UNIVERSITAT ROVIRA I VIRGILI
Màster en Seguretat Informàtica i Sistemes Intel·ligents

TESI DE MASTER

PREFERENCE-BASED RECOMMENDATION SYSTEM WITH UNSUPERVISED ADAPTATION OF USER PROFILES

Lucas Marin

Directors: Dr. Antonio Moreno i Dr. David Isern

Maig de 2010



iTAKA
Intelligent Technologies for Advanced Knowledge Acquisition

Agraïments

La realització d'aquesta tesis ha rebut el suport d'una beca pre-doctoral de la Universitat Rovira i Virgili i ha estat desenvolupada dins el grup de recerca *Intelligent Technologies for Advanced Knowledge Adquisition* (ITAKA).

En primer lloc, voldria agrair tot el suport i direcció que he rebut del Dr. Antonio Moreno i del Dr. David Isern durant tota la realització del treball, així com l'ajut i els consells rebuts durant la realització d'aquesta documentació i les publicacions relacionades amb el treball. Vull agrair també el consell i suggeriments rebuts de la Dra. Aïda Valls durant el treball i per donar-me l'oportunitat d'introduir-me al grup de recerca ara ja fa un temps.

Vull donar les gràcies també a la resta dels meus companys del grup de recerca ITAKA, la Montse, el David Sánchez i el Sergio, que fan possible un ambient immillorable i que sempre m'han donat un cop de mà quan l'he necessitat.

Finalment, també vull agrair als meus pares el suport durant tot el temps que he necessitat per dur a terme aquest treball.

Summary

List of figures ..... **ix**

1 Introduction **1**

1.1 Aims of this work **4**
1.2 Document organization **5**

2 Multiple Criteria Decision Analysis **7**

2.1 Basic concepts in decision making **8**
2.2 Defining a decision making problem **10**
2.3 Multi-criteria decision making methods **11**
2.4 Aggregation operators **12**
2.5 Mapping the user's preferences **14**
2.5.1 Categorical attributes 15
2.5.2 Quantitative attributes 17
2.6 Recommendation process **19**
2.6.1 Definition of the linguistic aggregation operator: ULOWA 21
2.6.2 Example of use of the ULOWA operator 25
2.7 Conclusions **28**

3 Management of Preferences **29**

3.1 State-of-the-Art on Preference Management **30**

3.1.1 Basic concepts 30
3.1.2 Information contained in profiles 31
3.1.2.1 Profiles structure 31
3.1.2.2 Generating initial profiles 32
3.1.3 Automatic learning of preferences 33
3.1.3.1 Learning techniques 33
3.1.3.2 Relevance feedback 35
3.1.3.3 Techniques for retrieving relevant information 36
3.1.3.4 Adaptation techniques 37
3.2 Information of the user profile 38
3.3 Adaptation of the user profile 40
3.3.1 Relevance feedback 41
3.3.2 On-line adaptation process 41
3.3.3 Off-line adaptation process 44
3.3.4 Regulation of the adaptation mechanisms 46
3.4 Conclusions 48

4 Evaluation 49

4.1 Testing platform 50
4.2 Testing algorithm 51
4.3 General performance 53
4.3.1 Testing environment 53
4.3.2 Performance on-line and off-line adaptation processes 56
4.3.3 Number of labels of distance 57
4.4 Analysis of the parameters 58
4.4.1 Characteristics extraction threshold 59
4.4.2 Number of stored selections 61
4.4.3 Number of changes in the profile 62
4.4.4 Evidences required to change a preference 63
4.5 Conclusions 65

5 Concluding remarks 67

6 References 71

Appendix A: Recommender platform 77

A.1. Structure 77

A.2. User interface 78

A.3. Functionalities and use 78

A.3.1. Platform access 78

A.3.2. Data files management 80

A.3.3. Context management 83

A.3.4. Profile management 84

A.3.5. Generation of recommendations 88

Appendix B: Publications 91

List of figures

Figure 1: Steps of the recommendation process 4
Figure 2: Linguistic preferences label set 16
Figure 3: Fuzzy number graphical representation 16
Figure 4: Representation of a numerical utility function 17
Figure 5: Numeric to linguistic translation example 19
Figure 6: Recommendation and user's profile adaptation components 19
Figure 7: Predefined preference label sets 20
Figure 8: Graphical example of ULOWA 26
Figure 9: Example of recommendations 46
Figure 10: Pseudo code of the simulator 52
Figure 11: Fuzzy term set *S* with 9 unbalanced labels 54
Figure 12: Profile distances with on-line and off-line methods (tourism) 56
Figure 13: Profile distances with on-line and off-line methods (medical centres) 57
Figure 14: Number of labels of distance 58
Figure 15: Influence of the characteristics extraction threshold (tourism) 60
Figure 16: Influence of the characteristics extraction threshold (medical centres) 60
Figure 17: Influence of the number of stored selections (tourism) 61
Figure 18: Influence of the number of preference changes (tourism) 62
Figure 19: Influence of the number of preference changes (medical centres) 63
Figure 20: Influence of the level of evidence (tourism) 64
Figure 21: Influence of the level of evidence (medical centres) 64
Figure 22: General patterns of behaviour in numerical attributes 69

1 Introduction

We are currently living in the so called *knowledge society* in which we are constantly in contact with means that facilitate our access to unlimited sources of information and knowledge that help us in our daily activities. Due to the astonishing speed at which new content is created and published, extracting *concrete* information usually becomes a difficult and overwhelming task.

In this scenario, users are constantly confronted with situations in which they have to select one alternative from a set of possibilities. Those daily scenarios, such as deciding which radio station to listen, or which program to watch in TV, or decide the location to go in holidays require the inclusion of user preferences to sort all the alternatives according to the user interests. Imagine the case of choosing a place to spend our holidays. Almost every user who wants to plan a travel uses Internet to find information about his destination, searching through thousands of web pages, trying to find useful content to help him decide at which place he should go. In that scenario, the user identifies locations that he is interested in, and evaluates them taking into account some *properties* that identify each destination, named *criteria*.

Before to select an appropriate alternative from a set, the system needs to create and maintain the *user profile* including relevant information about some criteria. When users interact with a computer, they provide a great deal of information about themselves. Successful interpretation of these data streams is required to tailor systems to each individual's behaviour, habits and knowledge [48]. In addition, users interests about an issue may evolve due to changes in their personal conditions (*e.g.*, change of location, having a child).

All the possibilities the user faces when searching for information to solve a decision problem (such as a "tourist destination") form a set of alternatives from which the user has to make a decision. As said before, having such a source of information as Internet generates such a huge set of alternatives that it is almost impossible for the user to make an accurate decision taking into account all of the criteria and alternatives.

This manuscript presents a *recommender system* designed in two separate modules: rating and ranking a set of alternatives, and the adaptation of the user profile. The recommender system receives a set of alternatives composed by several criteria. The user profile stores the preferences of the user over all those criteria. The first step of the recommender is to rate all the alternatives taking into account the user profile in order to prioritize them according to his particular circumstances. Then, the user selects the most appropriate alternative from this sorted list. Afterwards, the adapting algorithm collects the information provided by this selection, the set of non selected alternatives, and all past selections, to infer which changes can be made to the current profile.

When rating alternatives, the first decision to be taken is the domain in which each criterion should be described. There are two main possibilities: *numerical* and *linguistic* [10, 60]. This work aims to build a general framework that permits to work with both formats together giving more flexibility to the system and to the user in deciding how his preferences can be expressed. For instance, following the example of holidays, a numerical attribute can be *hotel price*, and a linguistic attribute can be *main activities*, or *available public transport*. As shown through the manuscript, the recommender system mainly deals with linguistic variables: all alternatives are aggregated using a linguistic aggregator, the information is shown to the user using this domain, and finally, the user profile is adapted under these circumstances. Numerical attributes are translated to linguistic and vice versa, in order to homogenise the algorithm.

Learning and adapting profiles are widely studied research areas. The user profile may be learnt by extracting patterns from training data, *e.g.*, using clustering techniques. Concerning the adaptation of the profile, some information about how the user is interacting with the system is needed. The two most common ways to obtain this information, known as *relevance feedback* [56], are to use information given *explicitly* by the user and to get the information *implicitly* by observing the user behaviour. Some *hybrid* systems combine implicit and explicit approaches [50].

On the one hand, explicit feedback is obtained when users are required to explicitly evaluate items, indicating how relevant or interesting an item is to the user in a numeric or linguistic scale. These systems, such as [50] and [52], offer high performance and simplicity. On the other hand, implicit feedback is obtained by monitoring the user actions and automatically inferring the user preferences. The amount of data collected is very large, the computation needed to derive recommendations for adaptations is extensive, and the confidence in the suitability of these adaptations is likely to be relatively low. This approach has been less explored, although some existing works (*e.g*, [4, 18]) show promising results [48].

Explicit feedback is sometimes not desirable because it has some serious limitations. One of those is that numeric scales may not be adequate for describing the reactions humans have to items. Moreover, as studied in [53], users are reluctant to spend time giving explicit feedback and only 15% of the users would supply it even if they were encouraged to do so. For these reasons, we present an algorithm that adapts the personal profile of the users in an unsupervised way. We assume that the system is used regularly, so that it can learn, from the usage history, how the preferences of the decision maker evolve. The main goal is to obtain adequate recommendations by adapting the preference functions over time.

As it will be described later, we have implemented a Web-based platform that permits to simulate all those stages and extract some conclusions in an unsupervised fashion. The Web-based system has been designed as a general

framework that permits to solve decision problems using multi-criteria decision aiding techniques. The separation between the domain-dependant (*e.g.*, rating and adaptation parameters) and the domain-independent data (*e.g.*, rating, ranking and adaptation processes) permits to understand the evolution of values, as well as to reuse the same methods in a wide range of problems. The system has been prepared to make automatic simulations, which permit to study the influence of several parameters of the proposed adaptation algorithm.

1.1 Aims of this work

To sum up the previous introduction, a *recommender system* must be able to aid the user when making a decision by evaluating and rating a set of alternatives taking into account the user interests, stored in a profile. Moreover, the whole process must be able to collect enough user feedback in order to maintain the user information up to date.

So, our work has been divided in two major parts: the first one including the algorithms and techniques to rank a set of alternatives, and the second one including the techniques to maintain the profile updated. The basic structure of these elements is shown in Figure 1.

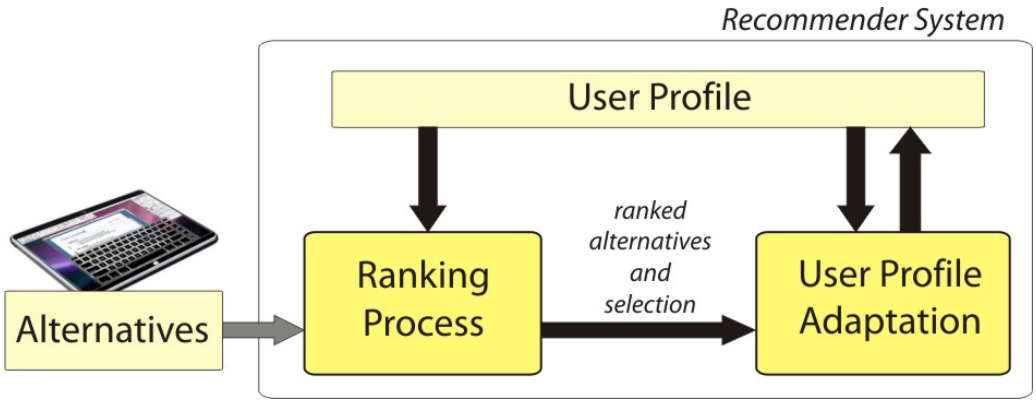


Figure 1: Steps of the recommendation process

Regarding the first part, current linguistic aggregation operators have some limitations when we want to add expressivity to the definition of the preferences. For this reason we have designed a *new linguistic aggregation operator* to deal

with that issue called Unbalanced Linguistic Ordered Weighted Averaging operator (ULOWA). Concerning the second part, the main goal is to design a *new method for dynamic preference learning*. This method will allow adapting a user profile in an unsupervised and dynamic way.

To achieve these goals it is necessary to fulfil the following objectives:

- Design and codify the structure which stores the user profile in a concrete domain.
- Design and implement an aggregation method for numeric and categorical data based on the LOWA operator.
- Use the previous aggregation to build a decision aid platform which helps the user to decide among a set of options using his personal data included in the profile.
- Design and implement an adaptation method that automatically evolves the user profile according to his interaction with the recommender platform.
- Evaluate the results and the performance of the recommender system on a particular domain (*e.g.*, recommendation of touristic products).

1.2 Document organization

This Master Thesis is organised as follows:

Chapter 2 presents a state-of-the-art on multi-criteria decision analysis for supporting recommendation tasks paying special attention to the aggregation of information. The steps of the procedure are well known in the literature, but in this chapter a new aggregation operator is proposed to deal with balanced and unbalanced fuzzy sets taking into account their membership functions. A deep description of this aggregation operation can be found in [30] and [42].

Chapter 3 is divided in three parts. The first one presents a state-of-the-art on management of preferences, studying current methodologies and techniques to represent and maintain user interests in a profile structure. In the second part the decisions about how we decided to store and manage user preferences are explained. In the third part a new methodology for adapting a user profile is presented, by using the feedback obtained from the user interaction with the platform. The work has been described in [40-41] and [40].

Chapter 4 includes the testing and evaluation of the adaptation methods explained in the previous chapter by using a web platform developed for this purpose. An study of how the different parameters affect the final result of the adaptation is also included.

Chapter 5 summarises a list of conclusions of this work and devises some lines of future work order to improve the results.

Appendix A presents in detail the web-based platform which has been implemented to test and evaluate the recommender system with profile adaptation designed along this Master Thesis. Its main functionalities are explained as well as the technologies used to develop the whole platform.

Appendix B includes the publications developed within this work. This includes the papers accepted in the "2010 IEEE International Conference on Fuzzy Systems" and in the "71st Meeting of the European Working Group ‹‹Multiple Criteria Decision Aiding››", and the submitted papers to the "13è Congrés Internacional de l'Associació Catalana d'Intel·ligència Artificial" and to the "5th Multidisciplinary Workshop on Advances in Preference Handling" within the "19th European Conference on Artificial Intelligence".

2 Multiple Criteria Decision Analysis

Traditional decision aid is the activity of the person who, through the use of explicit but not necessarily complete formalised models, helps to obtain responses to the questions posed by the stakeholder in a decision process [56]. These elements work towards recommending, or simply favouring a certain alternative. At the end, the decision maker is completely free to behave after the recommendation is made.

Even when decision aiding is provided for a single decision maker, it is rare for him to consider a single clear criterion. *Multiple criteria decision analysis* (MCDA) deals with various points of view (criteria). MCDA techniques can be used to identify a single most preferred option, to rank options, to list a limited number of options for subsequent detailed evaluation, or to distinguish acceptable from unacceptable possibilities.

The main contribution of this chapter is to define a method that permits the management of linguistic and numerical user preferences to evaluate alternatives containing them. A new linguistic aggregation operator aggregates the alternatives in a single value that is used to rate them.

2.1 Basic concepts in decision making

As indicated by Roy in [56], three concepts usually play a fundamental role for analysing and structuring the decision aiding process in close connection with the decision process itself. Those concepts are the *alternatives* which can be elected as possible solutions, the *criteria* which are used to evaluate those alternatives, and the *decision problem* itself. We also consider necessary to introduce other concepts which we will be using quite often along this document related to MCDA such as *decision maker*, *context* and *preference*.

As indicated on the introduction, the *decision maker* is the entity that faces the decision problem. In group decision making, a problem involves more than one decision maker, but that is not the case that we are considering in this work.

The *context* is the domain in which the decision must be taken. We consider as a context not just the environment in which the solutions of the problem can be taken but also the set of variables that interfere in the decision making process, which will be explained below.

Preferences model the decision maker's personal interests in order to help the decision making process to find the most suitable option for him. Preferences can be defined as the valuation the user does to each criterion supporting the decision.

The *alternatives* constitute the set of possible options in which the solution of the decision problem is among.

A *criterion* is a tool constructed for evaluating and comparing potential alternatives according to a point of view or preferences. This evaluation must take into account for each alternative, all the pertinent effects or attributes linked to the point of view considered. The *performance* of the alternative *a* according to the criterion *g* is denoted by *g(a)*. According to [3], criteria should be:

- Able to discriminate among alternatives.

- Complete: include all goals.
- Operational: meaningful to the decision maker's understanding of the implications of the alternatives.
- Non-redundant: avoid double counting.
- Few in number: to keep the problem dimensions manageable.

In MCDA, the word *problem* refers to the way in which decision aiding is envisaged. There are four main kinds of problems, although they are not the only possible ones, as can be seen on [5].

- In some cases, decision aiding consists only in the elaboration of an appropriate set of potential alternatives, building a suitable family of criteria and determining, for all the alternatives, their performances. For designating this manner of conceiving of decision aiding's aim without seeking to elaborate any prescription, or recommendation, the term *description problematic* is used.
- When the aid is oriented towards and lies on a *selection* of a small number of "good" alternatives in such a way that a single alternative may finally be chosen we are talking about a *choice problematic*.
- Sometimes the aid is oriented towards and lies on an *assignment* of each alternative to one category among those of a family of predefined categories. Then a *sorting problematic* is faced.
- We find a *ranking problematic* when the aid is oriented towards and lines on a *complete* or *partial pre-order* of the alternatives which can be regarded as an appropriate instrument for comparing alternatives between one another.

Having defined the previous concepts, the decision making process can be introduced.

2.2 Defining a decision making problem

Any decision making process involves all the actions needed to obtain the most feasible solution for a well-defined problem. As indicated in [3], a general multi-criteria decision making process can be divided into the following steps:

Step 1: Define the problem. The finality of this step is to define the decision problem in a concise and unambiguous way. It is crucial that the decision makers and stakeholders come to an agreement when defining the problem in order to proceed to the next step.

Step 2: Determine requirements. The next step is to determine which conditions any acceptable solution to the problem must meet.

Step 3: Establish goals. Goals go beyond the minimum essential requirements defined, in the previous step, to *wants* and *desires*. The goals may be conflicting but this is a natural concomitant of practical decision situations.

Step 4: Identify alternatives. In this step, alternatives that accomplish the requirements determined in the second step are identified. All of the alternatives may solve the problem, in one way or another.

Step 5: Define criteria. Decision criteria will discriminate among alternatives. They are defined as objective measures of the goals to quantify how well each alternative achieves the goals. It is necessary to verify that:

- What is apprehended by each criterion is sufficiently intelligible for each of the stakeholders.
- Each criterion is perceived to be a relevant instrument for comparing potential alternatives along the scale which is associated with it without prejudging their relative importance.
- The *n* criteria considered all together satisfy some logical requirements (exhaustiveness, cohesiveness, and non redundancy) which ensure coherence of the family of criteria.

Step 6: Select a decision making tool. The selection of an appropriate tool is not an easy task and depends on the concrete decision problem, as well as on the objectives of the decision makers. Sometimes "the simpler the method, the better" but complex decision problems may require complex methods, as well.

Step 7: Evaluate alternatives against criteria. Alternatives can be evaluated with quantitative methods, qualitative methods, or any combination. Criteria can be weighted and used to rank the alternatives. Also, both sensitivity and uncertainty analyses can be used to improve the quality of the selection process.

Step 8: Validate solutions against problem statement. After the evaluation process has selected a preferred alternative, the solution should be checked to ensure that it truly solves the identified problem. Compare the original problem statement to the goals and requirements. A final solution should fulfil the desired state, meet the requirements, and best achieve the goals within the values of the decision makers. Once the preferred alternative has been validated, the decision-making support staff can present it as a recommendation to the decision-maker.

2.3 Multi-criteria decision making methods

With respect to how the recommendation of the decision process provides a personalised answer to some decision problem, different approaches can be considered. For the problems faced in this document, we propose the use of a multi-criteria decision making (MCDM) technique [22].

MCDM methods have their foundations in philosophy, economics, social choice or even game theory. Note that aggregating the opinion of preferences of voters or individuals of a community into collective or social preferences is quite similar to devising comprehensive preferences of a decision-maker from a set of conflicting criteria. MCDM focuses on decision making problems when the number of the criteria and alternatives is finite, and the alternatives are given explicitly. MCDM techniques can partially or completely rank the alternatives, in

order to identify the single most preferred alternative or select a list of a limited number of preferred alternatives.

The two main approaches to MCDM are *outranking methods* and *multi-attribute utility theory* (MAUT) [22-23].

- *Outranking methods* seek to establish the strength of evidence favouring selection of one alternative over another, on the basis of pairwise comparison of alternatives. The result of those comparisons is represented by an outranking binary relation *S* defined on the set of alternatives *A*, such that for the alternatives *a* and *b*, there exists the relation *aSb* if there are enough arguments to decide that *a* is at least as good as *b*, whereas there is no essential argument to refute that statement.
- On the other hand, *MAUT* is based on assigning a global utility value to each alternative in A. This global utility is a combination of the marginal utilities that each criterion assigns to an alternative [39]. MAUT is a simple approach that is quite used in MCDM. Usually, a higher ranking value means a better performance of the alternative, so the alternative with the highest ranking value is the best of the alternatives.

