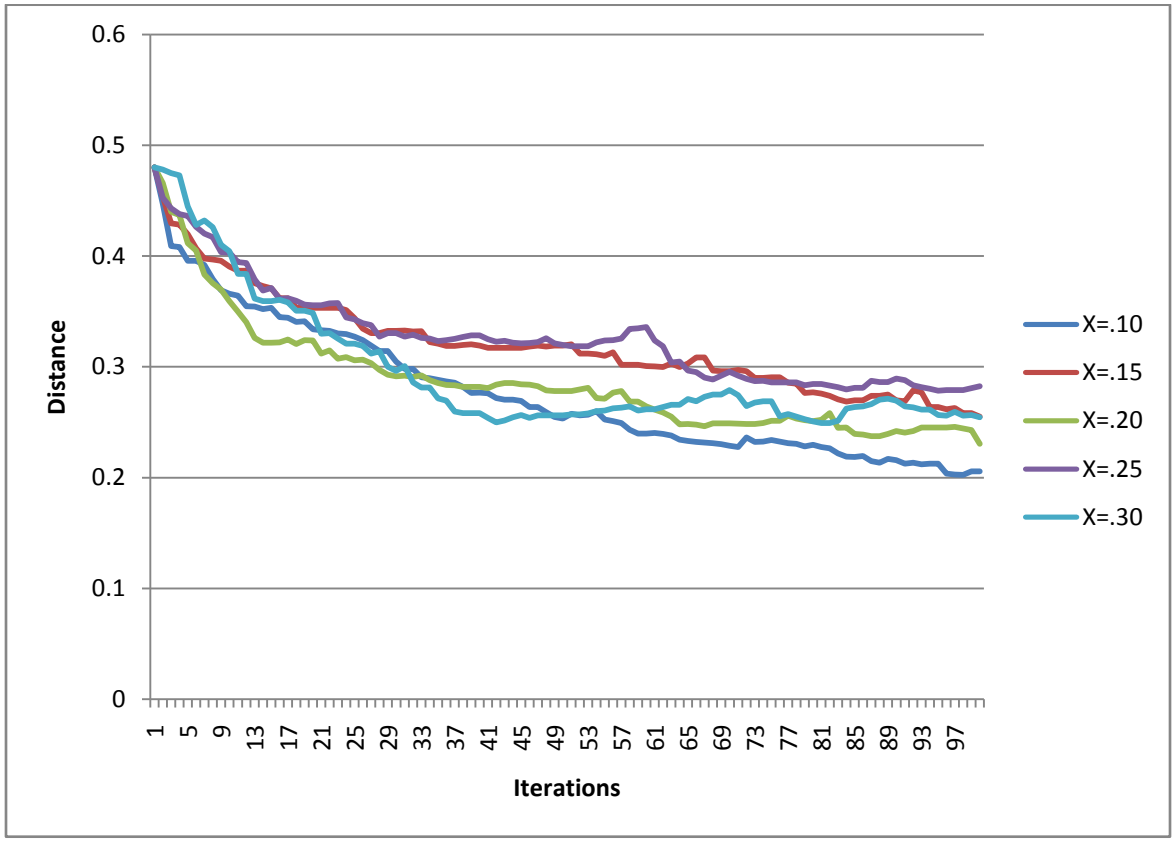

Figure 8: Influence of the X parameter of the learning algorithm

In Figure 8 it may be seen that in all the cases the distance to the ideal profile keeps decreasing from an initial value near 0.5, reaching in all cases a value lower than 0.3 after 60 iterations. This fact shows that the learning algorithm is indeed working and preferences are tending towards the ideal ones that we want to learn. In this test it is shown how the best values for the parameter X are 0.1 and 0.2. It may be seen that, for X=0.10, the profile distance reduces quite quickly to 0.3 (in around 30 iterations), then to 0.25 (in 30 more iterations) and finally to 0.2 (by the 100th iteration). Thus, the biggest learning is produced in the first iterations.

We can fix the value of X and analyze the influence of the Y parameter (the degree of reduction of the preference on the values that appear in the over-ranked alternatives). The results obtained (for X=0.2) are shown in the following figure:

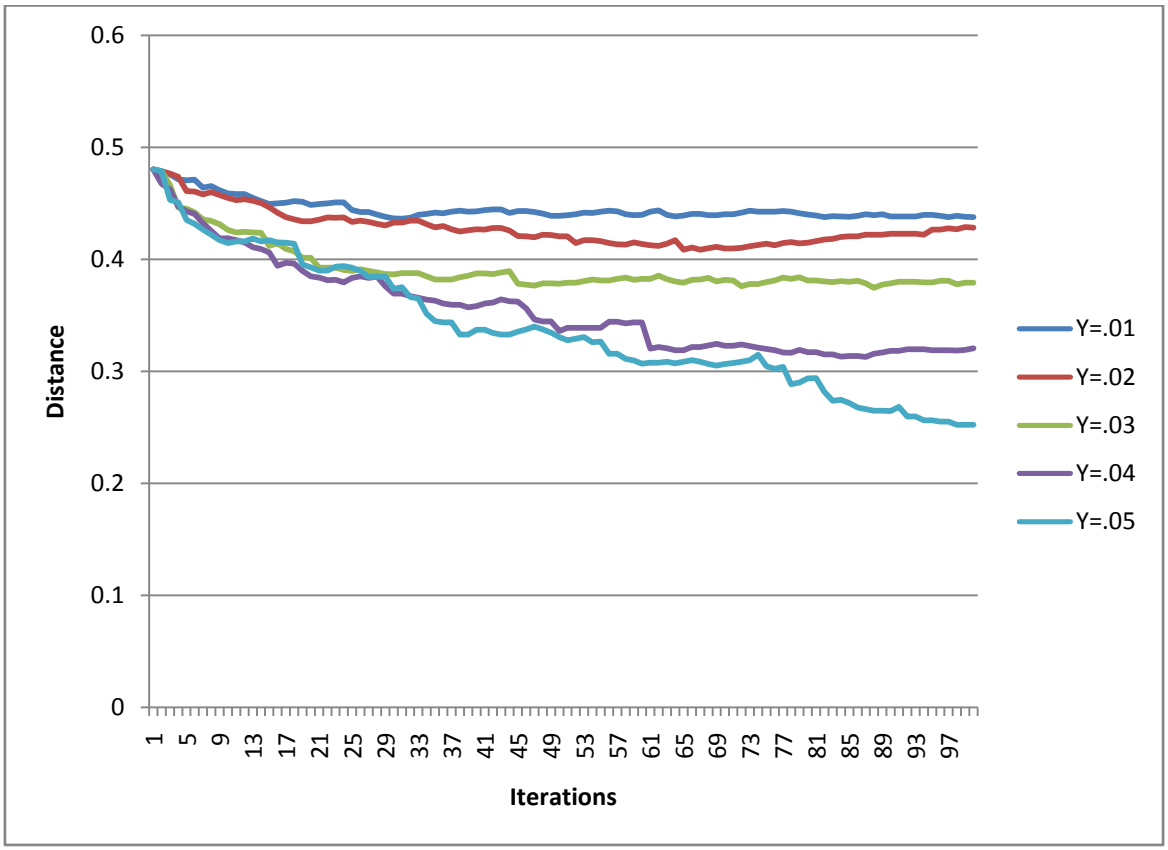

Figure 9: Influence of the Y parameter of the learning algorithm

In this test the highest reduction of the distance between the current profile and the ideal one is obtained when Y is 0.05, reaching a final distance around 0.25. It may be seen that the distance reduction with small values such as .01 or 0.02 is very low.