Besides those two main approaches, other methods have been developed; a revision of them can be found in [22].

2.4 Aggregation operators

When more than one criterion come into play, it is necessary to consider a way to take into account all criteria comprehensively in order to be able to compare alternatives to one another. Considering two alternatives *a* and *b*, *a* can be better than *b* for some of the criteria and *b* better than *a* for others. However, a final *judgement* has to be done in order to say if *a* is better than *b* taking into account all of the criteria. This problem is called the *aggregation problem*.

Some MCDA techniques are based on the use of *aggregation operators* [61], which are functions that combine (*aggregate*) a set of values into a single one (representative). This representative value is used to rate, to rank and to sort a set of given alternatives. These operators, which are formally described as follows, typically satisfy the unanimity (idempotency) and monotonicity properties.

Let $a = (x_1, \dots, x_n)$ be a set of information sources in a domain *D*, and let $C: D^n \to D$ be a consensus function defined in a given domain *D* to aggregate values $x_1, \dots, x_n$. The function satisfies the following properties:

a) Unanimity: $C(x, \dots, x) = x$
b) Monotonicity: $C(x_1, \dots, x_n) \leq C(x'_1, \dots, x'_n)$ if $\forall i\ 1 \leq i \leq n,\ x_i < x'_i$

Other properties such as continuity, symmetry, associativity or neutral element can also be satisfied by an aggregation operator although just unanimity and monotonicity are mandatory.

As an example, we can consider the harmonic mean as one such aggregation operator considering the domain *D* as numerical:

$$C(a) = C(x_1, \dots, x_n) = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

There exist a large number of aggregation operators applicable to a broad range of data representation formalisms including ordinal and nominal scales [67]. In general, aggregation operators can be classified according to the data type they handle (numerical, fuzzy, qualitative, and heterogeneous) or according to their mathematical properties. The main families of aggregation operators are ([8, 61, 68]):

- Means (averaging functions), like arithmetic mean, weighted mean, geometric mean, or harmonic mean.

- Medians, which try to find a value that is more representative of a typical value than the mean. It essentially discards very high and very low values.
- Ordered weighted averaging functions (OWA), which are also averaging aggregation operators which associate weights not with a particular input, but rather with its value. According to the nature of the data, numerical or linguistic, OWA or LOWA operators can be defined, respectively.
- Choquet and Sugeno integrals, which are two classes of averaging functions defined with respect to a fuzzy measure. They are useful to model interactions between the criteria.
- Conjunctive and disjunctive functions, like the so-called triangular norms and conorms respectively. Minimum and maximum functions, product and probabilistic sum, Lukasiewicz norms, or drastic sum and product, are several examples of these aggregation functions that are used in fuzzy set theory and fuzzy logic.
- Mixed aggregation, used in situations where high input values are required to reinforce each other, whereas low values pull the output down. In this case, the aggregation function has to be disjunctive for high values, conjunctive for low values, and perhaps averaging if some values are high and some are low.

2.5 Mapping the user's preferences

With the concepts explained before, an alternative can be considered as an array which contains a value for each criterion that the MCDA system takes into account. Those values, which inform about the state of the criteria in the specific alternative, are given a valuation according to the users' preferences before aggregating them.

For example, if we want to use MCDA techniques to choose a hotel from a set of hotels in a specific city of destination, we can consider criteria such as

"number of stars", "price" or "rooms views". When looking at the "rooms views" criteria, one tourist will prefer having a sea view while another can prefer having a mountain view. So, the same criteria value can have different valuations depending on the user.

When an alternative needs to be evaluated, *preference functions* are used to obtain the preference score of each attribute according to some particular user profile. Once the value of preference criteria has been obtained, an aggregation operator is used to calculate a global preference score. These scores are used to rank the options.

Two types of criteria have been considered in the recommendation system developed within this work:

- *Numerical criteria*: preference values are given in the [0,1] range.
- *Linguistic criteria*: preferences are given in a set of ordered linguistic terms.

The adaptation step, however, just manages linguistic criteria.

The next subsections give more details about how criteria are built from the different types of attributes.

2.5.1 Categorical attributes

A fuzzy linguistic approach is proposed for making assessments in a qualitative setting [26]. Linguistic preferences are defined as a set of fuzzy numbers, each one representing a value of preference. A linguistic preference scale permits to deal with uncertainty in the evaluation of the preferences in categorical variables.

A common fuzzy linguistic scale, *S*, is defined and the decision maker can attach each of the available linguistic terms to each of the possible values of the attribute. Figure 2 depicts an unbalanced set of preferences with seven possible values of preference (*VL*: Very Low, *L*: Low, *M*: Medium, *AH*: Almost High, *H*: High, *VH*: Very High, *P*: Perfect).

The linguistic scale $S = \{s_i\}, i\epsilon\{0, \dots, T\}$ is a finite and totally ordered set of terms on a reference domain $X = [0,1]$, with an odd cardinality. One of the labels corresponds to the neutral value and the remaining terms are placed around it. The cardinality of the set must be small enough so as not to impose useless precision and rich enough in order to allow an appropriate discrimination level. The usual cardinality values are 5, 7 and 9.

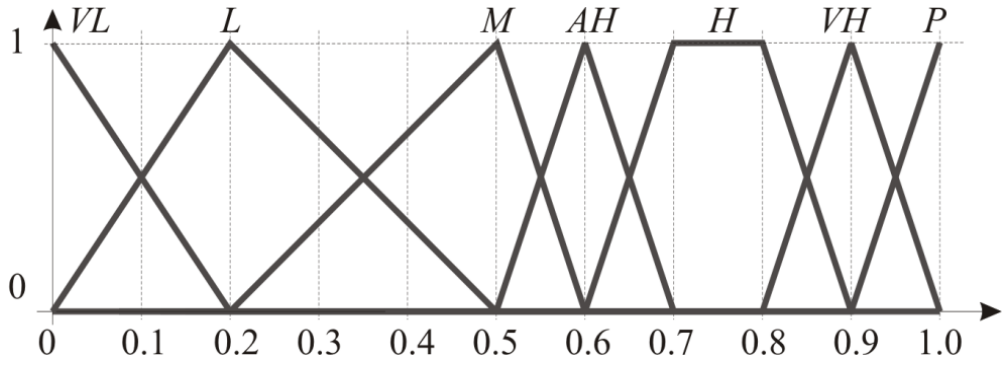


Figure 2: Linguistic preferences label set

In this work, we consider that all labels are represented by linear functions, which are defined by four points (a, b, c, d), as can be seen on Figure 3. Then, we can distinguish four different cases: a crisp value if $a = b = c = d$, a trapezoid if $a \neq b \neq c \neq d$ (for example H in Figure 2), a triangle if $a \neq b = c \neq d$ (for example L in Figure 2), and a rectangle if $a = b$, $c = d$ and $b \neq c$.

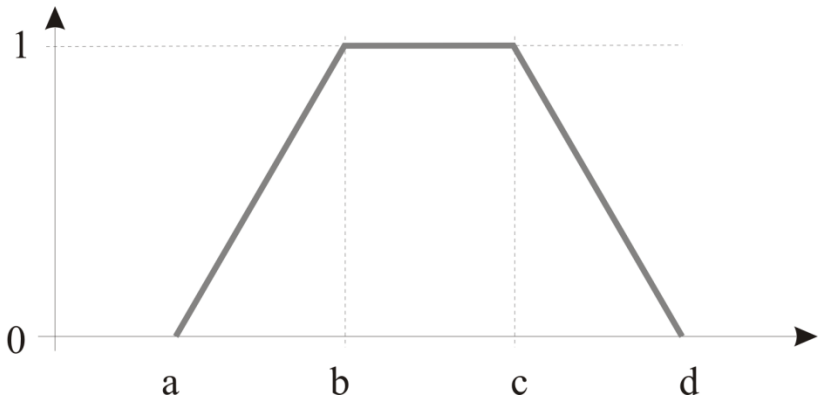


Figure 3: Fuzzy number graphical representation

Then, the preference function for categorical values consists on a mapping from the set of possible values of a categorical variable, *X*, into the set of preference terms *S*.

$$g_{Ling}: X \rightarrow S$$

The mapping is initially given by the user, and then it can be adapted using the learning algorithm included into the recommender system as explained later on in this document. So, in a "Tourist destination" example case, a user may express a "Low" preference for the "Alpine" value and "Very High" for the "Mediterranean" value in the "Destination climate" criterion.

2.5.2 Quantitative attributes

Quantitative attributes are evaluated using a numerical value function that gives a score in the [0,1] range, with 1 representing the maximum preference degree and 0 the minimum [21]. The value function assumes that the decision maker has a maximum preference for a single value of the reference domain of the variable, which is denoted as v_{pref}. The preference function is defined symmetrically, that is, the increasing and decreasing rate (left and right sides of v_{pref}) are equal. The parameter *d* indicates when the preference must have a 0 value. Finally, the form of the function is defined by the parameter *m*, as shown in Figure 4.

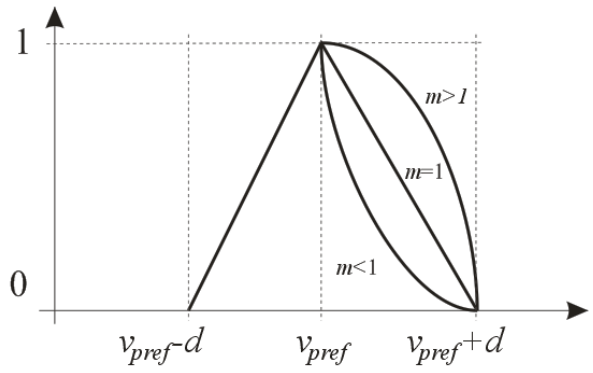


Figure 4: Representation of a numerical utility function

Formally, let x be the criteria value, v_{pref} the most preferred value for this attribute, m the slope of the utility function and d its width. Then, the preference function g_{Num} for the value x can be expressed as:

$$g_{Num}(x) = 1 - \left(\frac{|x - v_{pref}|}{d}\right)^m$$

Following the example of recommending touristic destinations, the user can define that he prefers a place located in the mountains, indicating that the preferred altitude is 1000 m. Using the defined model, the preferences will decrease as the altitude is below 1000 m and also if it is higher than 1000 m. If $d = 700$, this means that cities that are below 300 m from the sea level and cities that are above 1700m, will be evaluated with 0 (*i.e.*, $v(300) = 0, v(1700) = 0, v(1000) = 1$).

Although the user preference profile stores and manages these numerical value functions, during the aggregation stage, the numerical preference scores are translated to a common linguistic domain (S). This translation is required because the aggregation operator requires that all the values belong to the same linguistic scale. This translation is made using a numeric-to-linguistic function defined in [20].

The following example describes how that translation is done: consider that we want to translate the numeric value 0.45 to a linguistic value into the domain described in Figure 2. The linguistic translation for this value is the linguistic preference label which has the grater activation at this point. As can be seen in Figure 5, the 0.45 value (marked in red) has an activation value of 0.20 for the 'L' ('Low') preference and 0.80 for the 'M' ('Medium') level of preference. So, the linguistic translation of the 0.45 numeric value is 'M'.

When all of the levels of activation of the preferences in a certain numeric point are equal, the resulting linguistic label is the greater. For example, the

linguistic translation of 0.35 would also be ‘M’, although the levels of activation of ‘L’ and ‘M’ in this point are equal (0.5).

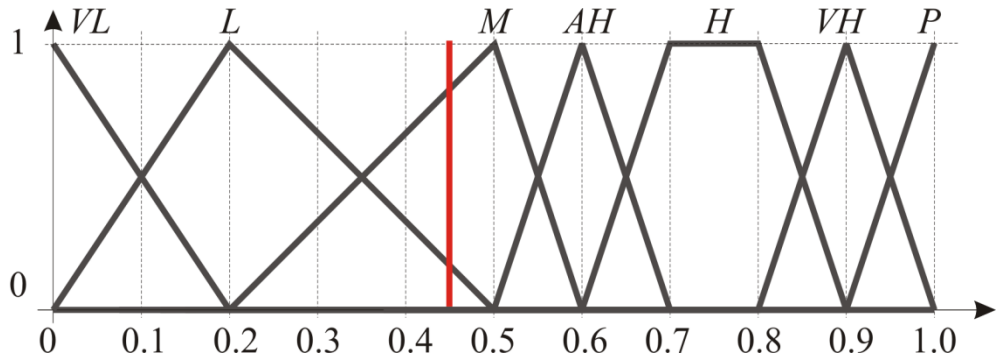


Figure 5: Numeric to linguistic translation example

2.6 Recommendation process

The goal of the recommendation process is to rate and rank all alternatives of a given domain. By applying an aggregation-based method, we obtain a label for each alternative that is used to rank them all. The proposed method allows handling both linguistic and numerical attributes. Figure 6 depicts the process followed in the ranking stage.

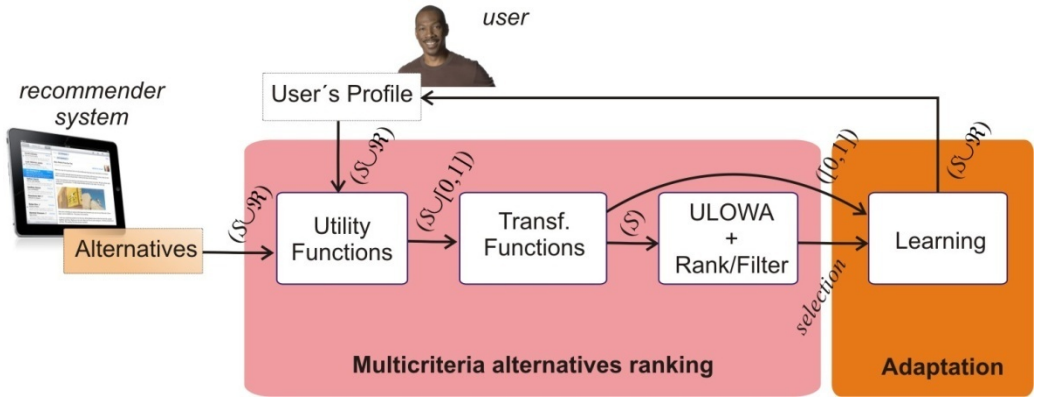


Figure 6: Recommendation and user’s profile adaptation components

The first step in the recommendation process is the definition of a context. In order to do it, it is necessary to choose the parameters that define the numeric and linguistic variables.

Numeric variables are defined by the slope (*m*) of the utility function and its width (*d*). To facilitate its definition, the *d* parameter is defined as a percentage over all of the values between the minimum and the maximum numbers that a numeric variable can have.

Although the system is designed to deal with any linguistic set, the platform stores six different sets shown in Figure 7. The sets (a), (b) and (c) are balanced with 5, 7 and 9 terms respectively. The sets (b), (d) and (f) define three unbalanced sets with different levels of preference.

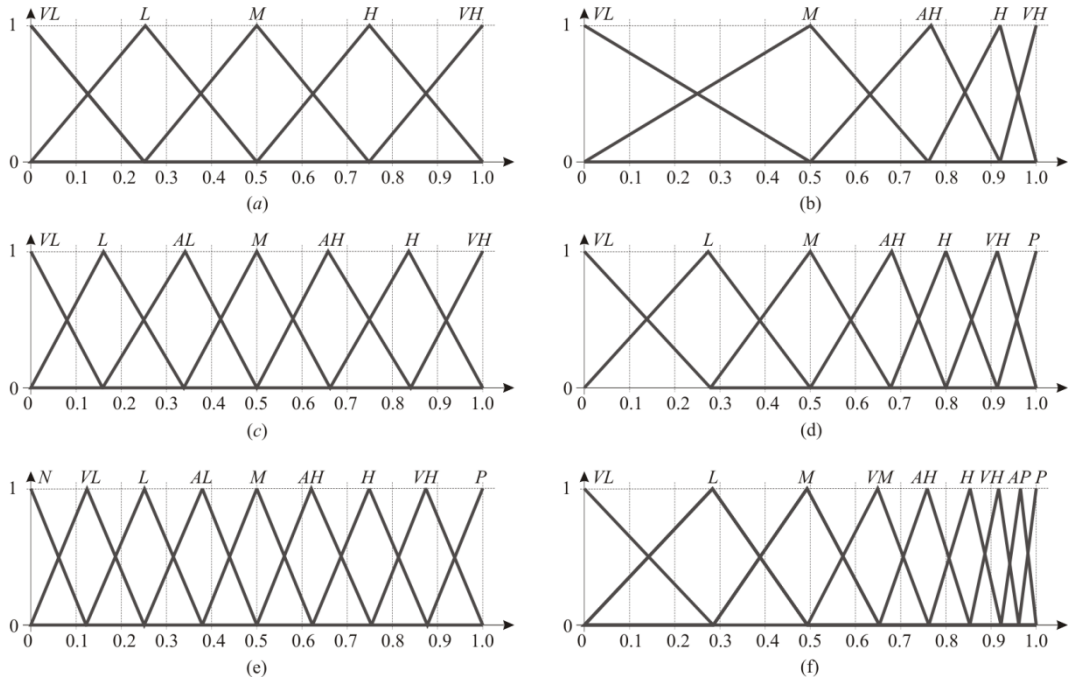


Figure 7: Predefined preference label sets

The aggregation-based rating process is described in the following steps:

Step 1) Transforming all the values contained in the alternatives into a common vocabulary by using the preference functions described previously. Numerical attributes are transformed into the [0,1] domain and the linguistic attributes into the *S* domain.

Step 2) The numerical preferences in $[0,1]$ are translated into the linguistic domain S by means of the numerical-to-linguistic transformation function defined by Delgado *et al.* [20] (linguistic preferences are left without changes) explained before, obtaining the transformed vector called $alt_i = \{a_k\}\ (a_k \in S)$.

Step 3) The aggregation operator ϕ is applied to all alt_i in order to obtain a linguistic rating for each of them in the S domain.

Step 4) Finally, all alternatives can be ranked using the rating values. Then, a filtering is performed to show to the user only the best alternatives, so that he can confirm one of them.

2.6.1 Definition of the linguistic aggregation operator: ULOWA

The problem of aggregating information has been widely studied [1, 26, 59]. There exist several methods to aggregate numerical values as well as linguistic terms. The family of OWA operators are in the class of mean operators. They are idempotent, monotonic and commutative [59, 68]. The LOWA aggregation operator was defined by Herrera and Herrera-Viedma [27], as an extension of the OWA operator defined by Yager [68] to deal with linguistic attributes.

In LOWA, a set of equally-informative (*balanced*) terms is assumed. That is, the membership functions of the fuzzy terms are symmetric and the terms are uniformly distributed around a mid (*i.e.*, neutral) term. For example, when evaluating the users' satisfaction on some topic, linguistic labels like "very low", "low", "almost low", "medium", "high", "very high", "perfect" can be used [64].

However, recently many authors have noticed that there are some problems that require a more flexible definition of the set of linguistic terms [28, 65]. In some cases, fuzzy sets should be asymmetric or more positive than negative terms are needed (or *vice-versa*), leading to an *unbalanced* set of labels. An example is the set of terms used in grading evaluations $S = \{F, D, C, B, A\}$

(being *D* the neutral; *A*, *B* and *C*, positive marks, and *F* – fail – the unique negative value). Many other real-life applications also make use of unbalanced terms, such as [35, 43, 65].

Some extensions of well-known aggregation operators have been defined to deal with unbalanced sets of terms. The proposals of [28] and [11] are based on the linguistic 2-tuple representation model, which attaches to each label a deviation value that permits to work with more precision. In [65] the Uncertain Linguistic Weighted Average is proposed. It restricts to the case of sets of labels where the absolute value of the deviation between the indices of two adjacent labels increases as the indices increase. An additional number is attached to each label, indicating its degree of separation from the mid term. The aggregation operator uses those additional numbers to calculate the result.

Here we assume the classic fuzzy linguistic model that associates a fuzzy membership function to each linguistic value and operates in terms of those fuzzy sets. The LOWA operator defined in [29] is the basis of the new operator, called ULOWA (Unbalanced Linguistic Ordered Weighted Average). LOWA has been chosen because it has been extensively used in many applications and it is computationally efficient. LOWA relies on a symbolic model of the set of linguistic terms, and the aggregation is done considering only the positions of the labels in the set of terms, but without taking into account any fuzzy set associated to the terms.

The aggregation procedure of ULOWA is the same than the one used in LOWA. However, in the convex combination of two labels, ULOWA exploits the knowledge given by the membership functions during the aggregation procedure. This approach, although it is focused on managing unbalanced label sets, can also be used with balanced label sets giving the same results than the LOWA operator.

Let $A = \{a_1, \dots, a_m\}$ be a set of labels, where $a_j \in S$, that have to be aggregated with respect to a set of weights $W = \{w_1, \dots, w_m\}$ such that $w_i \in [0,1]$ and $\sum w_i = 1$. Those weights specify the decision-maker policy, so that we can

emphasize different arguments based upon their ordered position. The weight vector is generated according to a selected policy or *linguistic quantifier*.

The classic binary logic allows for the representation of two quantifiers, "there exists" and "for all". As noted in [68], in natural language we use many additional quantifiers such as "almost all", "few", "many", "most", etc. As indicated by Yager [68], the use of a linguistic quantifier in multi-criteria decision making provides a deeper and more unifying interpretation of the weighting function associated with an aggregation operator.

Yager indicated that the weighted vector W is a manifestation of the quantifier underlying the aggregation process. In particular, if a decision-maker suggests that he wants Q of the objectives satisfied, then we obtain the weighting vector as $w_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right), i = 1, \dots, n$, being the membership function of Q, as follows:

$$Q(r) = \begin{cases} 0 & if\ r < a \\ \dfrac{r-a}{b-a} & if\ a \le r \le b \\ 1 & if\ r > b \end{cases}$$

With $a, b, r \in [0,1]$. Herrera *et al.* [27] pointed out some examples of non-decreasing proportional fuzzy linguistic quantifiers in the form of *quantifier(a,b)*: "most"(0.3,0.8), "at least half" (0,0.5) and "as many as possible" (0.5,1). When a fuzzy linguistic quantifier Q is used to compute the weights of LOWA operator $\emptyset$ it is symbolized by $\emptyset_Q$.

The LOWA operator, $\emptyset_{LOWA}$, is defined as:

$$\begin{aligned} \emptyset_{LOWA}(a_1, \dots, a_m) &= W \cdot B^T = C^m\{w_k; b_k; k = 1, \dots, m\} \\ &= w_1 \otimes b_1 \oplus (1 - w_1) \otimes C^{m-1}\{\beta_h; b_h; h = 2, \dots, m\} \end{aligned}$$

Where $\beta_h = \frac{w_h}{\sum_2^m w_h}$, $h = \{2, \dots, m\}$, and $B = \{b_1, \dots, b_m\}$ is a permutation of the elements of A, such that $B = \sigma(A) = \{a_{\sigma(1)}, \dots, a_{\sigma(m)}\}$, where $a_{\sigma(j)} \leq a_{\sigma(i)}, \forall\, i \leq j$.

C^m is the convex combination operator of m labels. If $\exists j \in \{1, \dots, m\}, w_j = 1$ and $\forall k \in \{1, \dots, m\}, k \neq j, w_k = 0$, then $C^m\{w_i; b_i; i = 1, \dots, m\} = b_j$.

When m=2, with $b_1 = s_j$ and $b_2 = s_i$ such that $s_j, s_i \in S(j \geq i)$ then $C^2\{w_i; b_i; i = 1{,}2\} = w_1 \otimes s_j \oplus (1 - w_1) \otimes s_i = s_k$ such that $k = \min\{T, i + round(w_i(j - i))\}$.

Notice that, in this definition, the resulting label s_k is obtained considering the position of the labels s_j and s_i in the linguistic domain, but without considering the form of the membership function associated to each term. Considering only the order between the terms implies the assumption that terms are defined with symmetric and uniformly distributed membership functions. For this reason, as it has been said, the LOWA operator is not able to deal with unbalanced sets of linguistic terms.

The new ULOWA operator is defined on the basis of the LOWA operator. In fact, the definition is the same until we have to make the aggregation of two labels (m=2). The convex combination of two terms $b_1 = s_j$ and $b_2 = s_i$, with $s_j, s_i \in S(j \geq i)$ is calculated taking into account the membership functions of labels s_j and s_i as follows:

$$C^2\{w_i; b_i; i = 1{,}2\} = w_1 \otimes s_j \oplus (1 - w_1) \otimes s_i = s_k,$$

$$\text{with } k = \operatorname{argmax}_{i \leq p \leq j}\{Sim(s_p, \delta)\}$$

Here, δ is an intermediate crisp number defined as $\delta = (x_k, x_k, x_k, x_k)$ with $x_k = COG_x(s_i) + w_1(COG_x(s_j) - COG_x(s_i))$. Where COG_x is the coordinate

x of the centre of gravity (*COG*, [15]) of a fuzzy label. The *COG* of a fuzzy term A is represented as $(COG_x(A), COG_y(A))$ and can be calculated as follows:

$$COG_y(A) = \begin{cases} \frac{1}{6}\left(\frac{a_3 - a_2}{a_4 - a_1} + 2\right) & \text{if } a_1 \neq a_4 \\ \frac{1}{2} & \text{if } a_1 = a_4 \end{cases} \quad \textbf{(1)}$$

$$COG_x(A) = \frac{COG_y(A)(a_3 + a_2) + (a_4 + a_1)\left(1 - COG_y(A)\right)}{2}$$

The function *Sim* evaluates the similarity between two fuzzy sets. Due to the number of available functions, we made a comparative study in [30] paying special attention in our requirements: the comparison of a crisp number and a fuzzy term. At the end, none of the analysed functions fit exactly these requirements and we use and adaptation of the function defined by Chen [16].

We adapted it by avoiding the scale factor designed to weight the similarity of two general fuzzy numbers with different heights. The adapted Chen's similarity function is defined as follows:

$$Sim(A, B) = \sqrt[4]{\prod_{i=1}^{4}(2 - |a_i - b_i|)} - 1 \quad \textbf{(2)}$$

2.6.2 Example of use of the ULOWA operator

Figure 8 illustrates the aggregation procedure of the two extreme labels (*VL* and *P*) with a "mean" weighting policy. The figure shows the *COG*s of both labels and their intermediate crisp number δ used to find the result of the aggregation. After the application of the similarity function, the resulting label is the neutral term *M*. Using the LOWA operator, the result would have been *AH*, which is not the intuitive expected result of the mean average of these two opposite evaluations, considering the meaning of the terms.

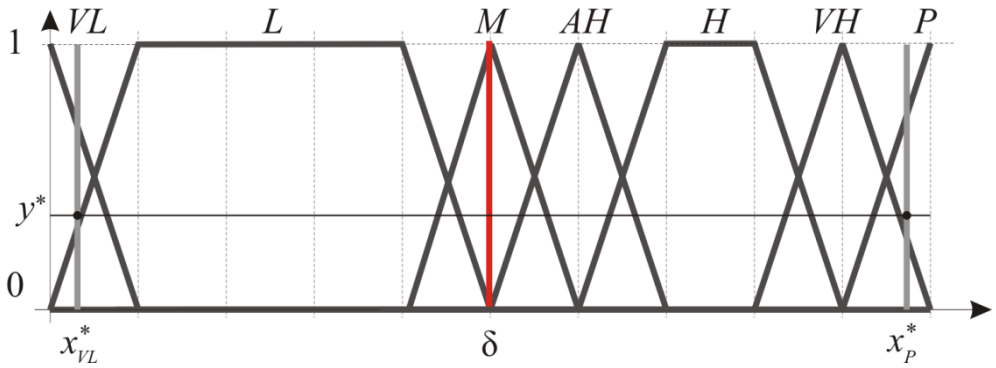


Figure 8: Graphical example of ULOWA

Considering that $S = \{VL, L, M, AH, H, VH\}$ is defined as $VL = (0,0,0,0.1)$, $L = (0,0.1,0.4,0.5)$, $M = (0.4,0.5,0.5,0.6)$, $AH = (0.5,0.6,0.6,0.7)$, $H = (0.6,0.7,0.8,0.9)$, $VH = (0.8,0.9,0.9,1)$, $P = (0.9,1,1,1)$, and, since the aggregation policy is 'mean' $w_{VL} = w_P = 0.5$, the steps followed to obtain the aggregation of 'VL' and 'P' are:

1) Calculate the *COG*s of the 'VL' and 'P' labels.

$$COG_y(VL) = \frac{1}{6}\left(\frac{a_3 - a_2}{a_4 - a_1} + 2\right) = \frac{1}{6}\left(\frac{0}{0.1} + 2\right) = 0.333$$

$$COG_x(VL) = \frac{COG_y(VL)(a_3 + a_2) + (a_4 + a_1)\left(1 - COG_y(VL)\right)}{2} = \frac{0 + 0.1\left(\frac{2}{3}\right)}{2} = 0.033$$

$$COG_y(P) = \frac{1}{6}\left(\frac{a_3 - a_2}{a_4 - a_1} + 2\right) = \frac{1}{6}\left(\frac{0}{0.1} + 2\right) = 0.333$$

$$COG_x(P) = \frac{COG_y(P)(a_3 + a_2) + (a_4 + a_1)\left(1 - COG_y(P)\right)}{2} = \frac{\frac{2}{3} + 1.9\left(\frac{2}{3}\right)}{2} = 0.967$$

2) Calculate the intermediate crisp number δ.

$$x_k = COG_x(VL) + w_{VL}(COG_x(P) - COG_x(VL)) = 0.033 + 0.5 * 0.934 = 0.5$$

$$\delta = (0.5,0.5,0.5,0.5)$$

3) Calculate the similarities with δ of each label comprised between 'VL' and 'P'.

$$Sim(VL,\delta) = \sqrt[4]{\prod_{i=1}^{4}(2-|VL_i-\delta_i|)} - 1 = \left[\sqrt[4]{1.5*1.5*1.5*1.6} - 1\right] = 0.5244$$

$$Sim(L,\delta) = \sqrt[4]{\prod_{i=1}^{4}(2-|L_i-\delta_i|)} - 1 = \left[\sqrt[4]{1.5*1.6*1.9*2} - 1\right] = 0.7378$$

$$Sim(M,\delta) = \sqrt[4]{\prod_{i=1}^{4}(2-|M_i-\delta_i|)} - 1 = \left[\sqrt[4]{1.9*2*2*1.9} - 1\right] = 0.9494$$

$$Sim(AH,\delta) = \sqrt[4]{\prod_{i=1}^{4}(2-|AH_i-\delta_i|)} - 1 = \left[\sqrt[4]{2*1.9*1.9*1.8} - 1\right] = 0.8987$$

$$Sim(H,\delta) = \sqrt[4]{\prod_{i=1}^{4}(2-|H_i-\delta_i|)} - 1 = \left[\sqrt[4]{1.9*1.8*1.7*1.6} - 1\right] = 0.7464$$

$$Sim(VH,\delta) = \sqrt[4]{\prod_{i=1}^{4}(2-|VH_i-\delta_i|)} - 1 = \left[\sqrt[4]{1.7*1.6*1.6*1.5} - 1\right] = 0.5984$$

$$Sim(P,\delta) = \sqrt[4]{\prod_{i=1}^{4}(2-|P_i-\delta_i|)} - 1 = \left[\sqrt[4]{1.6*1.5*1.5*1.5} - 1\right] = 0.5244$$

4) The aggregation result is the label with the greatest similarity, in this case ‘M’, with a similarity value of 0.94.

2.7 Conclusions

In this chapter we have presented how a decision making problem is formulated and what are the elements that intervene in its resolution. Our most important contribution in this part of the work is the ULOWA aggregation operator, which allows aggregating information such as preferences in an unbalanced term set, adding expressivity in the declaration of label term sets.

The ULOWA aggregation operator work has been accepted on the "2010 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)" and it is going to be presented in Barcelona in July. Furthermore, the recommendation process we designed in this work has been exposed as a whole in the "71st Meeting of the European Working Group ‹‹Multiple Criteria Decision Aiding››" in March 2010 in Torino, Italy. Both publications can be found in the appendix B.

3 Management of Preferences

As stated on the introduction, any decision making system needs enough information from the user who wants to be recommended in order to give asserted and personalised results. The methods to obtain this personal information (from now on, preferences) as well as the techniques to update them are the key to personalised and accurate recommendations.

The first section of this chapter presents the state-of-the-art on preference management and is divided in four subsections. The first one introduces some basic concepts related to preference management. The second one constitutes a study on techniques to manage user information by means of *profiles*, including the definition of its representation and its initial generation. The third and fourth subsections develop an overview on current methods to learn and update users' profiles, respectively, in order to give more accurate recommendations, including how relevant information for profile update is obtained and how it is used to modify the profile.

The second part of this chapter presents a novel methodology to adapt a user profile by using the information extracted from the user interaction with the system. This methodology constitutes an unsupervised adaptation process which updates the user profile by extracting implicit knowledge every time the user asks for a recommendation.

3.1 State-of-the-Art on Preference Management

3.1.1 Basic concepts

In [51] a *decision problem* is defined as the case of somebody who tries to compare objects taking into account different points of views or criteria. Comparing two objects can be seen as looking for one of the following situations:

- Object *a* is "before" object *b*, where "before" implies some kind of order between *a* and *b*, with such an order referring either to a direct *preference* (*a* is preferred to *b*) or being induced from a measurement and its associated scale (*a* occurs before *b*, *a* is longer, bigger, more reliable, than *b*).
- Object *a* is "near" object *b*, where "near" can be considered either as indifference (object *a* or object *b* will do equally well for some purpose), or as a similarity, or again could be induced by a measurement (*a* occurs simultaneously with *b*, they have the same length, weight, reliability).

From a decision aiding point of view we traditionally focus on the first situation. Ordering relations is the natural basis for solving ranking or choice problems. The second situation is traditionally associated with problems where the aim is to be able to put together objects sharing a common feature in order to form "homogeneous" classes or categories (a classification problem).

As stated in [23, 51], given a set of alternatives *A*, establishing how each element of *A* compares to each other element of the same set from a *preference* point of view enables to obtain an order which might be used to make either a choice on the set *A* (identify the best) or to rank the set *A*. It is necessary to consider whether it is possible to establish such an ordering relation and of what type for all pairs of elements of *A*. It is also necessary to establish what the meaning of the lack of preference is.

As pointed out before in this document, we use a linguistic scale to express the level of preference of each of the criteria values of the alternatives that the recommender system evaluates (such as *Very low, Low, Medium, High* and *Very high*).

Those preferences, which define the level of interest of a user in a concrete object, are grouped in a data structure known as *profile*. Through this document, we consider *profile learning* the action to extract enough information from a user to build an initial profile of him, accurate enough to start making correct recommendations. On the other hand, *profile adaptation* is considered as the action to update the initial user profile as that user preferences change. Those last terms will be discussed more widely below.

3.1.2 Information contained in profiles

3.1.2.1 Profiles structure

As briefly indicated in the previous section, the first step to design a preference-based system is to define data structures to store information about the users' interests. As stated in [47], the following models have been used to build this structure known as *user profile*: history-based model, vector space model, weighted n-grams, semantic networks, classifier-based models, user-item ratings matrix and demographic features.

A profile following the *history-based* model stores a list of activities resulting from the interactivity between the user and the system, for example a list of purchases or a web navigation history. An example of this kind of profiles can be found on online shops such as Amazon.com.

In the *vector space model*, preferences are extracted from items (usually documents) which are represented with a vector of features (terms) with an associated value. This value can be a Boolean (which indicates the presence of the feature) or a real number (which indicates the frequency, relevance or probability of the feature). As an example, WebMate [14] utilises a multiple vector

representation in which the basic idea is to represent each document as a vector in a vector space so that documents with similar content have similar vectors. Each dimension of the vector space represents a word and its weight.

In *weighted n-grams*, items are represented with a net of words with weights in the nodes and edges. This technique is applied in PSUN [58]: with the idea that words tend to occur one after another a significantly high number of times, fixed length consecutive series of *n* characters are extracted and organised with weighted links representing the co-occurrence of different words.

Semantic networks are able to represent and store semantic relations and meaning among concepts. In ifWeb [44] semantic networks are used to describe typical pattern of topics of interest to the users.

Systems which learn from the user using some sort of classifier as learning technique retain the structure of the *classifier* itself as the user profile. Examples of that kind of profiles are neural networks [9], decision trees [38], induced rules [6] and Bayesian networks [32].

Some systems maintain a matrix which stores *user ratings* on items. Each position of this matrix *(u,i)* contains the rating done to item *i* by user *u*.

The user profile representation can also consist in a list of *demographic features* which represent the kind of users. That kind of profiles is created through stereotypes which will be discussed in the next section.

3.1.2.2 Generating initial profiles

After defining how the user preferences are stored it is necessary to think about how to get the initial data for the profile in order to start using it in the recommender system.

Some systems allow starting to work with an *empty profile* structure which is filled through an automatic recognition method when the user begins to interact with the system.

A system can also require from the users a *manual registration* of their interests. This implies more work for the user and it is not accurate, since some interests may still be unknown at the moment the user starts to use the system.

The initial user profile can also be modelled by *stereotyping*. Users can be classified in stereotypical descriptions which represent the features of classes of users. This classification is done by using demographic data which the user may indicate in a registration form. This method is implemented in the LifeStyle Finder [36], which uses a commercially available database of demographic data which encompasses the interests of people nationwide. The main problem of this technique is that users usually provide incomplete or false data so the stereotyping is inaccurate in those cases.

Finally, another method for modelling the initial profile is using a *training set*. It contains a set of user interaction examples which are faced to the user, such as a list of products to rate. User interaction is then processed with a learning technique in order to generate the profile.

3.1.3 Automatic learning of preferences

As indicated before, the training set approach to generate an initial profile requires a *learning* technique to build a profile from the initial information of the user. Moreover, user preferences may evolve over time so profiles must be able to *adapt* to these changes in order to keep being useful. For both cases it is necessary to consider techniques which allow extracting user preferences from the interaction with the user when defining and when using the profile, distinguishing implicit and explicit interactions.

3.1.3.1 Learning techniques

Profile learning techniques build a user profile through data obtained by a training process. Two main techniques are used for preference learning: clustering and classifiers. These techniques can be seen as a preliminary step in representing a user profile.

The basic idea of the *clustering* technique is to group users that share similar information. Then, these clusters are matched against actual information to conclude whether it is interesting or not. For example, in ACR News [46] user transactions are mapped into a multi-dimensional space as vectors of URL references. This space is partitioned into clusters representing a group of transactions that are similar, based on co-occurrence patterns of URL references. Finally, clusters are matched against an active user session to recommend interesting URLs.

Classifiers are general computational models for assigning a category to an input. To build a recommender system using a classifier means using information about the item and the user profile as input, and having the output category represent how strongly to recommend an item to the user. Classifiers may be implemented using many different machine learning strategies including neural networks, decision trees, association rules and Bayesian networks.

Learning in *neural networks* is achieved by training the networks with a set of data. The input pattern is propagated forward through the layers of the network. When an error is detected, it is propagated backwards (backward-learning) in order to adjust the values of each cell of the network. As an example, Jennings and Higuchi [31] used a neural network to construct a user profile.

The goal of *decision tree* learning is to create a model that predicts the value of a target variable based on several input variables. A tree can be learned by splitting the source set into subgroups, obtaining source classes of items which can be classified, for example, into *interesting* and *not interesting* [37].

The *association rule discovery* methods initially find groups of items occurring frequently together in many transactions. Association rules capture the relationships among these items based on their patterns of co-occurrence across transactions. An example of an inductive learning technique is Ripper [12].

A *Bayesian network* (BN) is a directed acyclic graph where each node represents a domain variable, and each arc between nodes represents a probabilistic dependency, quantified using a conditional probability distribution for each node. Applying BN techniques to classification involves two sub-tasks [19]: BN learning (training) to get a model and BN inference to classify instances. First, a learner algorithm is applied to a set of training data, searching over various model structures in terms of dependencies for each item. In the resulting network, each item will have a set of parent items that are the best predictors of its votes. The model can be built off-line in a matter of hours. For this reason, this technique may prove practical for environments in which knowledge of consumer preferences changes slowly with respect to the time needed to build the model.

3.1.3.2 Relevance feedback

Relevance feedback [47] consists in the information resulting from the system-user interaction which can be analysed to update the user's profile. The two most common ways to obtain relevance feedback is to use information given *explicitly* by the user or to get it *implicitly* by observing the user's behaviour. Some *hybrid* systems combine implicit and explicit approaches.

Explicit feedback is obtained when users are required to explicitly evaluate items, indicating how relevant or interesting an item is to the user. There are three main approaches to get explicit relevance feedback: like/dislike, ratings and text comments:

a) In like/dislike systems, users explicitly evaluate items in a binary scale by classifying an object as "interesting" or "not interesting", as "relevant" or "not relevant" or as "like" or "hate", such as Websail [17].
b) The rating approach requires users to provide a judgement on a discrete scale, typically numeric or symbolic with a mapping to a numeric scale, such as Amalthea [49] or Google Stars [25].
c) The systems which gather user comments about items, present them to the users to facilitate the decision-making process. While textual comments

are helpful, they require the users to read and interpret to what degree it is positive or negative [24, 54].

Implicit feedback is obtained by monitoring the user's actions in order to automatically infer the user's preferences, following the idea that user's interests can be inferred from his behaviour [13]. Implicit feedback was defined by Rich [55] and first implemented by Mitchell *et al*. [45]. Since then, many systems have implemented implicit user feedback in their approaches and some systems have even combined it with explicit feedback providing hybrid solutions. Most implicit methods obtain relevance feedback by analysing the links followed by the user, a history of purchases, the navigation history, e-mail boxes, the time spent reading a web page and the parts quickly "scrolled down"[34] and even the gaze direction and head pose of the user when consulting content [2].