These tests show that the values of X and Y do indeed have an important influence on the learning process. If these values are not set appropriately, the learning process may not work correctly at all. For example, Figure 10 shows the result obtained if X=0.6 and Y=0.001 (the system increases very quickly the preferences on the values of the selected alternatives, and penalizes very slightly the values that appear in the over-ranked alternatives).

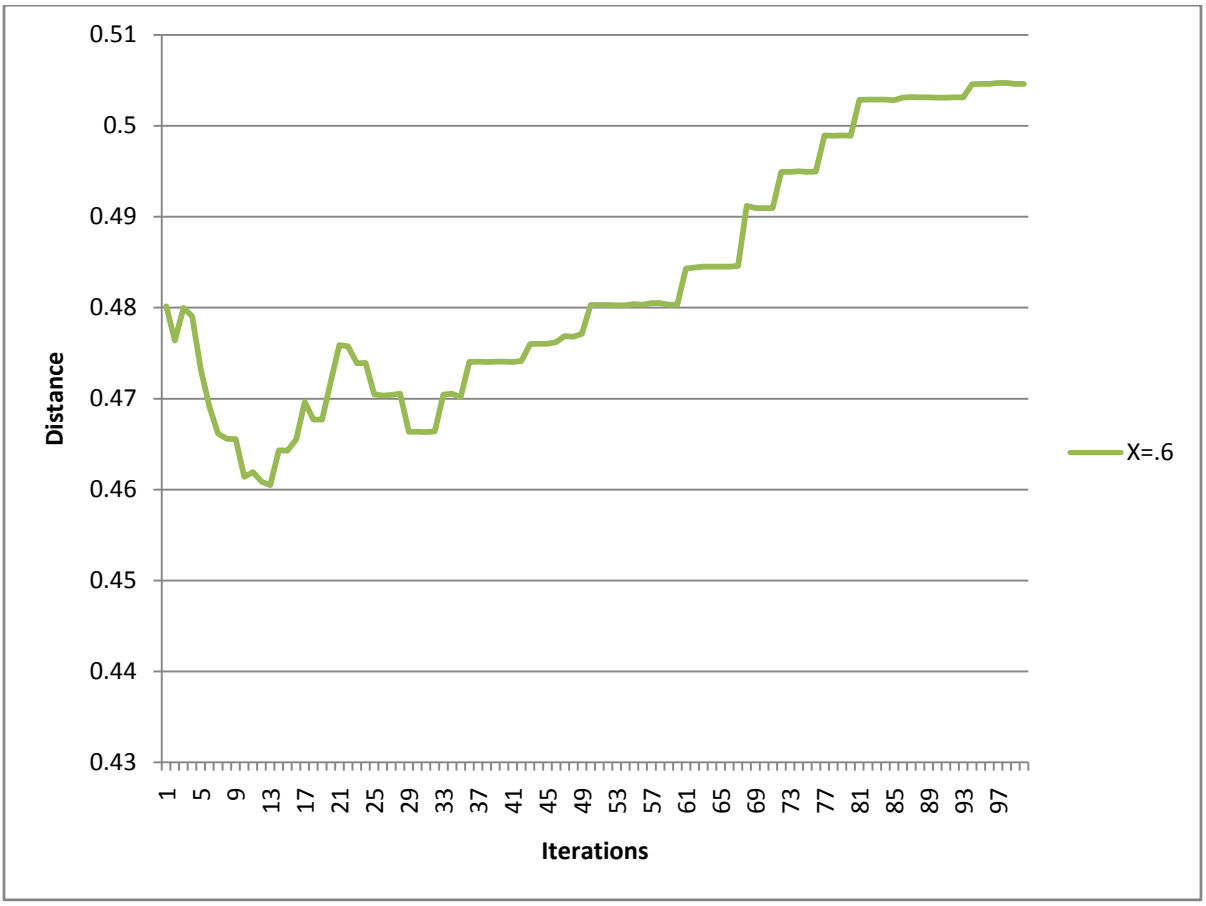

Figure 10: Example of a bad choice of parameter values

In this example the distance to the ideal profile decreases very slightly in the first iterations, but then it starts to increase and, after the 100 iterations, we have a profile that is actually worse than the random initial one. Thus, the system is not able to learn the preferences of the user with this setting of the parameter values.

The previous analysis has shown that, for a fixed value of X or Y, the value of the other parameter has a strong influence on the learning performance. In order to find the best combination for these values, a grid exhaustive search was performed, with X values in the interval [0.10, 0.40] and Y values in the interval [0.01, 0.12]. The best results, shown in Figure 11, where obtained with X=0.20 and 0.075. It may be seen that the distance between the current profile and the ideal profile decreases steadily down to a 0.23 value.

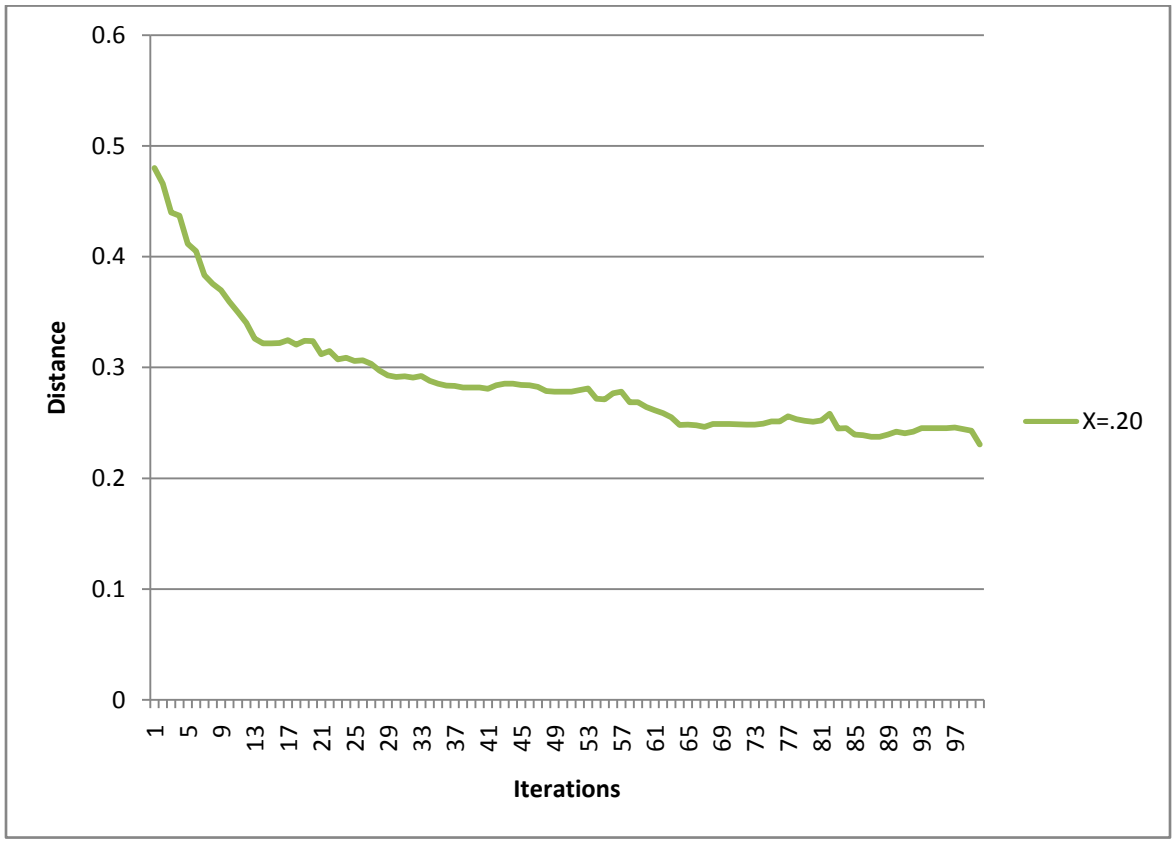

Figure 11: Best learning performance (X=0.2, Y=0.075)