As noted above, implicit feedback can be combined with explicit feedback systems in a hybrid system. Using implicit feedback decreases the user's efforts while the explicit feedback helps the system to infer user preferences accurately. One approach with this combination involves using implicit data as a check on explicit ratings [50]. For instance, if a user is explicitly rating an item, then there should be some implicit data to confirm that he has actually examined it. If there is no evidence that suggests this, then perhaps its rating should be ignored or reduced in importance. More examples of applications of these techniques can be found in areas such as text retrieval [66] and image retrieval [63]. The tourism recommender system 'Turist@' developed in the ITAKA research group is also an example of hybrid feedback management: the system extracts explicit information from the user evaluation of tourist activities and implicit information from the searches the user performs [7].

3.1.3.3 Techniques for retrieving relevant information

Feedback information obtained from the user interaction sometimes has no structure (*e.g.* plain text) so it is necessary to previously process it through

retrieval techniques to obtain relevant information. This process is typically done in two steps: feature selection and information indexing.

Step 1) *Feature selection* can be achieved through different approaches which reduce the numbers of words: stop-words, pruning, stemming, etc. [57].

Step 2) *Information indexing* uses the frequency word occurrence to calculate the potential relevance of an item. Term frequency-inverse document frequency (TF-IDF) is one of the most successful and best-tested techniques. It is a statistical measure [33] used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

3.1.3.4 Adaptation techniques

User interests cannot be assumed to stay constant for the whole period of time in which the user interacts with the recommender system. Adaptation techniques to maintain the profiles up to date have to be implemented in the recommender system. Those techniques will keep updated the information represented in the profiles by evaluating the relevance feedback obtained from user interaction.

Several approaches for profile adaptation indicated in [47] can be found on current recommender systems distinguishing between the manual and automatic techniques.

Recommender systems depending on the *manual adaptation* of the profiles require that the user changes his profile manually when he is interested in updating it. An example of it is SIFT Netnews [69], in which when a user wants to include/exclude one of the interests contained in his profile, he has to modify it manually. This approach has the same problems as the manual initial profile generation: it requires an effort from the user and, furthermore, people are not able to specify accurately their own interests because sometimes they are unknown.

Automatic techniques include techniques to add/forget information to/from the profiles. *Adding new information* is the most commonly used technique in current systems. The idea is to update the user profile by adding new information extracted from the user relevance feedback. The main problem of this is that old interests are not forgotten.

Implementing a *gradual forgetting function* allows the recommender system to eliminate old interests from user profiles (Webb and Kuzmycz [62]). They suggest a data aging mechanism which places an initial weight of 1 on each observation. A set proportion discounts the weight of every observation each time another relevant observation is incorporated into the model. So, the most recent observations become more "important", assuming they better represent the current users' interests than the older ones. A particular case of this technique is the *time window* approach: it consists on learning the description of the user's interests from only the latest observations.

3.2 Information of the user profile

Before detailing how the adaptation processes are conducted, the decision about how the profiles are structured and represented is explained, as well as the generation of the initial profiles.

Although the platform is able to store and manage numeric attributes, the adaptation of that kind of variables is still under development. We decided to implement and test the adaptation stages only with linguistic variables, which was the initial objective. Adaptation of numeric values is the next step to take after finishing the current work and some initial research in this area has already been done, as explained at the conclusions.

For each variable, the profile contains the necessary information to discover what the user interests in that criterion are.

User preferences over numeric variables are based in a single numeric value of preference, which we have previously referred to as v_{pref}, with $v_{pref} \in [v_{min}, v_{max}]$ (see section 2.5.2). Depending on how that preference value is chosen, there exist three possible cases of giving a preference about a numeric variable:

a) a *minimisation* interest, in which $v_{pref} = v_{min}$,
b) a *maximisation* interest, in which $v_{pref} = v_{max}$, and
c) an *intermediate* interest, in which $v_{min} < v_{pref} < v_{max}$.

Linguistic variables require a valuation for each of the possible linguistic values a single variable can take. This valuation is given in a linguistic way, using a preference label as seen in section 2.6.

Table 1 shows an example of a user profile describing interests about tourist destinations. Numeric attributes involve the distance to the nearest airport, the population density, and the terrain elevation. Linguistic attributes are the continent in which the tourist destination is found, its climate, the official language that is spoken and the type of tourism that can be done there.

Numeric criteria	**Preference value**
Nearest airport distance	0
Population density	1000
Elevation	200
Linguistic criteria	**<attribute,preference>**
Continent	<Africa, VL>, <Oceania,M>, <America,VH>, <Asia,L>, <Europe,H>
Climate	<Moderate, VH>,<Dry,VL>, <Tropical,VL>, <Continental,L>
Language	<English, VH>, <Spanish, VH>, <Chinese,VL>, <Japanese, VL>, <French, L>, <Indigenous,VL>
Type of tourism	<Sightseeing,M>, <Adventure,VH>, <Rural,M>, <Beach,H>

Table 1: Example of user profile for tourist destinations

The initial profile can be defined by the user *manually* or use an *empty* one that will be learnt through the adaptation processes. Obviously, even if the user chooses to introduce manually his preferences, they can be refined and updated through its utilisation with the recommender system.

3.3 Adaptation of the user profile

As said before, in order to maintain an updated and accurate profile of a user, it is necessary to extract important information of the feedback generated from his interaction with the recommender system. Our unsupervised adaptation approach extracts this information from the set of alternatives that were ranked and shown to the user when using the ranking module of the recommender system. Two units of information are extracted from here: the selected alternative, which is the final decision of the user, and the set of alternatives that were ranked above the selected one. The main idea is that the adaptation process must be able to internally find an answer to questions such as "why are those alternatives better ranked that the one the user really wants?" or "why the alternative the user really wants does not have a score good enough to be ranked in the first place?".

The profile adaptation is conducted by two processes. The first one, called *on-line adaptation process*, is executed every time the user asks the system for a recommendation, and just evaluates the information that can be extracted of the current ranked alternatives. The second one, called *off-line adaptation process*, is triggered after some user recommendations have been served, and uses the information of the previous ranked alternatives and selections.

Adaptation processes modify the user profile by increasing or decreasing the level of preference of the attribute values according to the reasoning explained below. The algorithm accumulates evidences to update all values, decides which values should be updated, and finally changes them by performing the adaptations, which means adding or subtracting one label of S to the preference

values in the profile. For instance, a value with the label *high* can change by increasing it to *very-high* or decreasing it to *almost-high*.

3.3.1 Relevance feedback

In order to adapt the user profile we considered that, for our purpose, the best option is to obtain the relevant information to update the preferences in an *implicit* way.

Our system ranks and shows to the user all of the alternatives according to his preferences, waiting for the user to make the final decision, which can be the first alternative or any of the following ones. If the user selects the first one, it means that the recommendation was accurate and user preferences are correctly expressed in the profile, so no adaptation is needed. However, if the user selects other alternative, it means that preferences in the profile are not accurate enough. This last case triggers the adaptation processes which will extract information from the alternatives and the user selection in order to adapt the profile.

The adaptation of the user profile is done with the information collected in the current selection. The current selection and the discarded options give relevant information to infer which criteria can be changed and its sense. However, we also noticed that the information about past selections provide valuable information to corroborate or not a change. These two behaviours are explained in the following. The former situation is called *on-line adaptation process*, while the second is called *off-line adaptation process.*

3.3.2 On-line adaptation process

The *on-line adaptation process* tries to maintain the user profile updated by evaluating each recommendation and the final decision the user makes individually.

As previously said, for each recommendation the system makes, two sources of information are evaluated: the selected alternative, which is the final

decision of the user, and the alternatives that were ranked above the selected one. Many conclusions can be extracted by evaluating this information. Those conclusions are found by extracting *characteristics* from the available data. A characteristic is a tuple composed by a linguistic attribute, a concrete label from the set of the possible values of said linguistic attribute, and the times it is repeated among a concrete set of alternatives. It is necessary to have a minimum number of alternatives to allow the extraction of characteristics since characteristics extracted from a reduced number of alternatives do not give useful information.

The main objective of this stage is to decrease the preference of the attribute values that are causing non desired alternatives to be ranked with higher scores, and also to increase the preference of the attribute values which are important for the user but are not well judged on the current user profile.

We call *over ranked characteristics* the features extracted from the set of alternatives that were ranked above the user final selection. In order to obtain relevant characteristics, a *repetition threshold* is defined: just the characteristics whose repetition value is equal or over that threshold will be considered. In our tests, we used a threshold defined as a percentage on all the over ranked alternatives.

Moreover, there is another threshold that determines that if there are not enough over ranked alternatives the system cannot extract any characteristic.

Finally, an over ranked characteristic is only taken into account if it does not appear on the selected alternative, since we consider that a characteristic that was in the selection cannot have his preference decreased.

Consider the set of alternatives of tourist destinations shown in Table 2, in which the sixth alternative is the one selected by the user, and a repetition threshold of 40%. The tuple (‘Continent’, ‘Asia’, ‘2’) is an example of over ranked characteristic, being ‘Continent’ the attribute name, ‘Asia’ a possible

attribute value, and ‘2’ the times it appears among the set of over ranked alternatives. Note that (‘Continent’, ‘Africa’, ‘1’) cannot be considered as an over ranked characteristic since it only appears once, thus it just represents a 20% of the over ranked alternatives (1 out of 5). (‘Tourism type’, ‘Sightseeing’, ‘2’) is not an over ranked characteristic either, since this value appears on the selected alternative.

Rank	***Distance to the nearest airport***	***Continent***	***Climate***	***Official language***	***Tourism type***
1	Far	Europe	Continental	English	Rural
2	Far	America	Continental	French	Rural
3	Very Far	Asia	Dry	Chinese	Adventure
4	Very Far	Asia	Dry	Japanese	Sightseeing
5	Near	Africa	Tropical	French	Sightseeing
6	**Very Near**	**Europe**	**Moderate**	**Spanish**	**Sightseeing**
7	Far	America	Moderate	English	Beach

Table 2: Example set of alternatives

Over ranked characteristics are used to *decrease* the level of preference of those attribute values pointed out by the characteristics. The intensity of that decrease is regulated by the number of repetitions: if a characteristic is repeated many times, we have more evidence to decrease the preference on that attribute value. More information about the intensity of that decrease is discussed forward in this document, in section 3.3.4.

We call *selection characteristics* the features extracted from the user final selection that do not appear on the set of over ranked alternatives more than a given number of times. As in the over ranked characteristics, a repetition threshold is also defined here, but its use is opposite as when extracting over ranked characteristics: if the characteristic appears less than that threshold in the over ranked characteristics, it is considered a selection characteristic. To compensate the extraction of over ranked characteristics with the extraction of the

selection characteristics, the threshold used here is the same as the one used for over ranked characteristics extraction.

The adaptation process only considers as selection characteristics the ones which appear on the over ranked alternatives equal or fewer times of said threshold. Following the values of Table 2, the data structures (‘Distance to the nearest airport’, ‘Very near’, ‘0’) and (‘Continent’, ‘Europe’, ‘1’) are two examples of correct selection alternatives. It is important to notice that while an over ranked characteristic cannot appear on the selection, a selection characteristic can appear among the over ranked characteristics.

Following a similar procedure, selection characteristics are also used to *increase* the level of preference of the attribute values pointed out by the characteristics. The intensity of that increase is greater the fewer times that characteristic appears among the over ranked characteristics. In other words, using the previous example, we have more evidence to increase the preference of the value ‘Very near’ of the attribute ‘Distance to the nearest airport’ than the one of the value ‘Europe’ of the attribute ‘Continent’, since the first does not appear among the over ranked alternatives and the second appears once. Again, that question will be discussed more deeply in section 3.3.4.

3.3.3 Off-line adaptation process

The on-line adaptation process can give an immediate response to a single user interaction with the recommender system. However, there are cases in which the information extracted from a single interaction is not enough to gather the sufficient amount of evidence to make changes in the profile. In those cases, data from past recommendations is stored and accumulated to be analysed later. The *off-line adaptation process* manages over ranked characteristics and selections differently as is explained below.

We defined the minimum number of over ranked alternatives as a requirement to start extracting characteristics. However, this causes the following

problem: when the profile is near to the ideal profile, the user selection is among the first ranked alternatives, so there are not enough over ranked alternatives to analyse. This leads to the system being unable to determine which preferences it has to increase or decrease to reach the ideal profile. A solution to this problem can be to store those small pieces of data and evaluate them as a whole when there is enough information. Since this information comes from different recommendations, it is not appropriate to compare over ranked alternatives with the selected alternatives as in the on-line process. However, evidences for preference decrease can be extracted by evaluating the over ranked alternatives separately, and evidences for preference increase can be extracted by evaluating the selected alternatives separately.

So, when the system faces cases when the number of over ranked alternatives is not enough to extract reliable characteristics, it stores that small number of over ranked alternatives into the database. After some cases of insufficient number of over ranked alternatives to evaluate, the system gathers enough alternatives to start evaluating them. When the number of saved over ranked alternatives is adequate characteristics are extracted from that set of accumulated alternatives. Those characteristics are used to decrease the preferences in the same way as it is done in the on-line process: characteristics with a repetition value over the defined threshold are decreased. After the stored over ranked alternatives have been treated, they are erased from the database. The number of stored over ranked alternatives to reach before evaluating them is the same than the one used in the on-line process to determine if there are enough over ranked alternatives to extract characteristics.

Figure 9 exemplifies three consecutive recommendations. Considering that the minimum number of over ranked alternatives to extract over ranked characteristics is ‘6’, the on-line adaptation process is not able to extract any over ranked characteristic. The off-line process, however, stores after each iteration the over ranked alternatives that the on-line process cannot manage and, after the third

recommendation, it has enough alternatives to extract over ranked characteristics an deduce possible adaptations.



Figure 9: Example of recommendations

On the other hand, user selections can be stored and, after a certain number of choices have been made, they can be evaluated. By extracting characteristics from a set of stored previous selections, the most repeated attribute values, which can be interpreted as the ones of most interest to the user, can have its preference increased as explained in the next section. After evaluating them, they are erased from the database. The number of selections stored before evaluating them is another parameter of the algorithm. If that number of selections is set to '3' in the example of Figure 9, after the third iteration this part of the off-line process will be executed.

3.3.4 Regulation of the adaptation mechanisms

After a single recommendation and the posterior selection of the final decision of the user, many characteristics can be found and the system can deduce a large amount of changes to do in the profile. Many of those changes or adaptations, although deduced using the reasoning techniques defined before, can be incorrect.

We have addressed this problem by restricting the number of adaptations that can be made to the profile for each execution of the adaptation process. A parameter limits the number of increases and decreases which can be made to a profile per adaptation step (a value of '2' allows a total of 4 changes: 2 increases

and 2 decreases). When a large number of possible adaptations are being considered, only the more evident ones are performed. That 'level of evidence' is measured differently for the decreases and for the increases of preference.

For the decreases (also known as negative adaptations) in the on-line and off-line adaptation process, the level of evidence is calculated by counting the number of the repetitions of that characteristic: the ones with more repetitions will be more likely to be decreased.

For the increases (also known as positive adaptations) it is calculated in a different way in the on-line and in the off-line adaption processes. In the on-line process, the 'level of evidence' of a characteristic is calculated by counting the number of over ranked alternatives which do not have that characteristic, so the ones with a greater counter will be increased. In the off-line process, since there are no over ranked alternatives directly related to the selections, that 'level of evidence' is calculated by counting how many times a selection characteristic appears among the set of stored selection characteristics.

This mechanism reduces the errors that can be done by increasing/decreasing wrong preferences, although it may be inefficient when those levels of evidence are equal among many possible adaptations, including the wrong ones. Our approach to solve this problem involves a *signed counter* which regulates the final increase/decrease of the preference of the profile. When the adaptation process detects an evidence for a possible adaptation, it increases/reduces a counter, initially set to 0, associated to that attribute value (for example the 'Europe' value of the attribute 'Continent'). Positive adaptations increase the counter while negative adaptations decrease the counter.

The amount increased or decreased is the 'level of evidence', calculated as explained before. When the counter reaches a certain value defined by a parameter, the increase/reduction is finally performed over the preference in the profile and the counter is reset to 0. Considering said parameter as '+/-5' and following the example of Table 2, the system finds out that the 'Europe'

preference value of the attribute 'Continent' needs to be increased so it increases its counter by '4', since it only appears once among the over ranked alternatives. So, the counter does not reach the necessary number to increase the preference value on the profile and it will be needed another positive evidence to finally change it.

This mechanism was motivated by the observation of the fact that the great majority of erroneous adaptations have low levels of evidence, so with this mechanism several consecutive erroneous evidences need to be treated before performing a wrong change in the profile.

3.4 Conclusions

In this chapter we have explained how preferences have been managed and used by different authors, including the types of profile that exists, their generation and their learning and adaptation. We have also pointed out which of these techniques we are using or based our work in.

Our profile adaptation approaches explained in this chapter have been described and implemented relying on parameters which tune their behaviour, adding flexibility to their use depending on the context the adaptation needs to be made. The next section develops a study on how these parameters affect the final adaptation result and how they should be tuned in order to obtain fast and accurate changes in the profile.

Moreover, a paper explaining how our system deals with linguistic variables when making recommendations and how they can be adapted has been submitted to the "5th Multidisciplinary Workshop on Advances in Preference Handling" within the "19th European Conference on Artificial Intelligence" which will be celebrated in Portugal in August 2010. A further study on the overall performance of the system and in particular the adaptation processes has been submitted to the "13è Congrés Internacional de l'Associació Catalana d'Intel·ligència Artificial" which will be celebrated in Tarragona in October 2010.

4 Evaluation

Previous chapters introduced the rating and ranking process prior to the implicit adaptation of the user profile. These processes have been implemented in a Web-based simulator that eases the analysis of all parameters according to the domain used.

The first section consists in a brief explanation of the testing platform developed including its main functionalities as well as the evaluation algorithm followed in the testing phase. The second section describes the testing environment used to evaluate the platform as well as the measures to appraise the adaptation results, consisting in measuring the distance between the profile currently being adapted and the ideal profile. The last section discusses about parameters which tune the performance of the adaptation mechanisms.

4.1 Testing platform

In order to test and evaluate our recommender method, a Web platform[1] has been implemented. Its main functionalities are:

- Definition of a recommendation problem (set of user criteria, aggregation and adaptation parameters).
- Definition of an initial profile.
- Definition of an ideal profile that we aim to learn.
- Upload of a data file and/or random generation of a corpus of alternatives for the problem.
- Simulation of tests of user interaction with the system.
- Visualisation of the evolution of the user profile.

The access to the platform is regulated by a user name and a password, allowing the maintenance of multiple users with different profiles for each recommendation domain. This approach permits to analyse the variability of results under different circumstances in the aggregation and adaptation stages.

Users can upload data files with information about the alternatives that must be analysed in order to make a decision. The platform stores that information in a database, letting users to define a decision making problem on those alternatives.

[1] Website: http://itaka2-deim.urv.cat/recommender. Last access: June 1st, 2010.

4.2 Testing algorithm

Figure 10 shows the pseudo code of the algorithm that permits an automatic validation of the adaptation mechanisms by simulating how the system adapts an initial random profile using the information extracted from choices calculated using an ideal profile. The platform also permits a real user to perform the selection.

The algorithm requires several parameters such as the corpus of alternatives initialised *a priori*, some thresholds analysed in detail in the next subsections, and the profiles to run a simulation:

a) Corpus of testing alternatives ($E(a_0,\ldots,a_{max})$).
b) Level of evidence required to change a preference (k).
c) Threshold required for extracting characteristics (t).
d) Minimum number of over ranked alternatives to extract characteristics (or).
e) Maximum changes of preference per recommendation (pc).
f) Alternatives treated at each iteration (ai).
g) Number of iterations to simulate ($MaxIter$).
h) Ideal profile (I).
i) Initial profile to be adapted (P).

At each iteration, the first thing to do is obtain the alternatives that will be treated from the whole corpus of alternatives. This subset of alternatives is limited by the parameters *beg* and *end* calculated in lines 15 and 16 of the pseudo code. Afterwards, this subset of alternatives is rated and ranked using the current profile (P) in line 17. Then, the same subset of alternatives is rated and ranked using the ideal profile (I) in order to obtain the selected alternative u, which is the alternative of R which a user with a profile I would choose (line 18).

```
1:  ADAPTATION-ALGORITHM(
2:    E(a0,...,amax), //corpus of alternatives
3:    k, //level of evidence
4:    t, //characteristics extraction threshold
5:    or, //minimum number of required over ranked alternatives
6:    pc, //maximum preferences changes threshold
7:    ai, //alternatives per iteration shown
8:    MaxIter, //iterations to simulate
9:    I, //ideal profile
10:   P //initial profile)
11:   begin
12:     iter,beg,end=0;
13:     ovr, sel=∅;
14:     while(iter<MaxIter) do
15:       beg=iter*ai;
16:       end=(iter+1)*ai-1;
17:       R(a'0,...,a'ai-1)=rating-and-ranking(P,E(abeg,...,aend));
18:       u=calculate-ideal-selection(R,I);
19:       sel=sel U R(a'u);
20:       if (u>0) then ovr=ovr U R(a'0,...,a'u-1);
21:       if (ovr>=or) then evidences=on-line(P,R(a'u),R(a'0,...,a'u-1),t);
22:       evidences=evidences U off-line(P,sel,ovr,t);
23:       P=adaptation(P,evidences,pc,k);
24:       iter=iter+1;
25:     end while;
26:   end;
```

Figure 10: Pseudo code of the simulator

The current selection $\left(R(a'_u)\right)$ and the set of over ranked alternatives $\left(R(a'_0, \ldots, a'_{u-1})\right)$ are stored and used to find the evidences in the on-line adaptation step: first, the over ranked characteristics are extracted (line 20) and then, if there are enough, the on-line process treats them (line 21). Evidences count if attributes should be increased or decreased. The off-line stage complements the on-line one by evaluating past selections and over ranked alternatives when required (line 22). Finally, if there are enough evidences to modify the profile, the adaptations are performed over the initial profile (*P*) (line 23).