It is also interesting to explore the evolution of the position of the selected alternative during the 100 iterations. The following figure shows this position in a test with X=0.2 and Y=0.075, one of the best learning configurations for this case study.

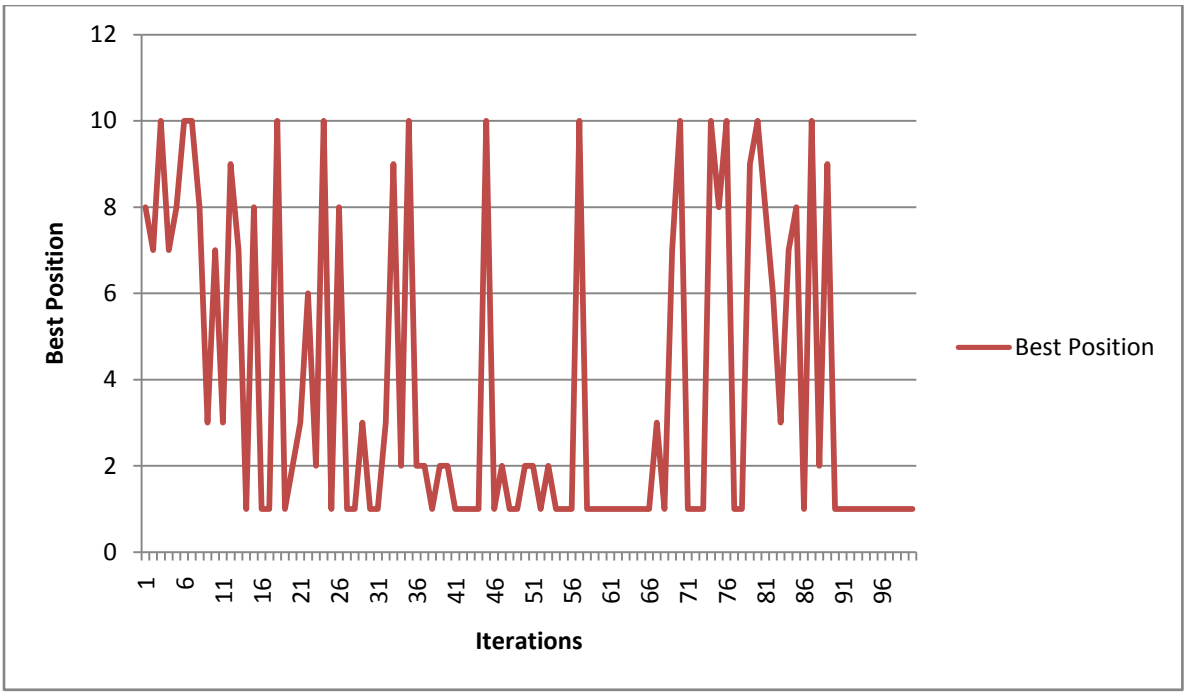

Figure 12: Evolution of the position of the selected alternative

The system considers 10 cities in each iteration. Therefore, the position of the selected alternative ranges from 1 to 10. In this test it may be seen that, in the initial 10-15 iterations, the selected option is in high positions (indicating that the system is not correctly ordering the alternatives, so the current profile is quite different from the ideal profile to be learnt). After that, there is a period (iterations 15-35) in which sometimes the position is very good (the first one) but sometimes it is still bad (8^{th}, 9^{th}, 10^{th} positions). However, from iteration 35 to iteration 70 in almost all cases the selected option is the first or the second one, showing that the system seems to have learnt the correct preferences. In these iterations the system only reinforces the preferences on the attribute values of the selected option, but it does not have many over-ranked alternatives in which a reduction of the preference values may be applied. Therefore, it is possible that in this phase some attribute values are being over-rated. Then, from iteration 70 to iteration 90 there are 7 cases in which the selected option is in a low position of the ranked list computed by the system. Thus, in these iterations there are many over-ranked alternatives and the preferences on many attribute values will be decreased again. Finally, in the last 10 iterations the selected option is again the first one. In summary, it may be seen that, in the last 50 iterations, the selected alternative is in the top position 31 times (62%); thus, in most of the iterations the system is able to sort the 10 alternatives in such a way that the best one for the user is shown in the first place.

4.3 Preference learning results (multi-valued case)

In the multi-valued case each of the destinations has a list of 1-5 values for each of the three semantic attributes that have been considered. An example of the values of some cities is given in the following table.

Dest.	Building	Landmark	Sport
Budapest	Opera	River	Football,Ballet, Formula_One,Diving
S.Francisco	Public_University	Refuge,Canal	Ballet
Orlando	Public_University	Canal,Zoo,River	Roller_Hockey,Tennis, Formula_One
Miami	Fort,Temple	Zoo	Ballet,Karate,Tennis, Formula_One
Munich	Prison,Fort,Church,Mall	River,Canal,Zoo, Urban_Park	Ice_Hockey,Ballet, Formula_One
Shenzhen	Cathedral	Zoo	Football,Ballet, Ice_Hockey,Ballet
Milan	Business_School,Cathedral, Shop,Tower	Zoo,Urban_Park	Rugby,Tennis

Table 4. Example of destination data for the multi-valued case

The process followed in the multi-valued case is very similar to the one of the uni-valued case. However, it has to be decided how to aggregate the score of each of the attribute values to obtain an overall score for that attribute. For example, in Table 4 it may be seen that Orlando has 3 kinds of Landmarks: Canal, Zoo and River. Therefore, when we are comparing this alternative with the current User Profile or with the Ideal User Profile we obtain a score for each landmark and they have to be merged to obtain the score for this attribute. Different merging procedures were already discussed in section 3.4.1. In this case study, after experimenting with some of these mechanisms (addition, average, maximum, minimum) we noticed that the best results were obtained when the maximum score of the attribute values was considered. Therefore, in the rest of this section this is the mechanism used to aggregate the scores of all the values of an attribute.

We will start the analysis by looking at the influence of the two parameters of the learning algorithm, X and Y, in the learning process. Figure 13 shows the distance between the current profile and the Ideal User Profile through the iterative learning process, fixing a value of Y=0.07 and testing different values for X, from 0.15 to 0.35. All the figures in this section correspond to the average results of 10 runs with different random initial profiles.

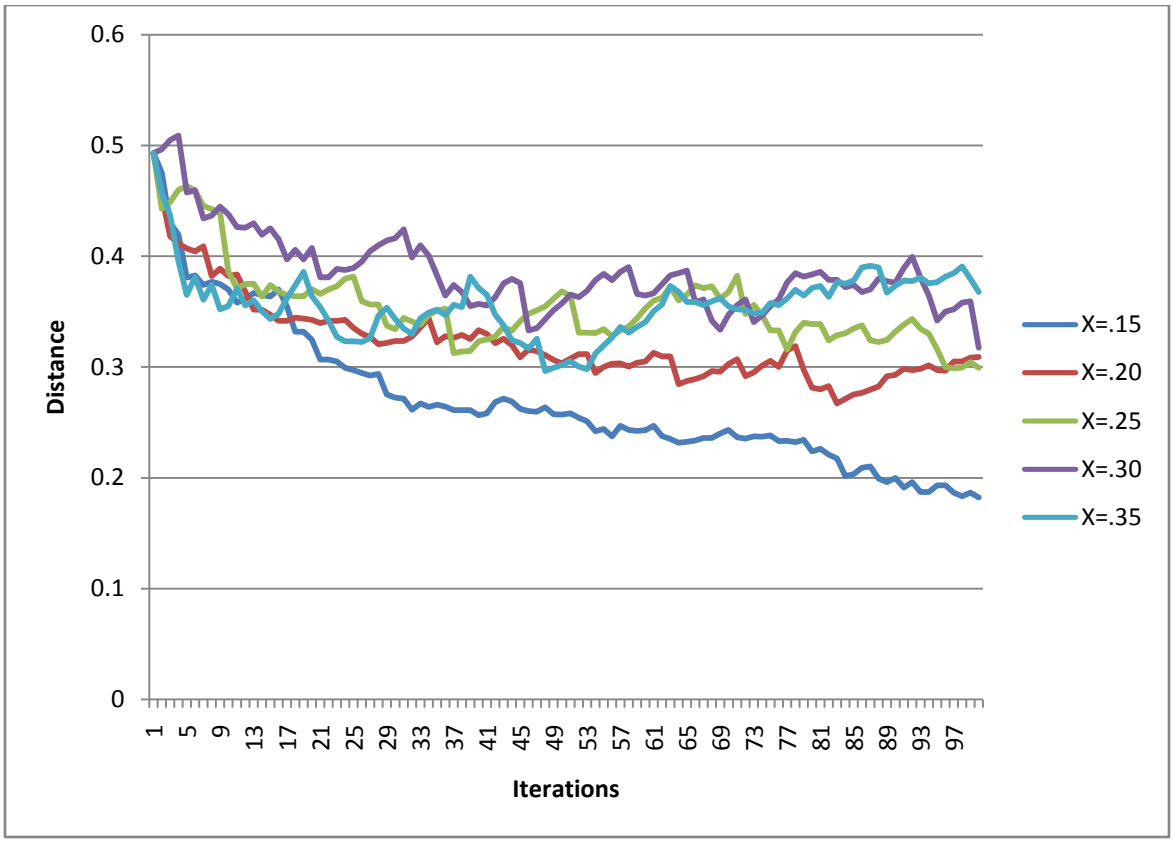

Figure 13: Influence of the X parameter (multi-valued scenario)

In this case it may be seen that the distance reduces quite quickly from 0.5 to a value between 0.3 and 0.4 (a very modest reduction), and then it stays quite uniformly in that interval for all the values of X over 0.20, without achieving much improvement. It may also be noticed that there are many increases and decreases of the distance, obtaining a figure which is much less smooth than the one of the uni-valued case (Figure 8). However, with a lower value of X (.15) the distance keeps decreasing in a very smooth way, reaching a value 0.18 in the last iteration (a result even better than the one obtained when a single value per attribute was considered). Thus, this value is the one that produces an appropriate learning of the user preferences for this case study.

We have also checked the influence of the Y parameter in the learning process. Recall that this parameter regulates how much the preference of a value is reduced when the value appears in an over-ranked alternative. It is possible to see how, fixing a value for X, the result can change with different values for Y. For example, figure 14 shows how the distance to the ideal profile evolves with X=0.27 and different values for Y.

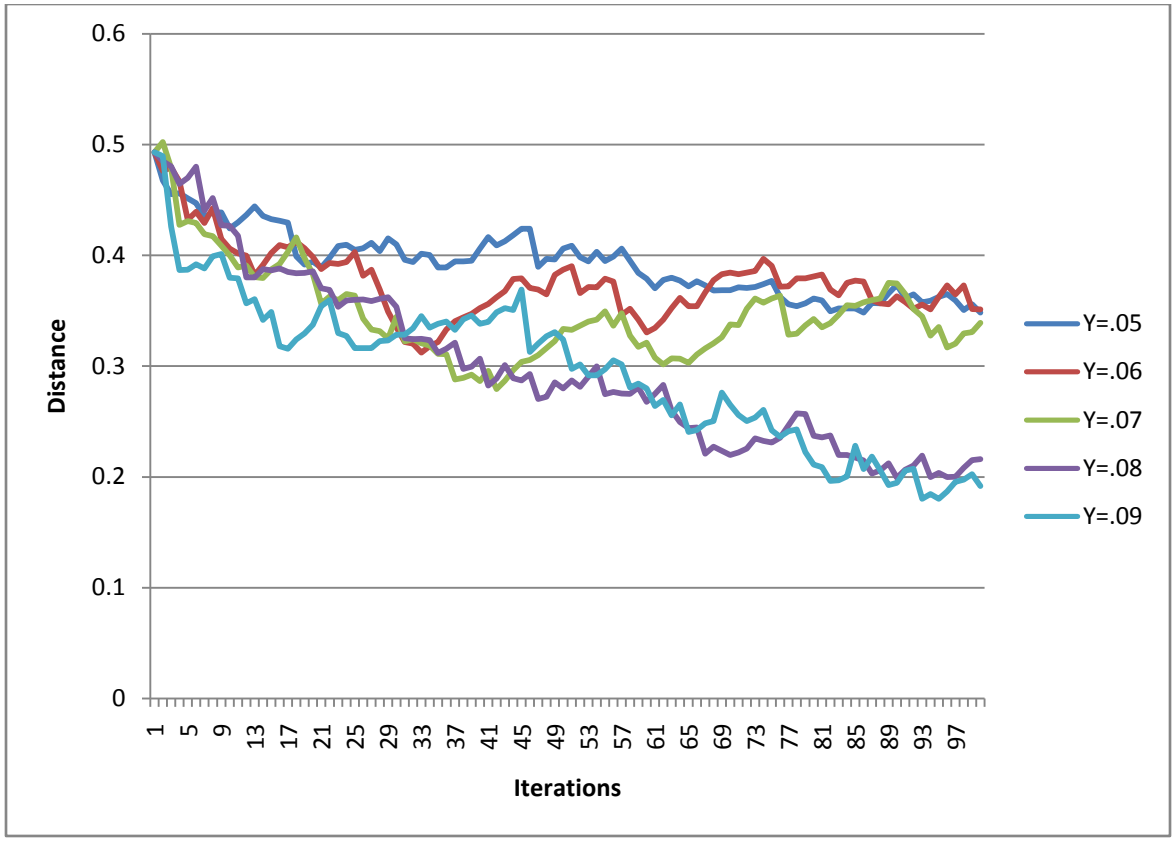

Figure 14: Influence of the Y parameter in the learning process (multi-valued scenario)

In this example the learning algorithm does not show a good performance when Y is between 0.05 and 0.07, but the distance to the ideal profile is clearly decreased, showing a good learning, when Y is 0.08 or 0.09.