In the next section, a study of the execution of this algorithm is studied and the steps followed to evaluate a concrete set of alternatives are detailed. Moreover, a more elaborate explanation about the platform functionalities and utilisation is given in the platform manual in appendix A.

4.3 General performance

In order to evaluate the framework, we propose to iteratively measure the distance between the current user profile and the ideal profile that we want to learn. Iteratively and automatically, the system selects the alternative that best fits the ideal profile, which is the one that the user which we want to have its profile learnt would choose. Then, the selection and the over ranked alternatives are used by the adaptation processes to evaluate which changes should be performed on the current profile. To compare the results of different tests, we initialize the user profile randomly, but we maintain the ideal user profile. Distances between both profiles decrease over time.

4.3.1 Testing environment

Two set of tests have been made using two corpus of 3000 alternatives initialised randomly, with blocks of 15 alternatives per iteration. The first corpus of alternatives contains information about cities (as the example used before in this document) generated randomly, considering the criteria and values indicated in Table 3. Each of those alternatives represents a possible tourist destination in which the decision maker needs aid choosing one.

The second one contains information about medical centres, randomly generated, considering the criteria and values indicated in Table 4. In this case, the decision maker needs a specialist medical visit and needs help deciding which doctor is the best to meet for him.

Criterion	Values
Distance	<Very far, Far, Near, Very near>
Continent	<Africa, Oceania, America, Asia, Europe>
Climate	<Moderate, Dry, Tropical, Continental>
Language	<English, Spanish, Chinese, Japanese, French, Indigenous>
Type of tourism	<Sightseeing, Adventure, Rural, Beach>

Table 3: Linguistic criteria possible values for describing tourist destinations

Criterion	Values
Distance	<Very far, Far, Near, Very near>
Doctors evaluation	<Excellent, Very high, High, Normal, Bad, Very bad>
Day of week	<Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday>
Time	<Morning, Afternoon, Night>
Parking	<No, Free, Blue zone>

Table 4: Linguistic criteria possible values for describing medical centres alternatives

Two kinds of profiles have been initialised for both tests in order to obtain comparable results, using the term set shown in Figure 11 (9 unbalanced labels): an ideal profile for each test, in which the user ideal preferences are indicated, and three initial profiles generated randomly, whose objective is to be adapted until they are very similar to the desired ones.

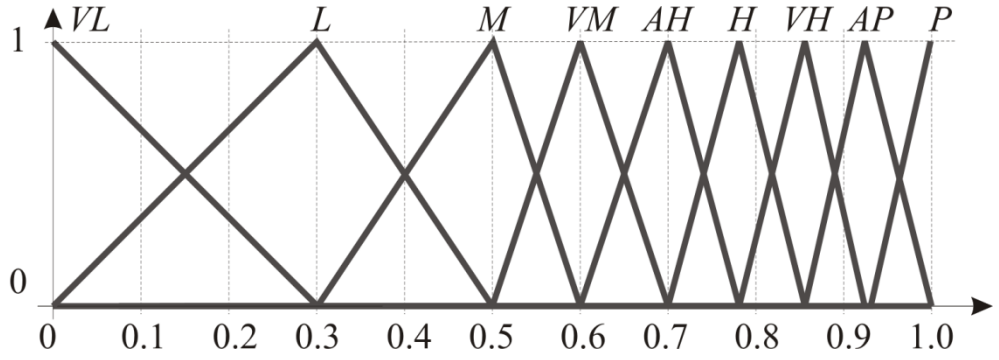


Figure 11: Fuzzy term set *S* with 9 unbalanced labels

As formalised in the algorithm of the Figure 10, each iteration step of the evaluation process with the corpus of alternatives defined above consists in the following points:

1. Obtain a block of 15 alternatives from the corpus of 3000 alternatives of the domain.
2. Rank those 15 alternatives using the current profile.
3. Rank the same 15 alternatives using the ideal profile.
4. Store as the user selection the first one ranked alternative in step 3.
5. Store as over ranked alternatives the ones ranked above the selection in step 2.
6. Execute the on-line and off-line adaptation processes (off-line just if required).
7. Calculate and store the distance between the ideal profile and the current profile.

Although we will analyse the influence of the main parameters later, these tests have been performed using the extraction characteristics threshold of 30%, with the level of evidence '+/-5' to consider a change in a value, and allowing only '2' changes in each sense in one iteration. Under these circumstances, we analyse the distance between the current (*P*) and the ideal (*I*) profiles, which is calculated as follows:

$$dist(I,P) = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{1}{m_j}\sum_{k=1}^{m_j}\frac{\left|COG_x\left(v(P,j,k)\right) - COG_x\left(v(I,j,k)\right)\right|}{\left|COG_x(s_0) - COG_x(s_r)\right|}\right) \quad \textbf{(3)}$$

Here, n is the number of attributes, m_j is the cardinality of the attribute j, COG_x evaluates the coordinate x of the centre of gravity of a linguistic label [15], v is a function that returns the linguistic value of a profile, an attribute and a given value. With this function we obtain an average of the distances between all pairs of labels. The distance is normalised by using the extreme values of the fuzzy set $S = \{s_0, \ldots, s_r\}$. The distance is 0 when both profiles are identical. The maximum distance is 1 when all the values contained in the user and the ideal profiles have just the opposite labels (s_0 and s_r). Moreover, this function is commutative ($dist(P,I) = dist(I,P)$).

4.3.2 Performance on-line and off-line adaptation processes

Figure 12 and Figure 13 compares the general performance of the adaptation algorithm using only the on-line process and using the on-line combined with the off-line stage, by means of the profile distance calculated using the equation (3). This test has been done by using the tourism data.

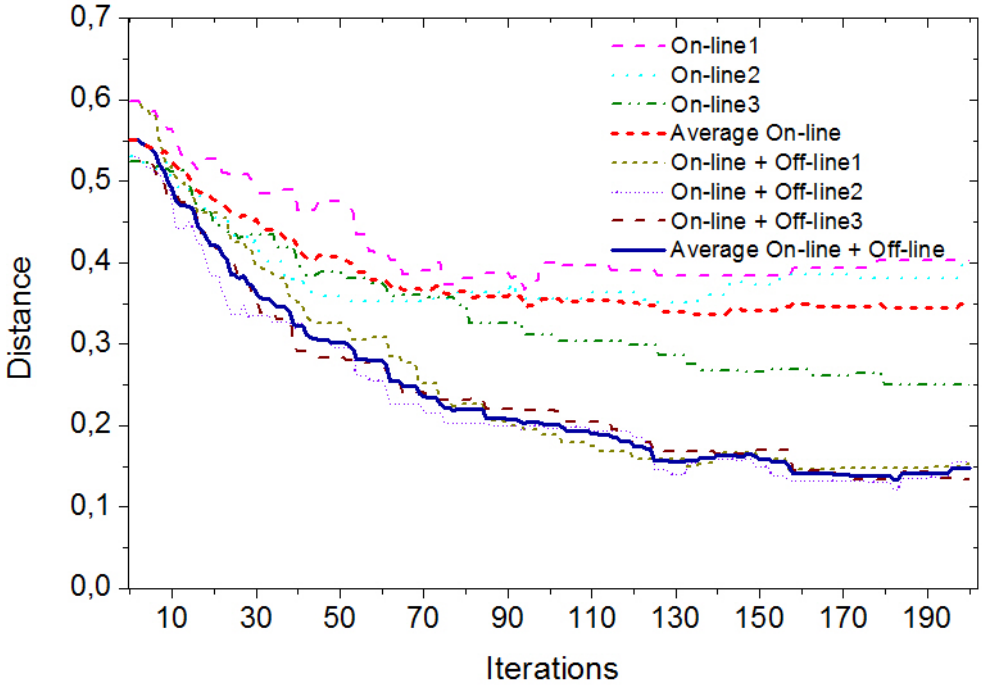


Figure 12: Profile distances with on-line and off-line methods (tourism)

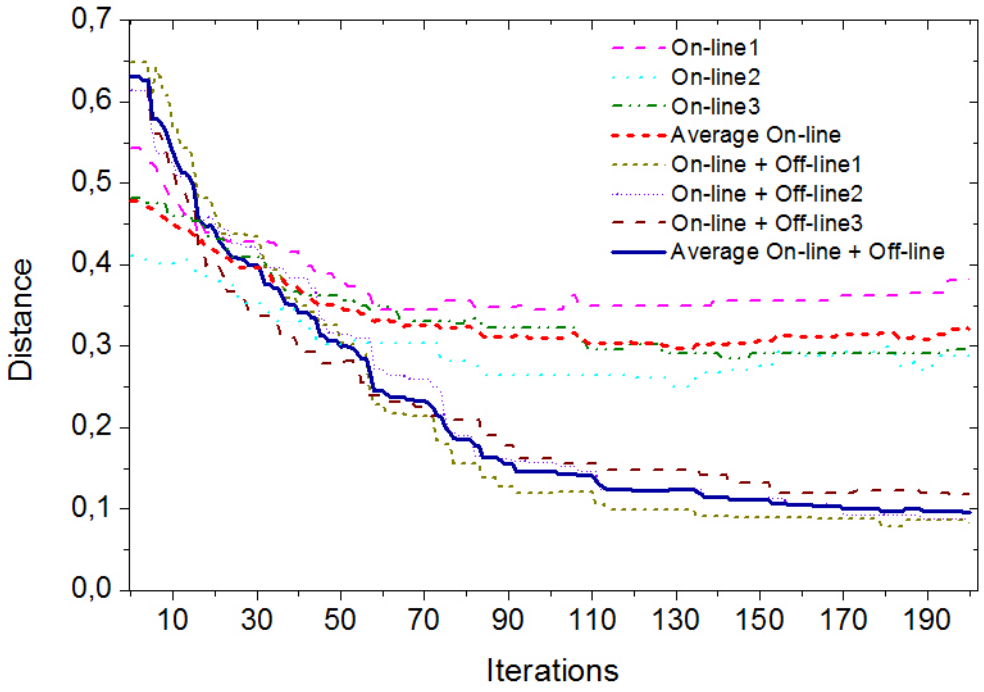


Figure 13: Profile distances with on-line and off-line methods (medical centres)

The figure shows the distances between the current and the ideal profile of three examples of each case and their average through 200 iterations. The three examples have different initial profiles but share the same ideal profile. The general performance accomplished when both processes are working together to adapt the profile is much better than when just using the on-line process. The parameters used in this simulation are the ones which gave the best results in the parameter's analysis conducted in the following section. It is clear how using the off-line adaptation process enhances the result obtained than when just using the on-line adaptation process.

4.3.3 Number of labels of distance

Equation (3) gives a general evaluation of the performance of the adaptation algorithm. However, it is also interesting to see the differences between profiles by examining the number of labels of distance between the current and the ideal label for each value of each criterion: for instance, if an attribute of the current profile is *H* and the ideal is *P*, the number of labels of distance is 3.

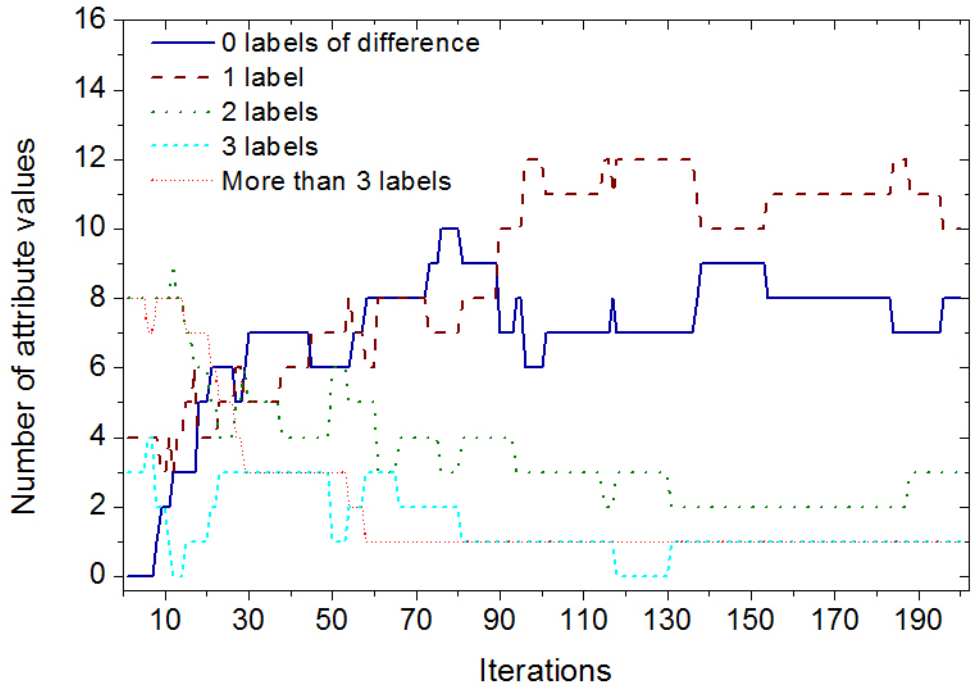


Figure 14: Number of labels of distance

Figure 14 focuses the study on one simulation, which corresponds to the "On-line + Off-line 1" test in Figure 12, and shows the labels of distance between the values of current and ideal profiles. The figure compares how many attributes there are with difference 0 (correctly classified), 1, 2, 3 and more than 3 labels. It can be seen how the number of the most misclassified labels (distance 3 and more) decreases drastically, while the set of well classified attributes (distances 0 and 1) increases. At the end, 19 of the 23 attribute values have a preference label that is at most at distance 1 of the label for that attribute value in the ideal profile.

4.4 Analysis of the parameters

The proposed implicit adaptation algorithm introduces several parameters that should be properly customised. This section explains the influence of these parameters in the final result, although exact parameter values may vary depending on the context in which the adaptation is taking place (number of attributes, number of values per attribute, linguistic label set used, etc.). We insist that we are not conducting those tests to find the ideal value for each parameter

but to find how the final result is affected by decreasing or increasing the value of said parameters.

Each parameter value has been tested in the adaptation of the three randomly generated initial profiles indicated in the previous section. The average of the distances obtained through those three tests is the result shown in the graphics below.

4.4.1 Characteristics extraction threshold

Over ranked characteristics are extracted by considering a threshold defined as a percentage over the number of over ranked alternatives. Figure 15 and Figure 16 (for the tourism and medical centres tests respectively) show that after several tests (using the threshold values of 15%, 30% and 50%) a threshold of 30% was the one that got better results in both cases.

Having a greater threshold implies that the extracted characteristics need to have a greater repetition value, thus fewer characteristics are extracted, reducing the number of possible changes in the profile. On the contrary, the extraction of valuable information is compromised as the threshold decreases, because incorrect characteristics (which usually have a low number of repetitions) are extracted more easily.

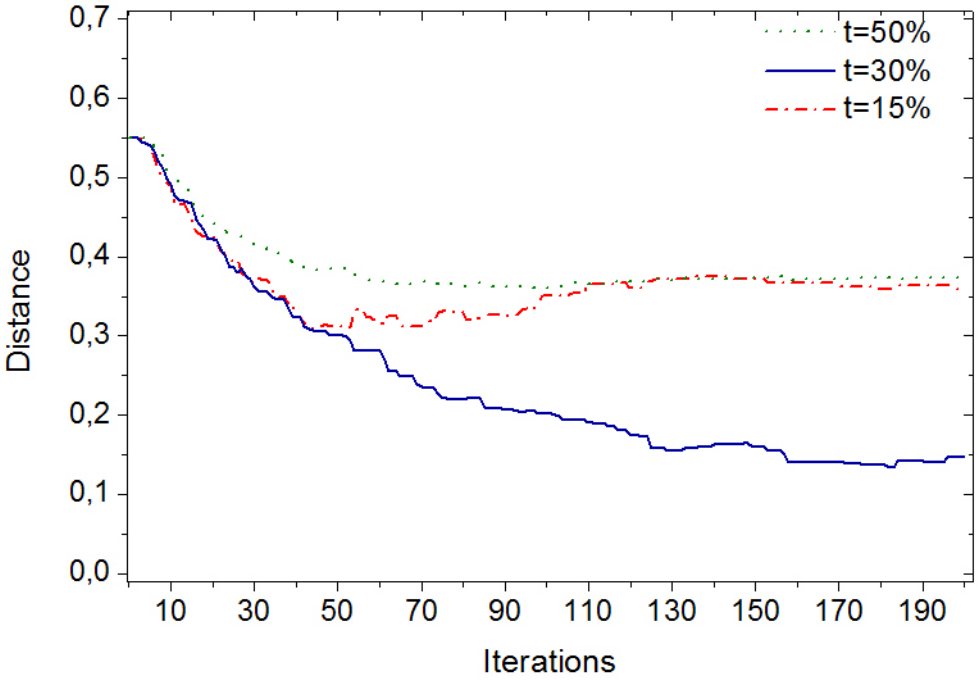


Figure 15: Influence of the characteristics extraction threshold (tourism)

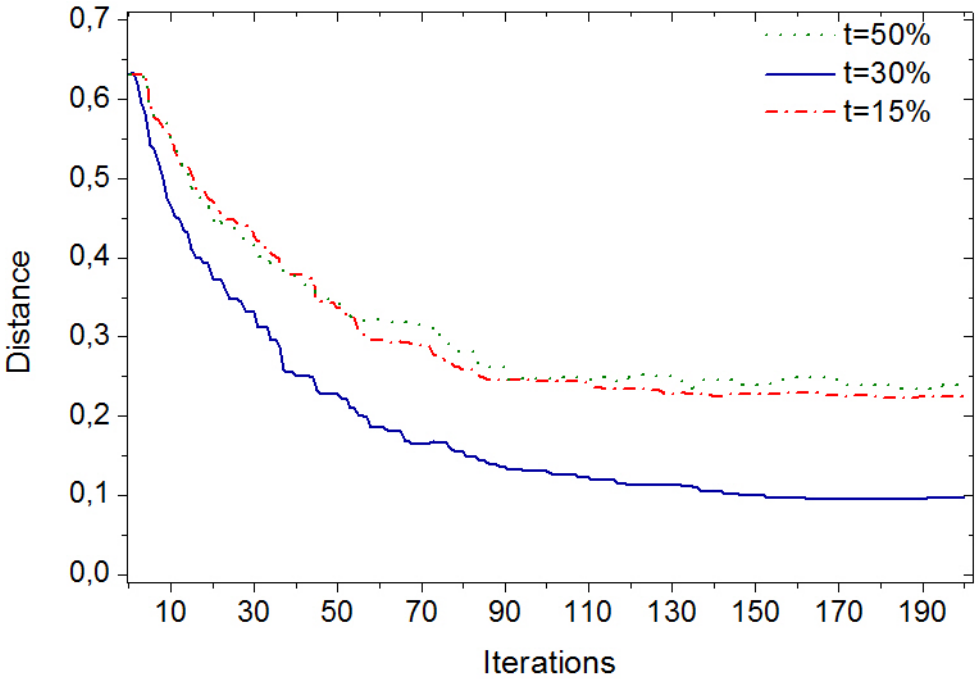


Figure 16: Influence of the characteristics extraction threshold (medical centres)

4.4.2 Number of stored selections

During the interaction with the platform, the selections the user does are stored in the database for being processed by the off-line adaptation process after a certain number of selections have been made. Figure 17 depicts the influence of the variation of that parameter in the adaptation process.

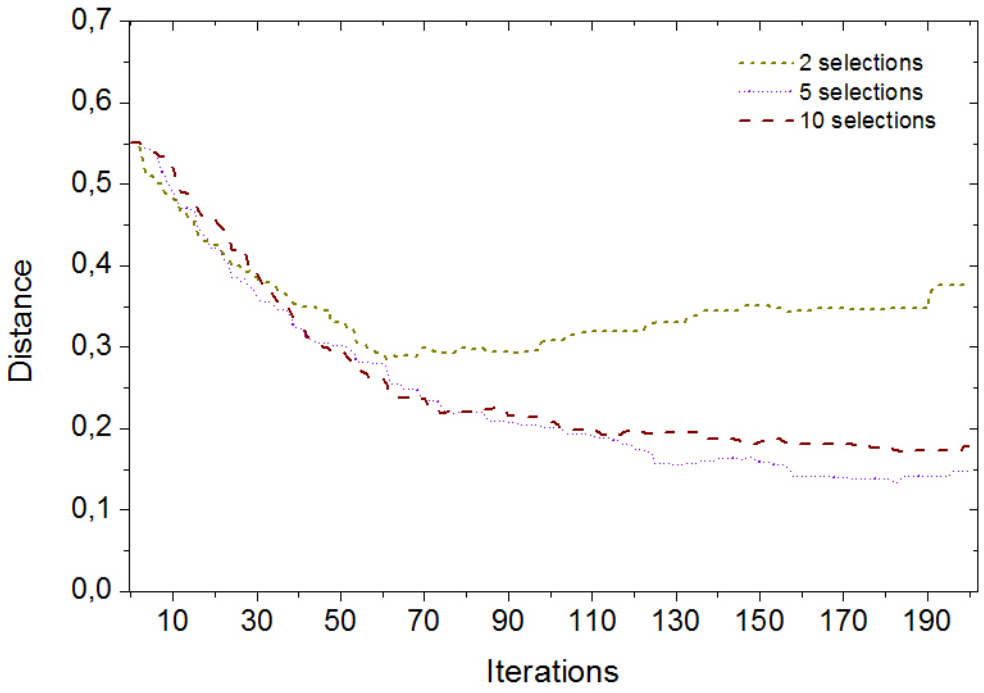


Figure 17: Influence of the number of stored selections (tourism)

It can be noted that with a low value of stored selections, such as 2, a bad result is obtained. It is because if the off-line adaptation process does not process enough selections, not very reliable characteristics are extracted, so the performance of the whole adaptation is similar to the one one studied when the off-line process was deactivated (see Figure 12).

4.4.3 Number of changes in the profile

On the adaptation mechanism section, a parameter for controlling how many increases/decreases of preference can be done at each adaptation process iteration was introduced. Figure 18 and Figure 19 compares the performance of the system with different values for that parameter.

Setting that value to '2' proved to give slightly better results than using a lower or greater number, letting the system do up to two increases and two decreases on any attribute value preference of the profile.

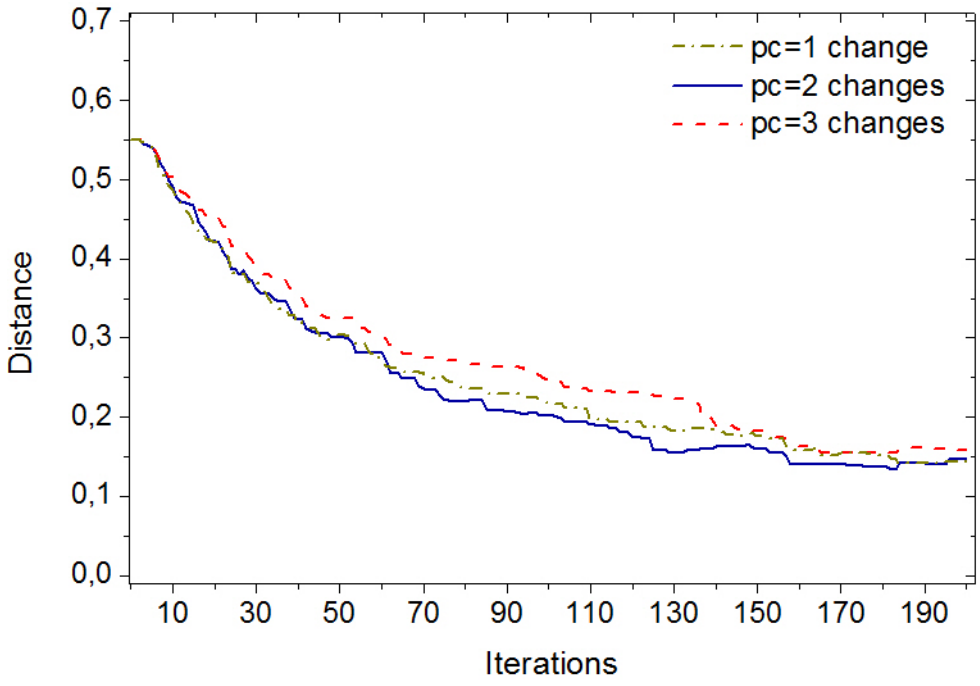


Figure 18: Influence of the number of preference changes (tourism)

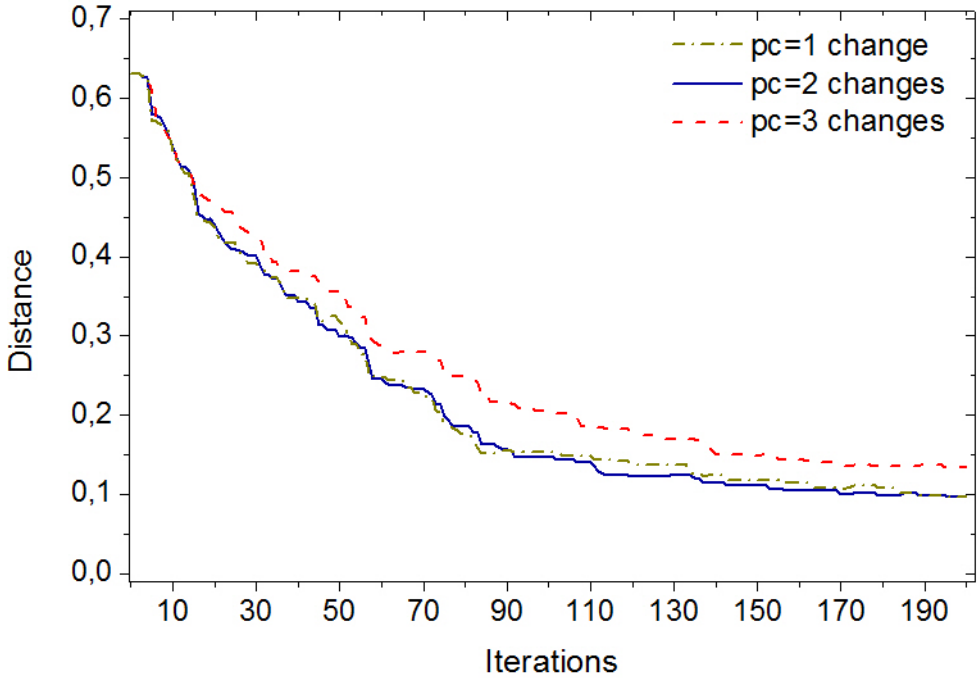


Figure 19: Influence of the number of preference changes (medical centres)

4.4.4 Evidences required to change a preference

Section 3.3.4 also introduces the parameter that varies the necessary number of evidences to increase/decrease a preference on the profile. In this case, with lower values, the number of required evidences is lower, and the quality of these changes is also compromised.

On the other hand, higher values for this parameter make it difficult to change preferences, so it requires more iterations (selections) to reach a near-ideal profile. Those differences, however, tend to decrease as the number of iterations increases. Figure 20 and Figure 21 show these behaviours and how an intermediate value such as '+/-5' exhibits a good performance.

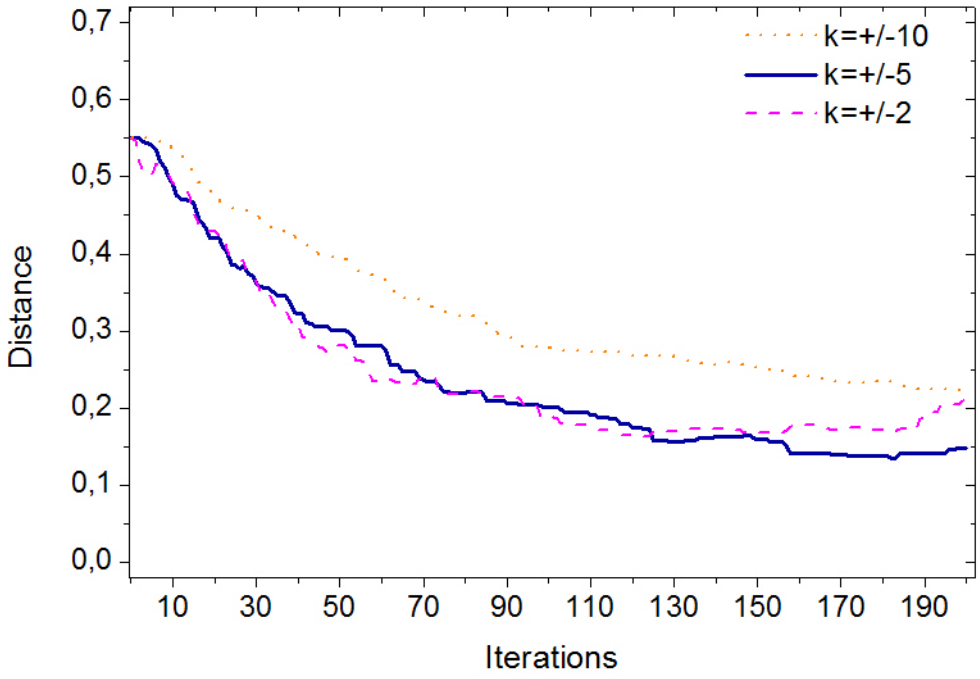


Figure 20: Influence of the level of evidence (tourism)

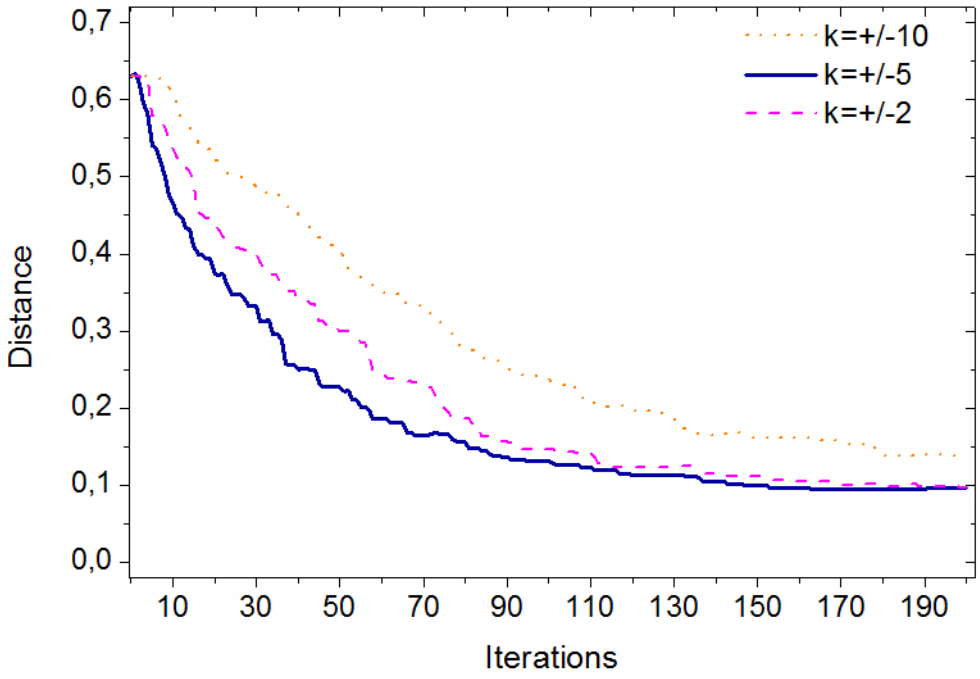


Figure 21: Influence of the level of evidence (medical centres)

4.5 Conclusions

In this chapter we have conducted a study on the parameters which tune the adaptation process, discussing how they affect the result and how they should be set up to obtain a faster and a more accurate adaptation process. However, some parameters external to the adaptation processes such as changing the label term set used or the number of criteria of the alternative may vary the ideal value for that parameters. So, we can conclude that before applying our adaptable recommendation system to a concrete environment it is necessary before to tune those parameters by conducting tests such as the ones studied in that chapter.

5 Concluding remarks

In this Master Thesis we have designed and implemented a multiple criteria recommender system, which captures and adapts the user profile in an unsupervised way. The objectives that were indicated at the beginning of the work have been successfully accomplished, differentiating the two main parts of the work: the development of a multiple criteria decision aiding system based on user profiles, and the design and evaluation of adaptation techniques to learn and update those profiles.

For the first part of the work, a recommender system has been developed. That platform evaluates a set of alternatives and rank them according to a user profile in which his interests about the criteria that define the alternatives are declared. In order to aggregate all of the information that defines each alternative, a new aggregation operator has been defined, called ULOWA. It deals either with numeric and linguistic data, and allows the declaration of preferences under an unbalanced term set.

The second part of the work consisted in designing techniques to automatically adapt the profile that is being used in the recommendation process to generate more accurate recommendations. Those techniques extract implicit information from the user interaction with the recommender platform in order to increase or decrease some preference values of the profile.

Finally, the whole system has been tested and evaluated in two synthetic environments. The first one involves the recommendation of a touristic destination from a set of cities that are described through attributes, such as

"continent", "distance to the nearest airport", "official language", "climate" and "tourism type". The second scenario supposes that the user requires a medical visit with a specialist and could be attended at some medical centres. In this case, alternatives include attributes, such as "doctors evaluation", "distance to the medical centre", "day of the week of the visit", "time of the visit" and "parking available near the medical centre". The same tests have been conducted in both domains, evaluating the influence of all parameters in the accuracy of the results during the adaptation processes.

The results of those tests show that the adjustment of the parameters depends on great measure on the recommendation domain, such as the number of criteria and the possible values of each criterion. So, it has been concluded to be necessary to tune those parameters through an evaluation phase such as the one conducted in this document before implementing the recommender system in a given domain.

As future work, many lines of research have been devised during the development of the previous parts, such as the adaptation of the numeric variables or the consideration of possible dependencies between the possible values of a linguistic variable.

Although the adaptation of numeric variables it has not been implemented, some preliminary ideas have been devised. Preferences over numeric values are represented by a single value and an area of influence (as shown in Figure 4). So, the adaptation of numeric attributes consists on increasing or decreasing that number depending on the extracted user feedback. A possible approach to evaluate numeric variables to decide how a numeric value of preference needs to be adapted starts by finding the distribution of the values of a certain numeric variable over the alternatives which are being evaluated.

Figure 22 shows the general patterns of adaptations to be considered in a numerical domain, being v_{min} and v_{max} the minimum and maximum possible values of the numeric variable respectively, v_{pref} the value of preference, v_{select} the

value the numeric variable takes in the alternative selected by the user, and $v_{over\ ranked}$ the values the numeric variable takes in the over ranked alternatives.

The casuistry of combinations of the elements v_{min}, v_{pref}, $v_{over\text{-}ranked}$ and v_{select} can be summarised in the three depicted cases.

In case (a) the preference value is located between the values of the over ranked alternatives and the value of the selected alternative. In order to adapt the preference value for this case, that value should need to be 'moved' towards the value of the selected alternative, as indicated by the red arrow. With this we achieve that our preference value is near to the value which the user selected and further from the values of the over ranked alternatives.

The over ranked alternatives values in case (b) are in the opposite side. In this case, the adaptation should also be accomplished by moving the preference value towards the selection value. However, that movement should not be as strong as in case (a) since, in this case, moving towards the selected value also means moving towards the over ranked values.

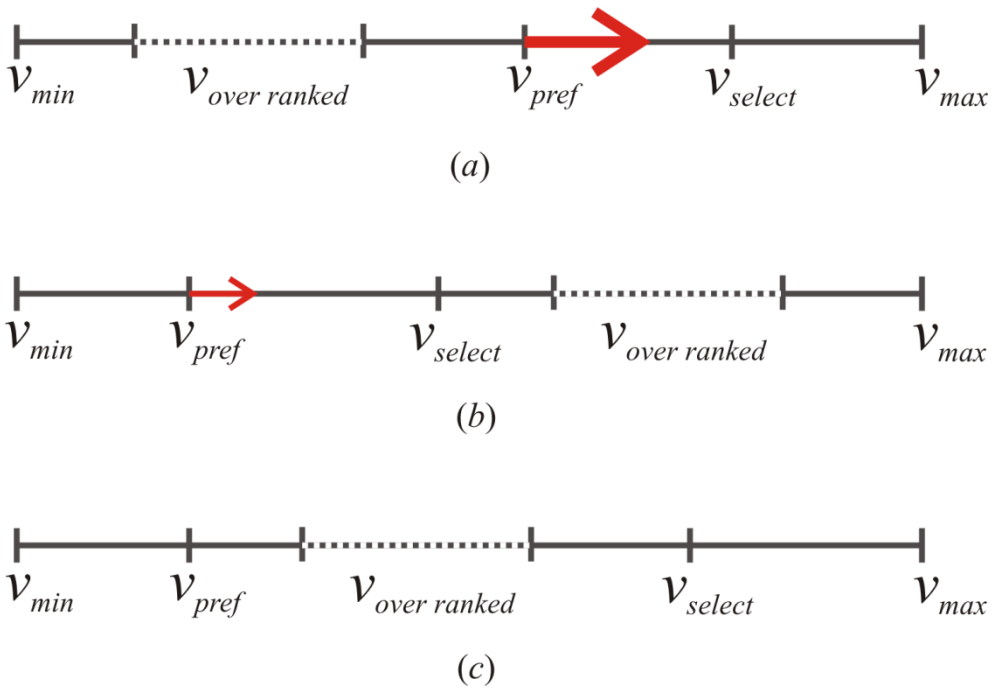


Figure 22: General patterns of behaviour in numerical attributes

Case (c) would imply no changes in the preference value of the numeric variable since moving towards the selected value would result in increasing the preference of the numeric values of the over ranked alternatives, since their values are found between.

Finally, following the same process than categorical values, several tests should be performed to know exactly the influence of parameters introduced in this part of the recommender system. For instance, we can vary the utility function to translate numerical attributes into the [0,1] domain. Additionally, these tests will be repeated when combining numerical and categorical attributes together.

Another interesting line of future work is the inclusion of additional knowledge into the recommendation process. If the system is able to obtain knowledge related to the possible values of a criterion, such as an ordinal relation or its dependency with another criterion, their mapping within a set of linguistic preference terms could be more efficient and accurate.

As a personal experience, with this work I have been introduced to the research area of the computer science. I have learned to study a concrete area of research in order to identify fields were new techniques and methodologies could represent an important contribution.

6 References

[1] B. S. Ahn, "The uncertain OWA aggregation with weighting functions having a constant level of orness," *International Journal of Intelligent Systems,* vol. 21, pp. 469-483, 2006.

[2] S. Asteriadis, P. Tzouveli, K. Karpouzis, and S. Kollias, "Non-Verbal Feedback on User Interest Based on Gaze Direction and Head Pose," Proceedings of the Second International Workshop on Semantic Media Adaptation and Personalization, 2007.

[3] D. Baker, D. Bridges, R. Hunter, G. Johnson, J. Krupa, J. Murphy, and K. Sorenson, "Guidebook to Decision-Making Methods," Dept. of Energy, USA, 2001.

[4] L. Baltrunas and X. Amatriain, "Towards Time-Dependant Recommendation based on Implicit Feedback," in *Workshop Context-aware Recommender Systems in conjunction with the 3rd ACM Conference on Recommender Systems, RecSys 09*, New York, USA, 2009, pp. 25-30.

[5] C. A. Bana E Costa, T. J. Stewart, and J. C. Vansnick, "Multicriteria decision analysis: Some thoughts based on the tutorial and discussion sessions of the ESIGMA meetings," *European Journal of Operational Research,* vol. 99, pp. 28-37, 1997.

[6] C. Basu, H. Hirsh, and W. Cohen, "Recommendation as classification: Using social and content-based information in recommendation," 1998, pp. 714-720.

[7] M. Batet, "Turist@: Sistema multi-agent recomanador d'activitats culturals," ed. Tarragona: Universitat Rovira i Virgili, Dept. Enginyeria Informàtica i Matemàtiques, 2005.

[8] G. Beliakov, A. Pradera, and T. Calvo, *Aggregation Functions: A Guide for Practitioners*: Springer Publishing Company, Incorporated, 2008.

[9] G. Boone, "Concept features in Re:Agent, an intelligent email agent," 1998, pp. 141-148.

[10] R. K. Brouwer, "Clustering feature vectors with mixed numerical and categorical attributes," *International Journal of Computational Intelligence Systems,* vol. 4, pp. 285-298, 2008.