These analysis of the X and Y parameters show that, as in the uni-valued case, they have to be correctly tuned to obtain a proper learning of the preferences. The following figure shows an example in which a bad choice of the parameters leads to a situation in which no learning takes place. Concretely, Figure 15 shows how the distance to the ideal profile does not actually decrease, but it rather increases, when X=0.5 and Y=0.02.

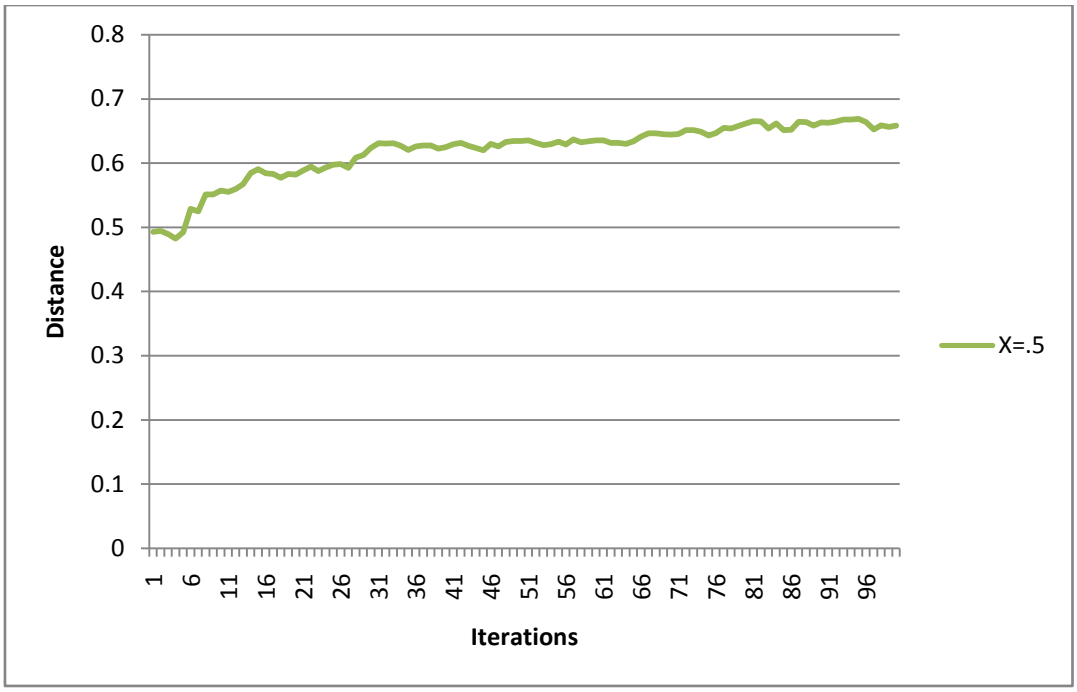

Figure 15: Example of a bad choice of the parameters (multi-valued case)

In the multi-valued case a grid search was also performed to find the best combination of values for the X and Y parameters, with X in the [0.15, 0.50] interval and Y in the [0.001, 0.15] interval. Among the best combinations, which reach a final distance below 0.2 between the current profile and the ideal profile, we can find X=0.15, Y=0.007 (seen in the deep blue line in Figure 13) and X=0.27, Y=0.09 (shown in the light blue line in Figure 14).

Another way to check that the system is correctly learning the preferences is to look at the position of the selected alternative in each of the iterations. If the preferences in the current profile are similar to those in the Ideal User Profile, the selected alternative should be in the first positions of the list of ranked alternatives. Figure 16 shows the position of the selected alternative in a test with X=0.27 and Y=0.07.

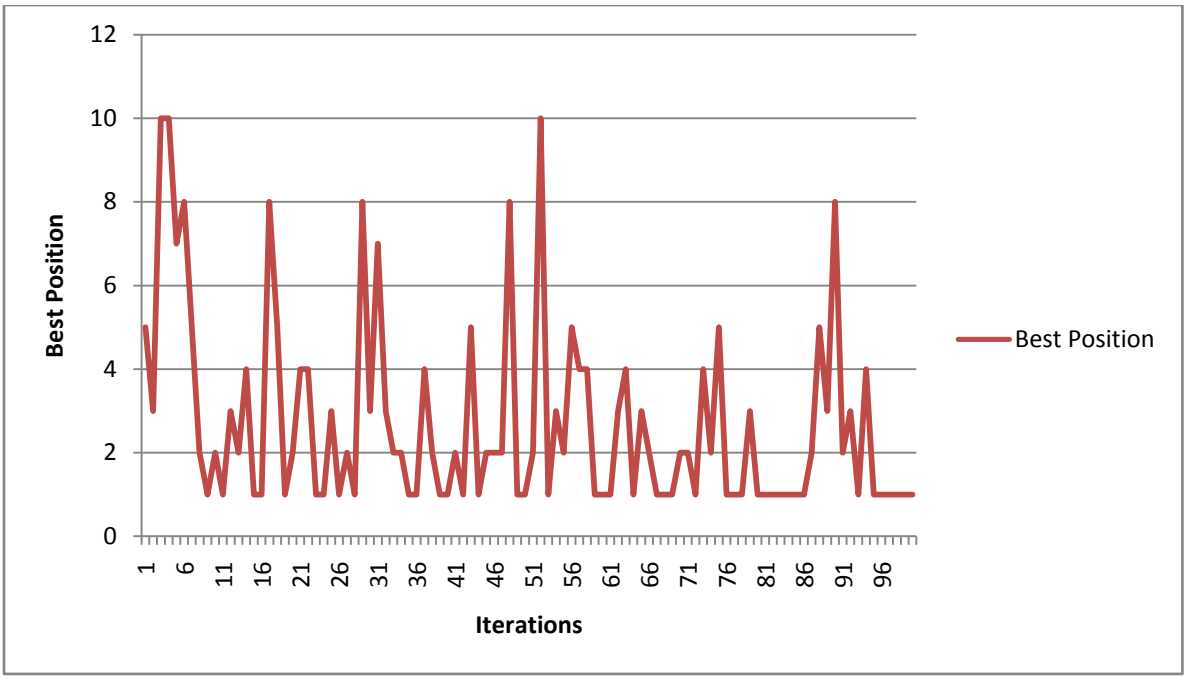

Figure 16: Evolution of the position of the selected alternative (multi-valued scenario)

It may be seen that the results are better than in the uni-valued case. There are 14 iterations in which the position of the selected option is below the 4th position. In particular, in the last 50 iterations the selection is 26 times in the first position, 8 times in the second position and 6 times in the third position. Therefore, in 80% of these iterations the preferences are good enough to order the alternatives in such a way that the one actually preferred by the user is in the top three (and in more than 50% of these iterations the best option is actually in the top position).

4.4 Conclusions

In summary, the results of this case study seem to indicate that the preference learning algorithm works correctly, both in the uni-valued and the multi-valued scenarios. In both cases after a few iterations the distance between the current preferences and the ideal ones reduces to a low level (below 0.2), and the system is able to analyze sets of 10 options and order them in such a way that the alternative actually preferred by the user appears in one of the top positions. Therefore, the algorithm would be usable by a recommender system to sort the alternatives to be shown to the user.

Chapter 5 Conclusions

In this Master thesis we have proposed a new method of preference learning in the case of semantic attributes (attributes whose values are classes of a domain ontology, which is a structured representation of the main concepts and relationships in the domain).This new method has been applied both on uni-valued and multi-valued semantic attributes. As a result, the work has made contributions in the area of Multi-Criteria Preference Learning in Recommender Systems in the case of semantic attributes.

The thesis started with the identification of different kinds of attributes to describe a set of decisional alternatives, including the new semantic attributes. After that, it showed how the general preference learning framework was instantiated in previous works for the particular case of numerical and categorical attributes. Then we explained how the preference learning methodology has been adapted in this Master Thesis to the new case of dealing with semantic attributes, defined as those that can take as values the most specific concepts of an ontology. Here the idea was to take advantage of the structure of the underlying domain ontology to guide the learning process. It is an iterative selection procedure in which, in each step, we have increased the preference on the values of the selected option and we have decreased the preference on the values of the over-ranked alternatives. In the case of this work, the structure of the ontology is basically used to decide how much the preference on an attribute value of the over-ranked alternatives should be decreased, depending on its semantic distance to the attribute value(s) in the selected option. Moreover, two different scenarios have been considered, depending on whether the semantic attributes can take only one value or multi-value.

The basic idea of the preference learning algorithm is to increase the preference on the values that appear in the attributes of the selected alternative (the one actually chosen by the user in a decisional process) and to decrease the preference on the values that appear in the attributes of the over-ranked alternatives (those that had been scored higher than the selected option, according to the current user preferences).When an attribute value appears both in the selected alternative but also in some of the over-ranked ones, we have given preference to the positive information of the user's actual selection so that we only increase the preference on that value. For the values of the selected option, they increased by a given fixed factor which is *X*. The value of this increasing factor is a parameter of the learning algorithm. When *X* is very high, the system changes the preference values in the user profile very quickly and it tends to prefer those values that appear in the selected options even if they do not appear very often. If *X* is lower, then the system does not have a high preference associated to a given value unless that value has appeared frequently in the user's selections. As a result, the learning process becomes smoother and slower. When the system adds *X* to the current preference degree, the system ensures that the new preference will not exceed 1. In the case of the preferences on the values that appear only in the over-ranked alternatives, they are decreased by a factor that depends on a parameter Y and on the distance to the attribute value of the selected option. Thus, at this point is where the learning system takes advantage of the information provided

by the ontology associated to the semantic attribute in order to decide how much the preference on a particular value should be decreased.

Finally, the new preference learning algorithm on semantic attributes has been applied to a particular scenario to test its performance under different circumstances. In particular, we have checked how successful was the algorithm for different values of its parameters X and Y. In this case study, the alternatives which are considered are tourist destinations, defined on some semantic attributes. A particular ideal profile to be learnt was defined, and it was analysed if the algorithm learnt these ideal preferences after analysing some selections of the user (which is simulated with the help of the ideal profile). After that, we applied the learning algorithm in the two cases considered in this work: uni-valued and multi-valued semantic attributes. The results of this case study indicated that the preference learning algorithm works correctly, both in the uni-valued and the multi-valued scenarios, if the values of the parameters X and Y are properly tuned. In both cases after a few iterations the distance between the current preferences and the ideal ones reduced to a low level (below 0.2), and the system was able to analyze sets of 10 options and order them in such a way that the alternative actually preferred by the user appears in one of the top positions. Therefore, it has been empirically proven that the algorithm could be used by a recommender system to sort the alternatives which are shown to the user.

Some future lines of research can be devised in the areas studied in this thesis. Some of them are the following:

- We would like to apply the learning algorithm in a real world domain.
- We should study more carefully how the values of the parameters X and Y could be (semi-) automatically tuned in a specific case.
- It would be interesting to integrate the preference learning algorithm for semantic attributes described in this work with the ones on numerical and categorical attributes developed in previous works ([Marín et al. 13a, Marín et al. 13b, Marín et al. 14]).
- A deeper analysis of the performance of the algorithm on the mulñti-valued case should be made, including the study of different ways to aggregate the preferences on the values of the attributes to obtain an overall preference for a given criterion, and different methods to merge the preference on each attribute to obtain the final score for each alternative.
- It would also be interesting to study which is the best number of alternatives to consider in each of the iterations. A preliminary study with fixed values for X and Y, depicted in Figure 17, shows that may be the learning process could be improved if more objects were considered in each of the iterations. The figure shows the best learning results when 25-30 options are considered.

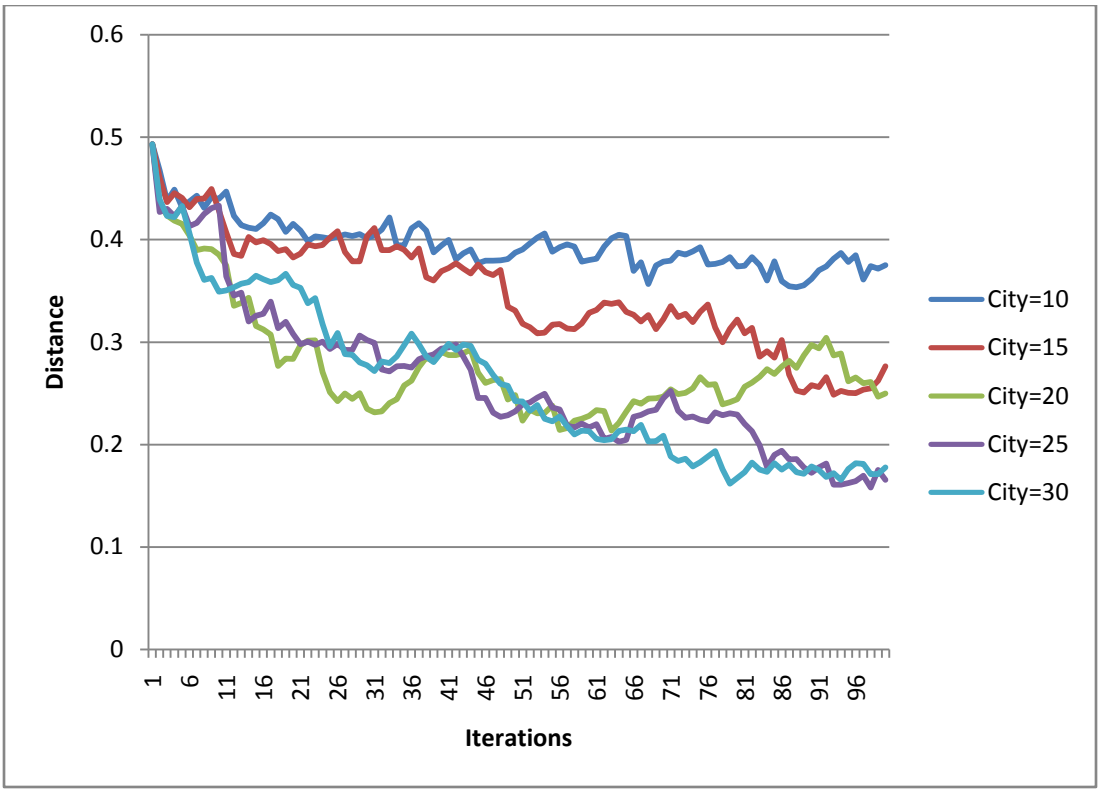

Figure 17: Influence of the number of alternatives considered per iteration

(X=0.2, Y=0.075, uni-valued case)

References

Burke, R.: Hybrid Web Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web, vol. 4321. Lecture Notes in Computer Science, pp. 377-408.Springer Berlin / Heidelberg, 2007.

Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. In Guarino, N., Poli, R. (eds) International Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation. Padova, Italy, pps 907-928, 1993.

Marín, L., Isern. D., Moreno, A.: Dynamic adaptation of numerical attributes in a user profile. Applied Intelligence, Vol. 39, pp. 421-437, 2013a.

Marín, L., Isern, D., Moreno, A., Valls, A. On-line dynamic adaptation of fuzzy preferences. Information Sciences, Vol. 220, pp. 5-12, 2013b.

Marín, L., Moreno, A., Isern, D.: Automatic preference learning on numeric and multi-valued categorical attributes. Knowledge-Based Systems, Vol. 56, pp. 201-215, 2014a.

Marín, L., Valls, A., Isern, D., Moreno, A., Merigó, J.M..Induced Unbalanced Linguistic OrderedWeighted Average and its application in multi-person decision making.The Scientific World Journal, Vol. 2014, Article ID 642165, 19 pages, 2014b.

Moreno, A., Valls, A., Isern, D., Marin, L., Borràs, J.: SigTur/EDestination: Ontology-based personalised recommendation of Tourism and Leisure Activities. Engineering Applications of Artificial Intelligence 26(1), 633–651, 2013.

Resnick, P., Varian, H.R.: Recommender Systems. Communications of the ACM 40(3), 56-58, 1997.

Ricci, F., Rokach, L., Shapira, B.: Introduction to Recommender Systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook. pp. 1-35. Springer US, 2011.

Sánchez, D. Domain ontology learning from the Web.PhD Thesis, Artificial Intelligence Program, Technical University of Catalonia, 2007.

Sánchez, D., Batet, M., Isern, D., Valls, A. Ontology-based semantic similarity: A new feature-based approach. Expert Systems and Applications 39 (9), pp. 7718-7728, 2012.

C.Vicient, D.Sánchez, A.Moreno.An automatic approach for ontology-basedfeature extraction from heterogeneous textual resources.Engineering Applications of Artificial Intelligence, Vol. 26, pp. 1092-1106, 2013.

Yager, R.R. On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Transactions on Systems, Man and Cybernetics 18(1), 183-190, 1988.