[11] F. J. Cabrerizo, S. Alonso, and E. Herrera-Viedma, "A consensus model for group decision making problems with unbalanced fuzzy linguistic

information," *International Journal of Information Technology and Decision Making,* vol. 8, pp. 109-131, 2009.
[12] W. Cohen, "Fast Effective Rule Induction," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 115-123.
[13] P. Chatterjee, D. L. Hoffman, and T. P. Novak, "Modeling the Clickstream: Implications for Web-Based Advertising Efforts," *Marketing Science,* vol. 22, pp. 520-542, 2003.
[14] L. Chen and K. Sycara, "WebMate: A personal agent for browsing and searching," 1998, pp. 132-139.
[15] S.-M. Chen and J.-H. Chen, "Fuzzy risk analysis based on similarity measures between interval-valued fuzzy numbers and interval-valued fuzzy number arithmetic operators," *Expert Syst. Appl.,* vol. 36, pp. 6309-6317, 2009.
[16] S. J. Chen, "A new similarity measure of generalized fuzzy numbers based on geometric-mean averaging operator," 2006, pp. 1879-1886.
[17] Z. Chen, X. Meng, B. Zhu, and R. Fowler, "Websail: From on-line learning to web search," in *International Conference on Web Information Systems Engineering*, 2000.
[18] E. Cheng, F. Jing, M. Li, W. Ma, and H. Jin, "Using Implicit Relevance Feedback to Advance Web Image Search," in *IEEE International Conference on Multimedia and Expo, ICME 2006*, Toronto, ON, Canada 2006, pp. 1773-1776.
[19] J. Cheng and R. Greiner, "Learning Bayesian Belief Network Classifiers: Algorithms and System," Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, 2001.
[20] M. Delgado, F. Herrera, E. Herrera-Viedma, J. L. Verdegay, and M. A. Vila, "Aggregation of Linguistic Information Based on a Symbolic Approach," in *Computing with Words in Information/Intelligent Systems I. Foundations*, L. A. Zadeh and J. Kacpryzk, Eds., ed: Physica-Verlag, 1999, pp. 428-440.
[21] J. Dyer, "Maut — Multiattribute Utility Theory," 2005, pp. 265-292.
[22] J. Figueira, S. Greco, and M. Ehrgott, *Multiple Criteria Decision Analysis: State of the Art Surveys*: Springer, 2005.
[23] J. Fürnkranz and E. Hüllermeier, "Pairwise Preference Learning and Ranking," Cavtat, Croatia, 2003.
[24] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM,* vol. 35, pp. 61-70, 1992.
[25] Google. (2010). *Stars make search more personal*. Available: http://googleblog.blogspot.com/2010/03/stars-make-search-more-personal.html
[26] M. Grabisch, S. A. Orlovski, and R. R. Yager, "Fuzzy aggregation of numerical preferences," in *Fuzzy sets in decision analysis, operations research and statistics*, ed: Kluwer Academic Publishers, 1998, pp. 31-68.

[27] F. Herrera and E. Herrera-Viedma, "Linguistic decision analysis: Steps for solving decision problems under linguistic information," *Fuzzy Sets and Systems,* vol. 115, pp. 67-82, 2000.

[28] F. Herrera, E. Herrera-Viedma, and L. Martínez, "A fuzzy linguistic methodology to deal with unbalanced linguistic term sets," *IEEE Transactions on Fuzzy Systems,* vol. 16, pp. 354-370, 2008.

[29] F. Herrera, E. Herrera-Viedma, and J. L. Verdegay, "Direct approach processes in group decision making using linguistic OWA operators," *Fuzzy Sets and Systems,* vol. 79, pp. 175-190, 1996.

[30] D. Isern, L. Marin, A. Valls, and A. Moreno, "The Unbalanced Linguistic Ordered Weighted Averaging Operator," in *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2010*, Barcelona, Catalonia, 2010, p. (to appear).

[31] A. Jennings and H. Higuchi, "A user model neural network for a personal news service," *User Modelling and User-Adapted Interaction,* vol. 3, pp. 1-25, 1993.

[32] F. V. Jensen, "An Introduction to Bayesian Networks," New York, 1996.

[33] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," in *Document retrieval systems*, ed: Taylor Graham Publishing, 1988, pp. 132-142.

[34] D. Kelly and N. J. Belkin, "Reading time, scrolling and interaction: exploring implicit sources of user preferences for relevance feedback," presented at the Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, New Orleans, Louisiana, United States, 2001.

[35] J. S. Kim, B. A. Sohn, and B. G. Whang, "A tolerance approach for unbalanced economic development policy-making in a fuzzy environment," *Information Sciences,* vol. 148, pp. 71-86, 2002.

[36] B. Krulwich, "Lifestyle finder: Intelligent user profiling using large-scale demographic data," *AI Magazine,* vol. 18, pp. 37-45, 1997.

[37] B. Krulwich and C. Burkey, "ContactFinder: Extracting Indications of Expertise and Answering Questions with Referrals," in *Working Notes of the 1995 Fall Symposium on Intelligent Knowledge Navigation and Retrieval*, Cambridge, MA, 1995, pp. 85-91.

[38] B. Krulwich and C. Burkey, "Learning user information interests through extraction of semantically significant phrases," in *AAAI Spring Symposium on Machine Learning in Information Access*, Stanford, CA, 1996.

[39] I. Linkov, F. K. Satterstrom, G. Kiker, C. Batchelor, T. Bridges, and E. Ferguson, "From comparative risk assessment to multi-criteria decision analysis and adaptive management: Recent developments and applications," *Environment International,* vol. 32, pp. 1072-1093, 2006.

[40] L. Marin, D. Isern, and A. Moreno, "A Generic User Profile Adaptation Framework," in *13th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2010*, Espluga de Francolí, Catalunya, 2010, p. submitted.

[41] L. Marin, D. Isern, and A. Moreno, "Unsupervised Adaptation of a User Profile," in *5th Workshop on Multidisciplinary Workshop on Advances in Preference Handling, M-PREF 2010 in conjunction with ECAI 2010*, Lisbon, Portugal, 2010, p. submitted.

[42] L. Marin, D. Isern, A. Valls, and A. Moreno, "Web-based recommender using linguistic preferences," in *71st Meeting Euro Working Group 'Multiple Criteria Decision Aiding'*, Torino, Italy, 2010, pp. 50-63.

[43] L. Martínez, M. Espinilla, and L. G. Pérez, "Sensory Evaluation Model with Unbalanced Linguistic Information," in *International Conference on Intelligent Systems and Knowledge Engineering*, Chengdu, China, 2007.

[44] M. Minio and C. Tasso, "User Modeling for Information Filtering on Internet Services: Exploiting an Extended Version of the UMT Shell," UM96 Workshop on User Modeling for Information Filtering on the WWW, Kailua-Kowa, Hawaii, USA, 1996.

[45] T. M. Mitchell, S. Mahadevan, and L. I. Steinberg, "LEAP: a learning apprentice for VLSI design," in *Machine learning: an artificial intelligence approach volume III*, ed: Morgan Kaufmann Publishers Inc., 1990, pp. 271-301.

[46] B. Mobasher, R. Cooley, and J. Srivastava, "Automatic Personalization Based on Web Usage Mining," *Communications of the ACM,* vol. 43, pp. 142-151, 2000.

[47] M. Montaner, "Collaborative Recommender Agents Based on Case-Based Reasoning and Trust," PhD Thesis, Universitat de Girona, 2003.

[48] M. Montaner, B. López, and J. L. d. l. Rosa, "A Taxonomy of Recommender Agents on the Internet," *Artificial Intelligence Review,* vol. 19, pp. 285-330, 2003.

[49] A. Moukas, "Amalthea: Information discovery and filtering using a multi-agent evolving ecosystem," in *Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1996.

[50] D. M. Nichols, "Implicit Rating and Filtering," in *5th DELOS Workshop of Filtering and Collaborative Filtering*, Budapest, Hongary, 1997, pp. 31-36.

[51] M. Öztürk, A. Tsoukias, and P. Vincke, "Preference Modelling," in *Preferences: Specification, Inference, Applications*, ed. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[52] S.-F. Pan and J.-H. Lee, "eDAADe: An Adaptive Recommendation System for Comparison and Analysis of Architectural Precedents," in *4th International Conference of Adaptive Hypermedia and Adaptive Web-Based Systems, AH 2006*, Dublin, Ireland, 2006, pp. 370-373.

[53] M. Pazzani and D. Billsus, "Learning and Revising User Profiles: The Identification of Interesting Web Sites," *Machine Learning,* vol. 27, pp. 313-331, 1997.

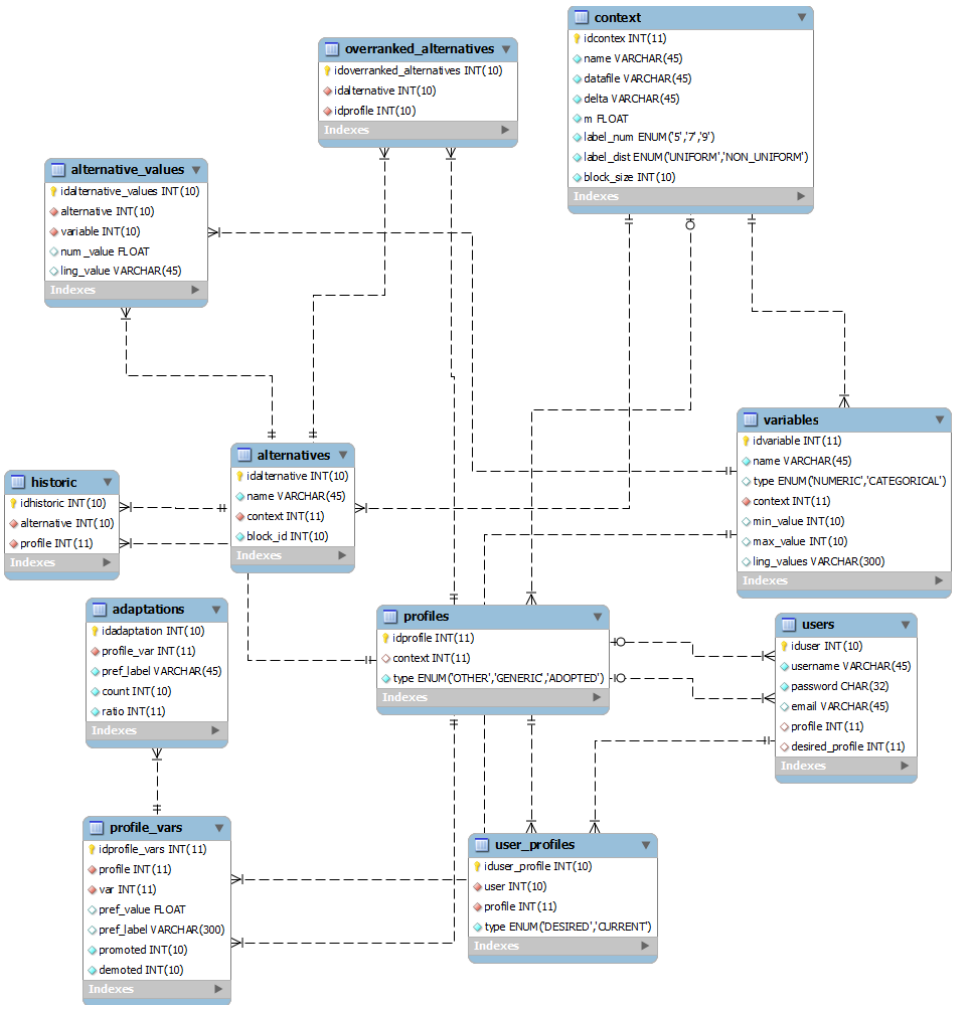
[54] P. Resnick and H. R. Varian, "Recommender Systems," *Communications of the ACM,* vol. 40, pp. 56-58, 1997.

[55] E. Rich, "User modeling via stereotypes," *Cognitive Science,* vol. 3, pp. 329-354, 1979.
[56] B. Roy, "Paradigms and Challenges," in *Multiple Criteria Decision Analysis: State of the Art Surveys*. vol. 78, ed: Springer New York, 2005, pp. 3-24.
[57] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*: McGraw-Hill, Inc., 1986.
[58] H. Sorensen and M. McElligot, "PSUN: A Profiling System for Usenet News," in *CKIM'95 Workshop on Intelligent Information Agents*, 1995.
[59] V. Torra, "The weighted OWA operator," *International Journal of Intelligent Systems,* vol. 12, pp. 153-166, 1997.
[60] V. Torra, J. Domingo-Ferrer, J. M. Mateo-Sanz, and M. Ng, "Regression for ordinal variables without underlying continuous variables," *Information Sciences,* vol. 176, pp. 465-474, 2006.
[61] V. Torra and Y. Narukawa, *Modeling Decisions: Information Fusion and Aggregation Operators (Cognitive Technologies)*: Springer-Verlag New York, Inc., 2006.
[62] G. I. Webb and M. Kuzmycz, "Feature Based Modelling: A methodology for producing coherent, consistent, dynamically changing models of agents' competencies," *User Modeling and User-Adapted Interaction,* vol. 5, pp. 117-150, 1995.
[63] X. Xu, D.-J. Lee, S. K. Antani, L. Rodney Long, and J. K. Archibald, "Using relevance feedback with short-term memory for content-based spine X-ray image retrieval," *Neurocomputing,* vol. 72, pp. 2259-2269, 2009.
[64] Z. Xu, "Linguistic aggregation operators: An overview," *Studies in Fuzziness and Soft Computing,* vol. 220, pp. 163-181, 2008.
[65] Z. Xu, "An interactive approach to multiple attribute group decision making with multigranular uncertain linguistic information," *Group Decision and Negotiation,* vol. 18, pp. 119-145, 2009.
[66] Z. Xu, X. Xu, K. Yu, and V. Tresp, "A Hybrid Relevance-Feedback Approach to Text Retrieval," ed, 2003, pp. 77-77.
[67] Z. S. Xu and Q. L. Da, "An Overview of Operators for Aggregating Information," *International Journal of Intelligent Systems,* vol. 18, pp. 953-969, 2003.
[68] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decisionmaking," *IEEE Trans. Syst. Man Cybern.,* vol. 18, pp. 183-190, 1988.
[69] T. W. Yan and H. Garcia-Molina, "SIFT: a tool for wide-area information dissemination," Proceedings of the USENIX 1995 Technical Conference Proceedings on USENIX 1995 Technical Conference Proceedings, New Orleans, Louisiana, 1995.

Appendix A: **Recommender platform**

A.1. Structure

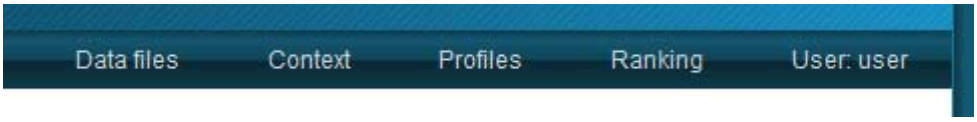
The recommender has been implemented as a web platform optimized to use with Mozilla Firefox 3.0 with a minimum resolution of 1024x768. It has been coded in PHP 5.3 using a MySQL 5.1 database. The database structure is shown in Screenshot 1.



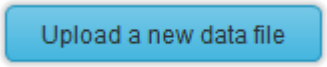
Screenshot 1: Data base model

A.2. User interface

The recommender platform has been implemented following a simple and user friendly style. The main functionalities are accessible at any time from the menu bar shown in Screenshot 2. When placing the mouse over each field of the menu, a pull-down menu appears, showing the functionalities of the platform explained in the following section.



Screenshot 2: Menu bar

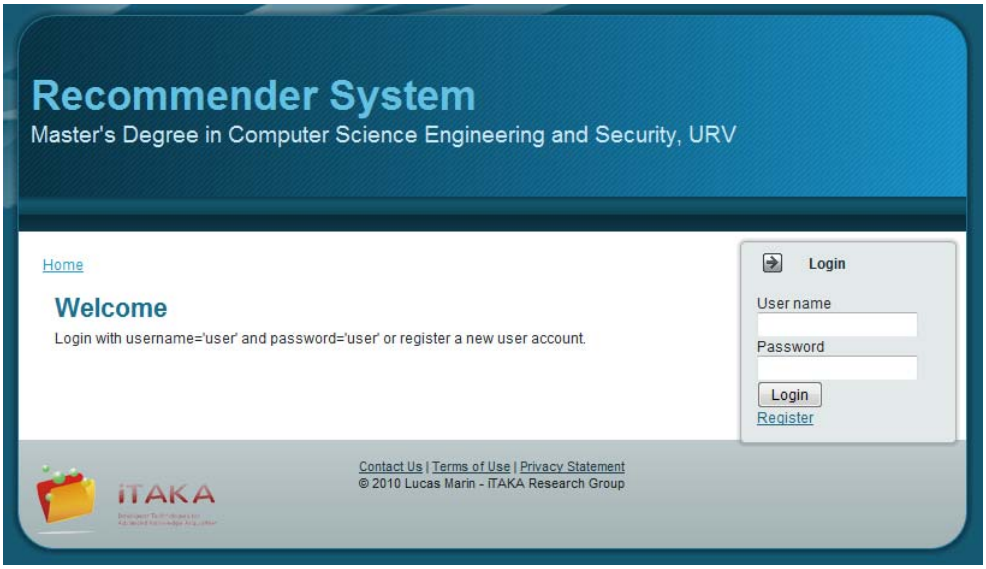
While working with the platform, bigger special buttons appear such as 

. Those kinds of buttons are quick links to the most related pages to the ones in which they appear. The usual cycle of use of the platform is similar to "upload a data file" > "create a context" > "create a profile" > "ask for a recommendation", so you will find a quick link button taking to the "create a new profile" page in the "create a context" page and so on.

A.3. Functionalities and use

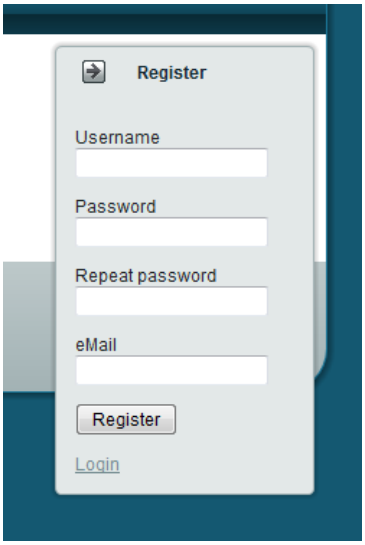
A.3.1. Platform access

When accessing the recommender address the welcoming web page in Screenshot 3 appears. To access the platform it is necessary to have a registered user name and password. However, it is possible to access by using a default user name although it is just recommender for testing purposes.

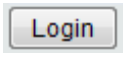


Screenshot 3: Welcome page

To register a new user, click on the Register link under the Login button. The form in Screenshot 4 appears. Introduce a new user name (which will identify you within the platform) and your desired password twice. You can also introduce your e-mail address. After introducing all that information, click on the Register button to proceed to create the new user. If any introduced information is incorrect a warning will appear. Otherwise, a confirmation message will indicate that we can now log into the platform with our new user name and password.



Screenshot 4: New user registration form

After registering a new user, the welcome page appears again. Now, you can introduce you user name and password on the login form and click on the  button to access the platform. Once logged into the platform, the upper menu bar is filled with the main functionalities, as shown in Screenshot 2.

A.3.2. Data files management

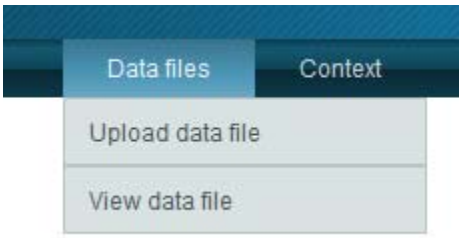
The application gives recommendations considering a set of alternatives extracted from a data file which contains the details that describe those alternatives. In order to use a custom data file it is necessary to generate it with the following structure: the first line contains name of the criteria and the following lines contain the alternatives, starting by its name and followed by the values of each criteria, separated with a blank space.

```
#Attribute names
City Continent Climate Nearest-Airport-Distance Official-Language Pop-Density Elevation Tourism-type

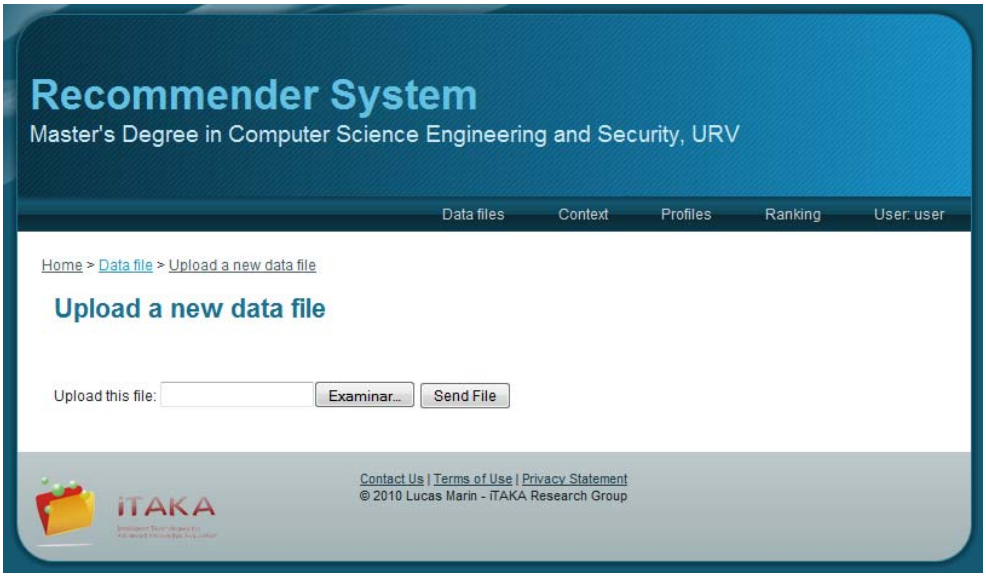
#Values
Zurich Europe Humid_continental 10 German 4100 410 Sightseeing
Buenos_Aires South_America Humid_subtropical 22 Spanish 15000 0 Adventure
Rio_De_Janeiro South_America Tropical_Savanna 0 Portuguese 4800 0 Beach
Honolulu North_America Tropical 5 English 1700 0 Beach
Tokyo Asia  Humid_subtropical 14 Japanese 5800 0 Sightseeing
Queenstown Oceania Alpine 7 English 2 350 Adventure
Cairo Africa Desert 15 Arabic 31600 120 Sightseeing
Rome Europe Mediterranean 40 Italian 2200 40 Sightseeing
Topeka North_America Humid_continental 5 English 840 290 Rural
Oviedo Europe Oceanic 45 Spanish 1180 230 Rural
```

Screenshot 5: Data file

After creating a data file with the correct format (such as the example in Screenshot 5), it can be uploaded to the platform by clicking on the "*Upload data file*" option from the pull-down menu that appears when you place the mouse over the "*Data files*" button in the menu bar, as seen in Screenshot 6.