Annex-Ontologies used in the case study

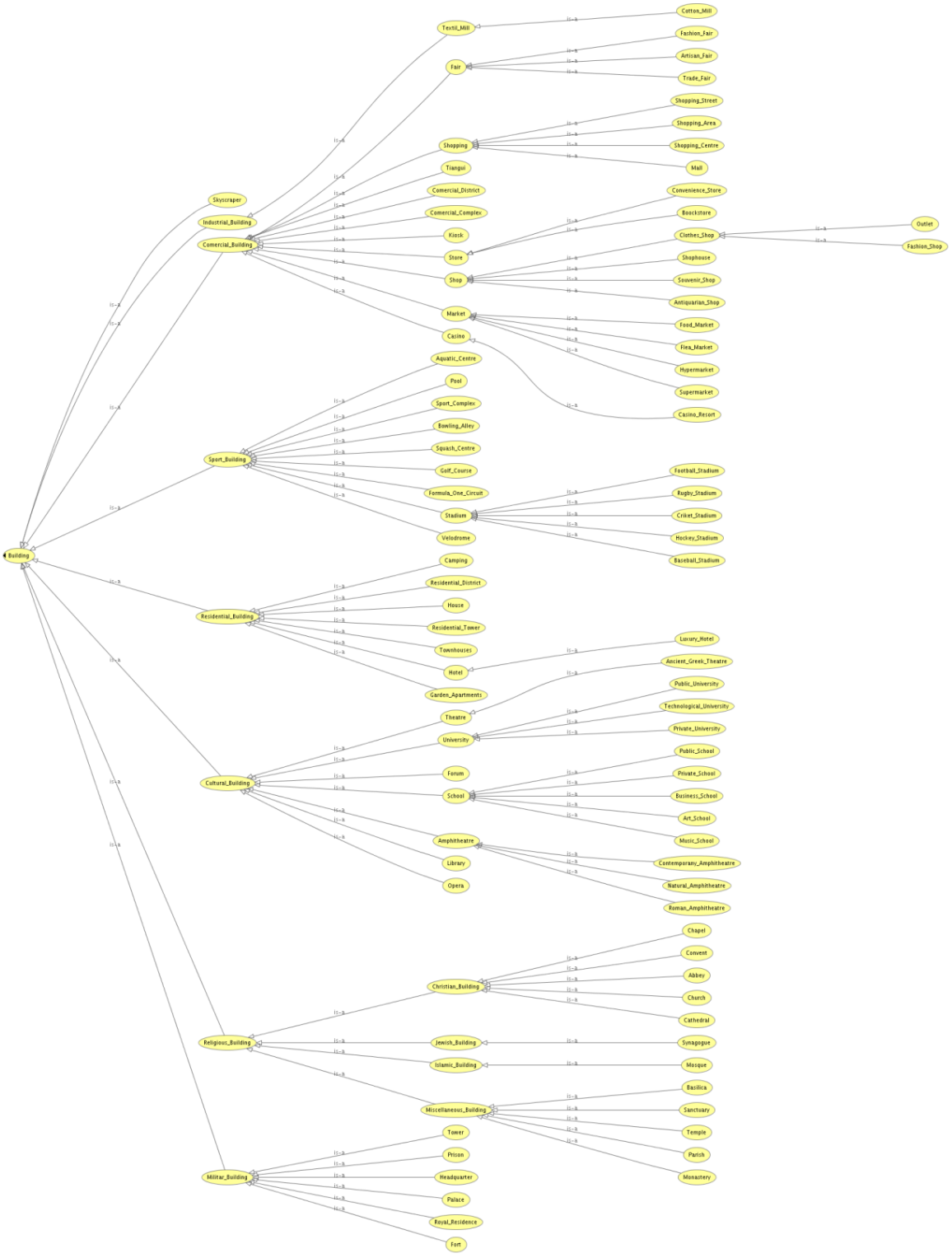

Figure 18. Buildings ontology

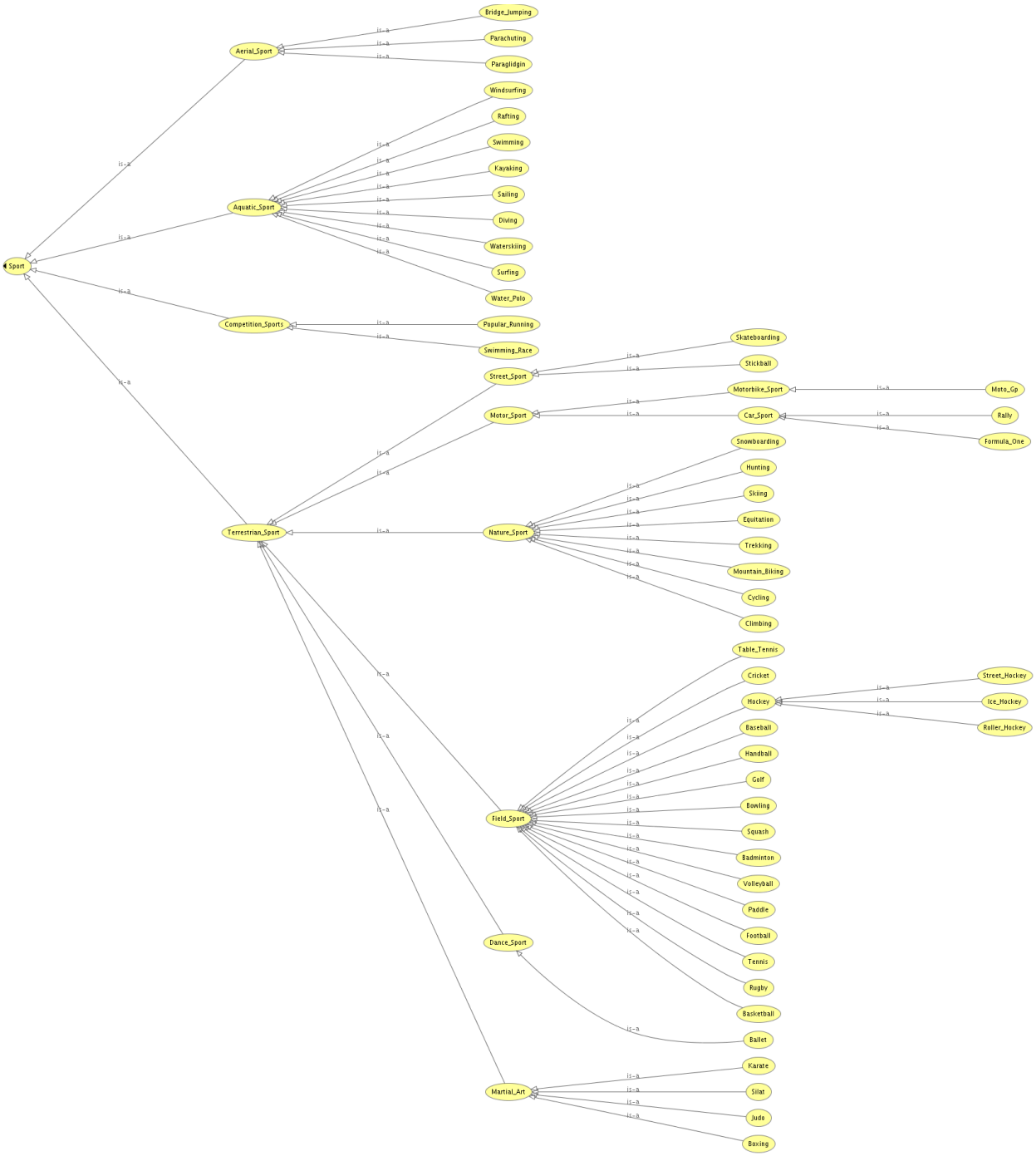

Figure 19. Sports ontology

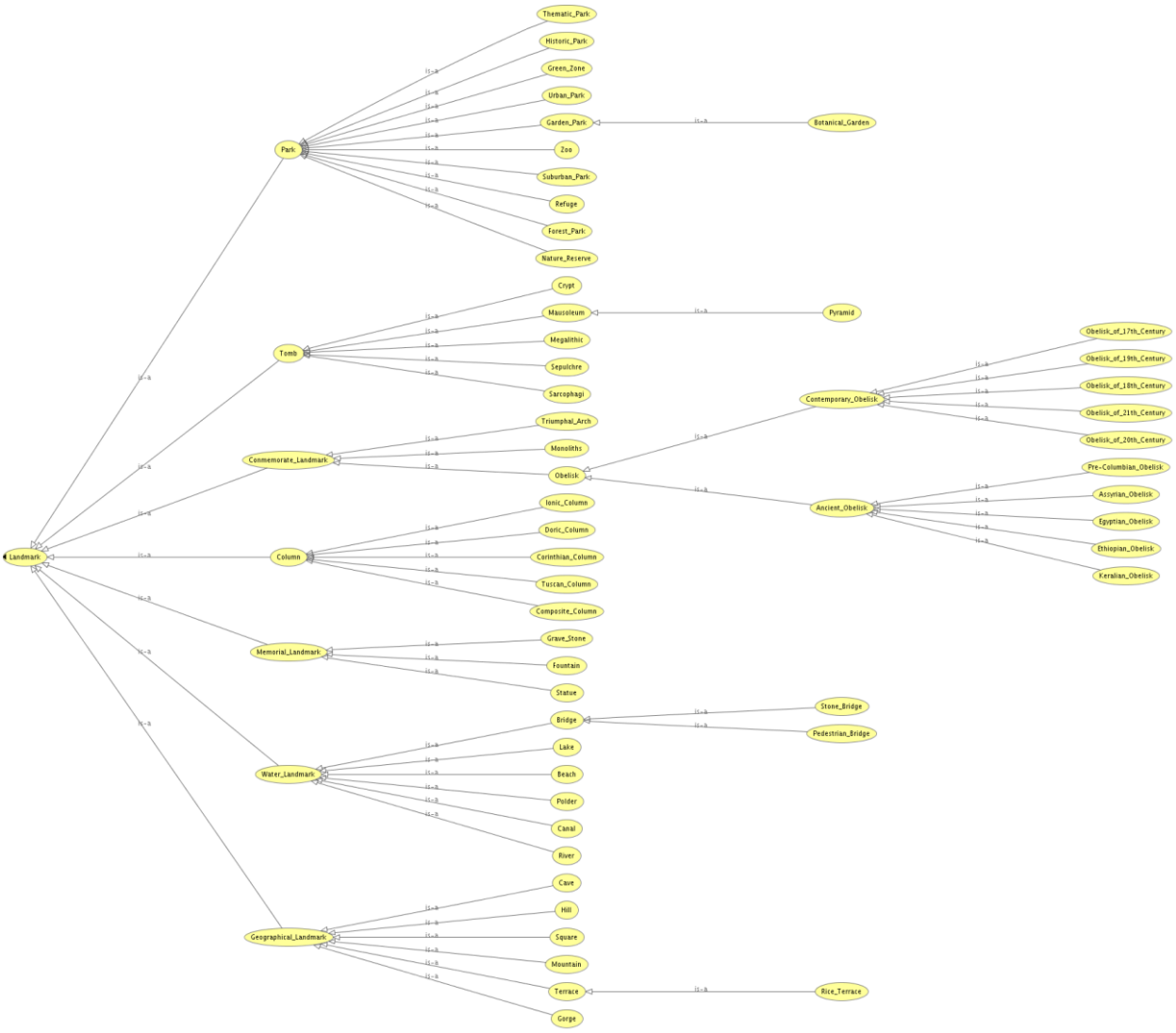

Figure 20. Landmarks ontology