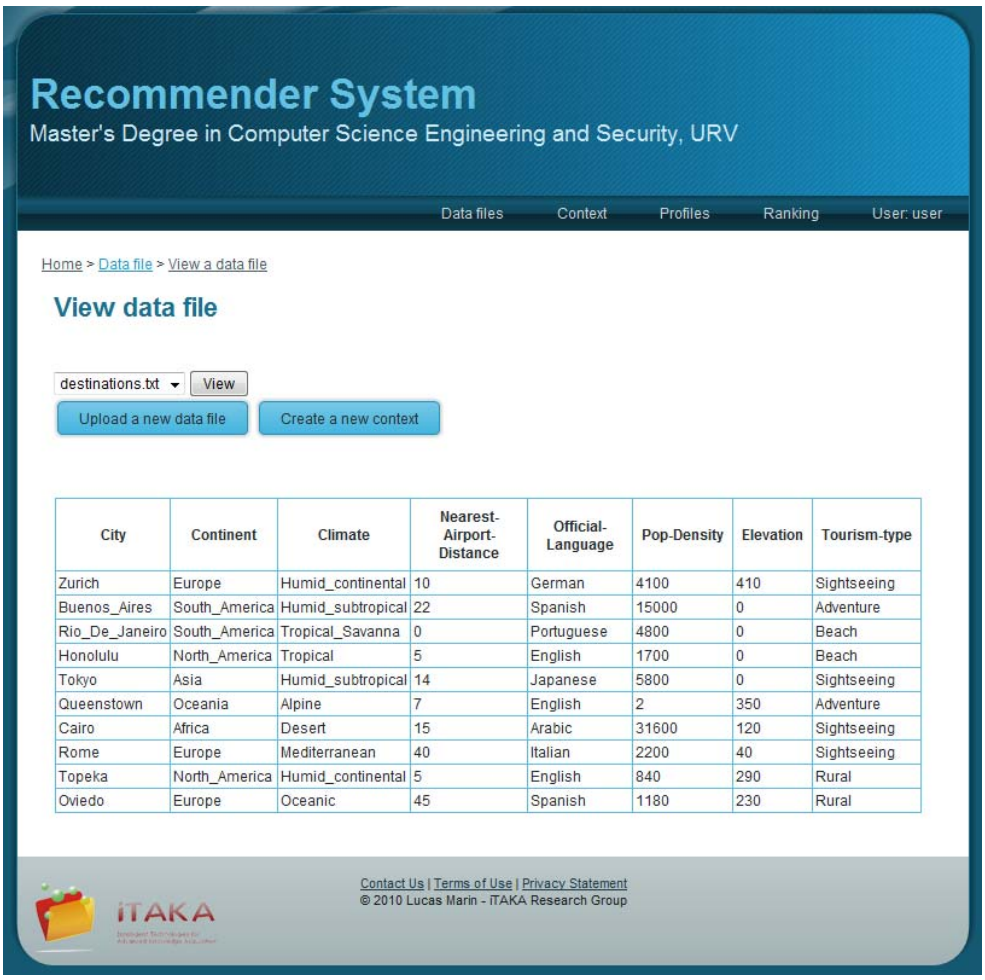
Screenshot 6: Data files pull-down menu



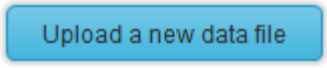
Screenshot 7: Data file upload form

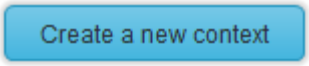
The page shown in Screenshot 7 appears. By clicking on the Examinar... button you can search on your computer the data file to upload. After finding it, you can start the upload process by clicking on the Send File button.

Once you have uploaded a data file, you can view it by going to the "*View data file*" page from the pull-down menu. A drop-down list appears in which you can select any of the data files previously uploaded to the platform. By clicking on the View button, the information in the selected data file is shown in a table, as can be seen in Screenshot 8.



Screenshot 8: Data file view

Two examples of the quick link buttons explained in the previous section appear here: 

and 

. The first one takes you to the data file upload form described before and the second one to the context creation form, which is the operation that needs to be done after uploading a data file before using it in any recommendation process.

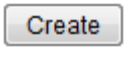
A.3.3. Context management

Creating a context configures the environment in which recommendations will be made. Before creating it, it is necessary to have a correctly uploaded data file with a corpus of alternatives. To create a new context, select the "*Create new context*" option from the "*Context*" pull-down menu, as in Screenshot 9.



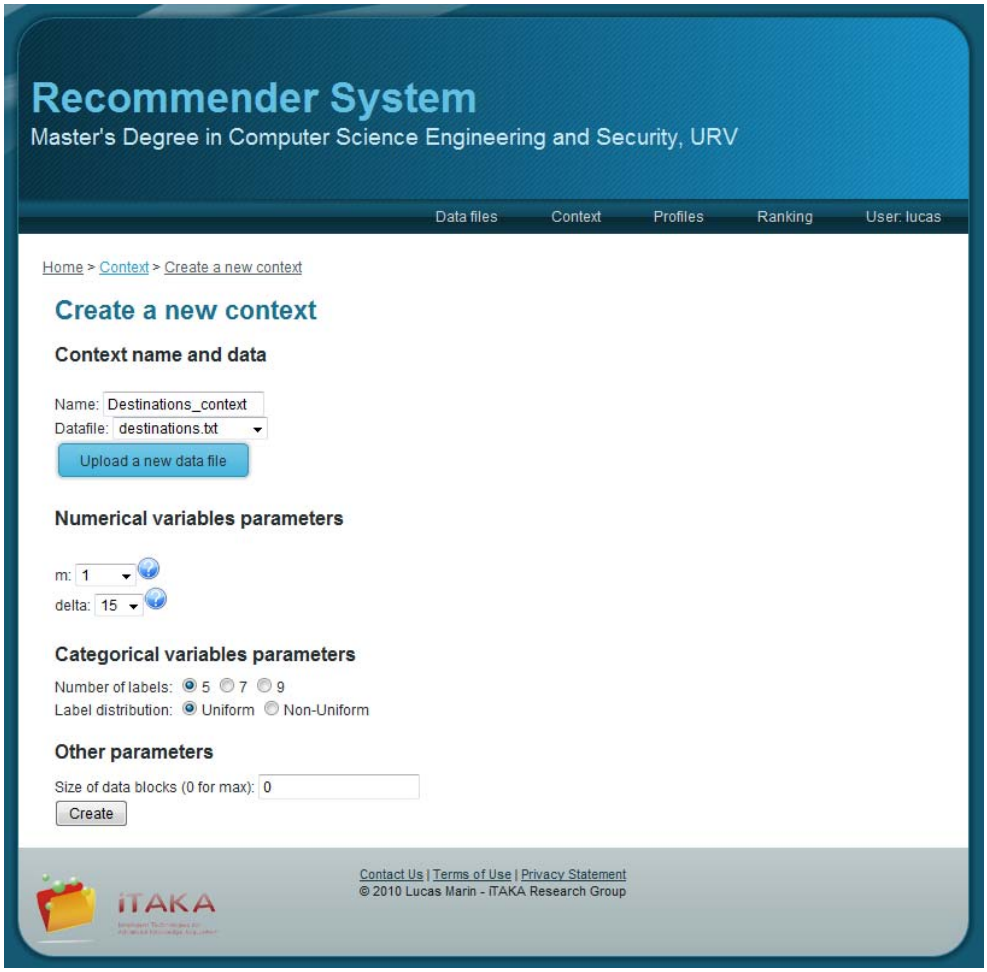
Screenshot 9: Context pull-down menu

It is necessary to indicate five parameters. For the numeric variables, it is necessary to select the slope (*m*) and the percentage of deviation (*delta*) of the numeric utility function explained in the section 2.5.2 of the Master Thesis document. For the linguistic variables, it is necessary to select the number of labels to express the preferences (5, 7 or 9) and their distribution (uniform or non-uniform). Finally, a size of the data block can be expressed; this field is just used for the simulations, when many recommendations (evaluating at each one a "block" of alternatives the size of the number introduced here) are done consecutively and automatically.

After introducing all of the required parameters, you can click the 

button to store the context in the system.

Created contexts can be viewed by selecting the "*View created contexts*" option from the pull-down menu in Screenshot 9. In that page, you can select a previously created context from the drop-down list and click on the View button to visualise it. You can change any of the parameter and store the changes by clicking on the Update button.



Screenshot 10: Creating a new context

A.3.4. Profile management

Once the context is created, a new profile can be created in order to express the user preferences over said context alternatives. This can be done by selecting the

"*Create new profile*" option from the "*Profiles*" pull-down menu as shown in Screenshot 11.

The first step to create a profile is to select the context in which it will operate and if you want it to be initialised randomly if you don't want to fill it manually. Once decided that, you can click on the Load button.

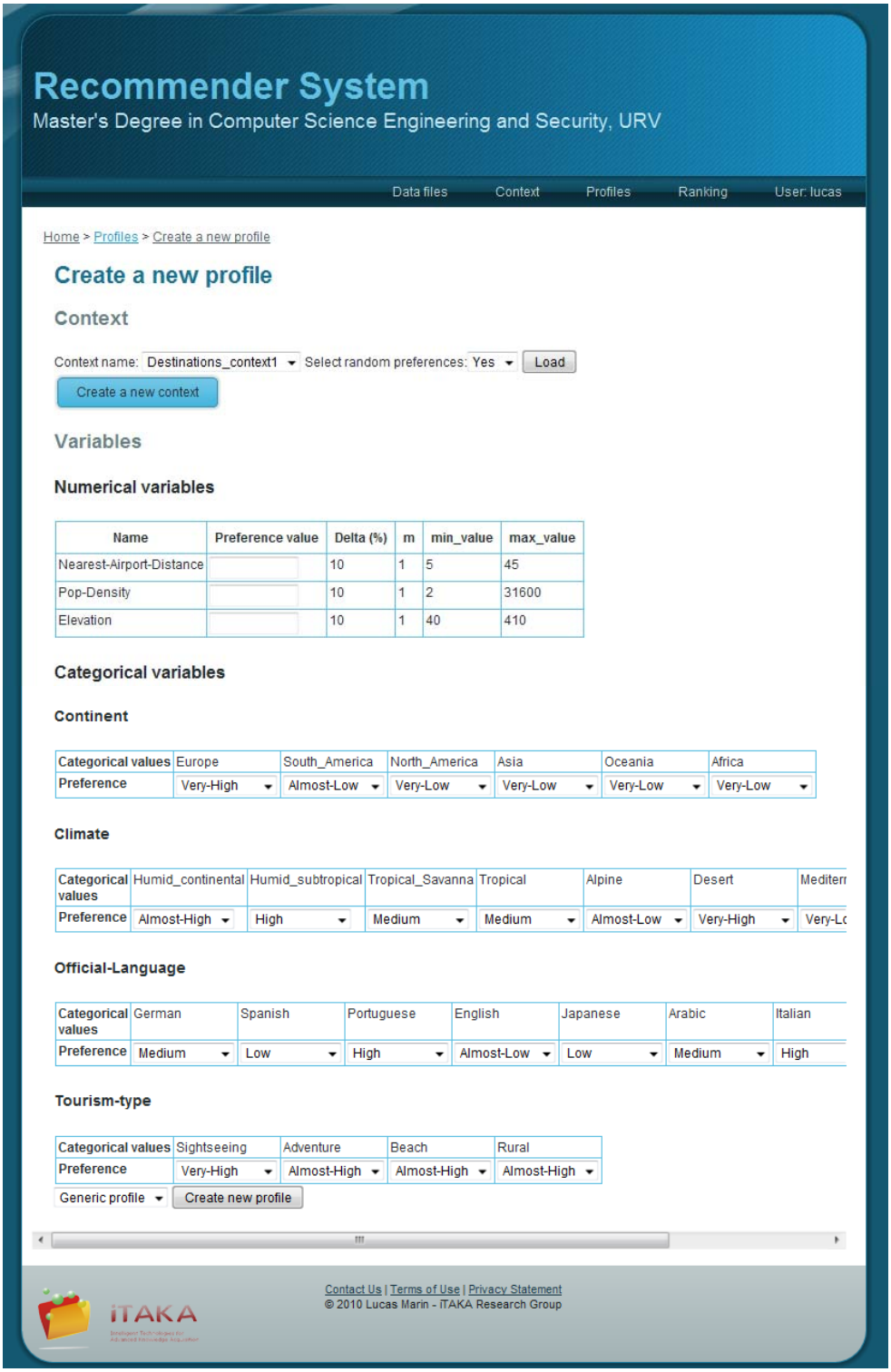


Screenshot 11: Profiles pull-down menu

Then, all of the attributes which define the context alternatives are shown, separated by its type: numerical and categorical. For the first ones it is necessary to select a numeric value. Information related the variable is also shown, such as the minimum and maximum values found on the data file for this variable and its *delta* and *m* configured in the context. For the categorical (linguistic) variables, it is necessary to select a level of preference for each possible value of each variable (see Screenshot 12). After indicating the preference for all of the variables, you can click on the Create new profile button and store the new profile on the data base.

Stored profiles can be viewed by selecting the "*View created profiles*" option in the pull-down menu of Screenshot 11 and choosing them on the drop-down list that appears. Here, you can also delete previously created profiles by selecting them on the list and clicking on the Delete button.

Before using the profile to make a recommendation, there is one last step that needs to be taken. It consists in associating a certain profile to a user. This is done by selecting the "*Adopt profiles*" option from the "*User*" pull-down menu shown in Screenshot 13.

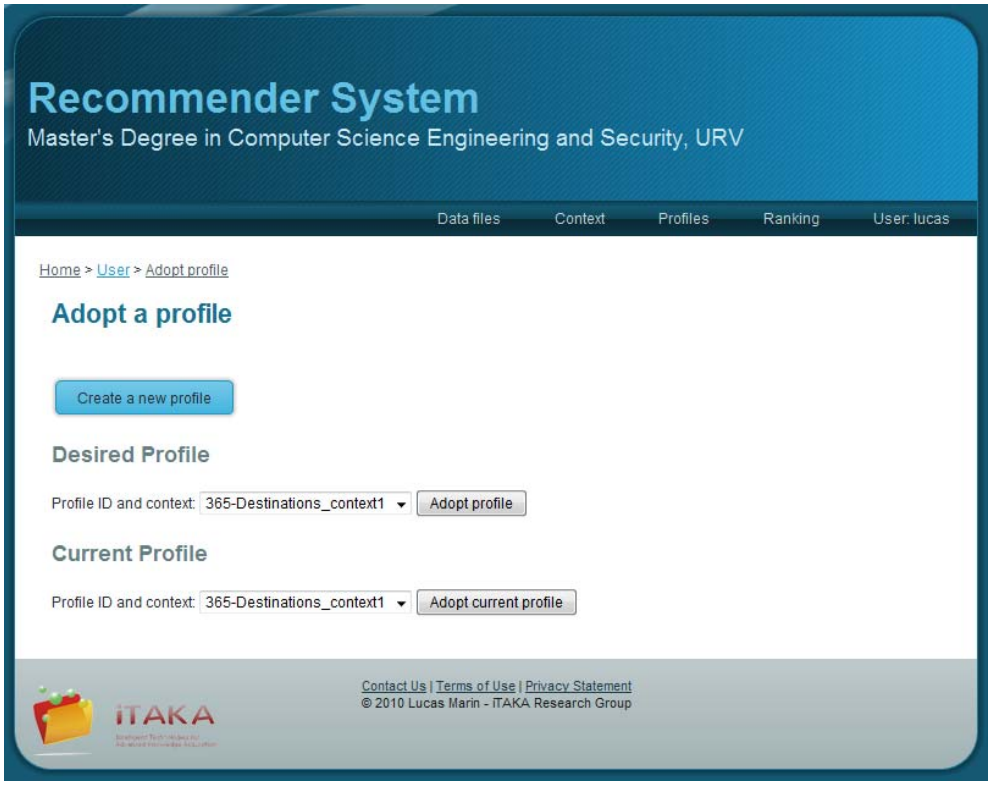


Screenshot 12: Creating a new profile



Screenshot 13: User pull-down menu

Here you can adopt two kinds of profiles. If you want to adopt a profile which needs to be adapted, you need to adopt it as a current profile by selecting it on the "Current profile" section drop-down list and clicking on the Adopt current profile button. If you want to adopt a profile which will be used in the automatic evaluation process as an ideal profile instead, select it in the "Desired profile" section drop-down list and click on the Adopt profile button.

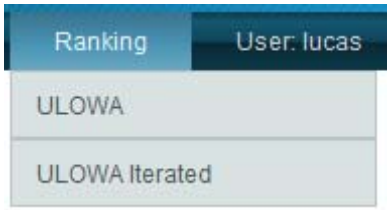


Screenshot 14: Adopting profiles

Note that for the automatic evaluation process it is also necessary to have an adopted profile, as is explained below.

A.3.5. Generation of recommendations

Once the data file has been uploaded, the context configured, and a profile created and adopted, it is time to ask for a recommendation. This is done through the "Raking" pull-down menu shown in Screenshot 15.



Screenshot 15: Ranking pull-down menu

You have to click on the "*ULOWA*" option to ask for a recommendation. In the page that appears, you can select your previously created profile from the list and click on the Load button. Then, the selected profile is shown and before asking for the recommendation, you can select the aggregation policy between "most", "at least half", "as many as possible", "average" or "most". Those aggregation policies are more detailed in the chapter 2 of the Master Thesis documentation. After selecting the desired policy, you can press the Rank alternatives button to start the recommendation process. A new page appears showing the result of the recommendation (see Screenshot 16).

Here you can select your favourite alternative by marking the radio button on the "*User's choice*" field. Then, if you click on the Submit choice button, the adaptation process is started.

If you want to run the automatic evaluation simulation, you have to select the "*ULOWA Iterated*" option from the menu shown in Screenshot 15. Here you need to select the profile you want to adapt and the ideal profile you want to

reach. Then you can click on the Load button and select the aggregation policy and the number of recommendations (iterations) of the automated process. When those last parameters are introduced, you can click on the Start adaptation process button. A new page loads, showing in real time the changes that are being made to the user profile.

Ranking	Name	Valoration	User's choice
0	Rome	Medium	
1	Tokyo	Medium	
2	Buenos_Aires	Almost-Low	
3	Topeka	Almost-Low	
4	Oviedo	Almost-Low	
5	Cairo	Almost-Low	
6	Queenstown	Almost-Low	
7	Honolulu	Almost-Low	
8	Zurich	Almost-Low	
9	Rio_De_Janeiro	Low	

Submit choice LOWA calculation took 0.022 seconds.

Screenshot 16: Recommendation result

Appendix B: **Publications**

B. 1 The Unbalanced Linguistic Ordered Weighted Averaging Operator 93

Abstract: Aggregation operators for linguistic variables usually assume a uniform and symmetrical distribution of the linguistic terms that define the variable. A well-known aggregation operator is the Linguistic Ordered Weighted Average (LOWA), which has been extensively applied. However, there are some problems where an unbalanced set of linguistic terms is more appropriate to describe the objects. In this paper we define the Unbalanced Linguistic Ordered Weighted Average (ULOWA) on the basis of the LOWA operator. ULOWA takes into account the fuzzy membership functions of the terms during the aggregation process. There is no restriction on the form of the membership functions of the terms, which can be triangular or trapezoidal, non symmetrical and non equally distributed. The paper demonstrates the properties of ULOWA. Finally, a comparison of this operator with some other aggregation operators for unbalanced sets of terms is done.
Accepted in the "2010 IEEE International Conference on Fuzzy Systems", Barcelona, Spain, 2010. (category A).

B. 2 Web-based Recommender using Linguistic Preferences.....101

Abstract: In this work we propose a recommender system that permits to represent a user's profile including qualitative and quantitative criteria. The system has been implemented as a Web application. It is designed in a generic way so that it can be applied to any decision making problem. The alternatives are aggregated using a linguistic averaging operator. Then, they are ranked and presented to the user. The system includes some predefined sets of linguistic terms to be used to represent qualitative preferences. Some unbalanced sets have been defined. The ULOWA operator is proposed to make the aggregation in balanced and unbalanced sets of terms. The recommender has been designed to permit the update of the user's preferences in an unsupervised way. The evolution of the user's profile is based on the information provided by his selections, taking into account both the alternative selected and the set of discarded alternatives.
Accepted in the "71st Meeting of the European Working Group ‹‹Multiple Criteria Decision Aiding››", Torino, Italy, 2010.

B. 3 Unsupervised Adaptation of a User Profile..... 115

Abstract: The paper shows an algorithm to deal with user preferences using linguistic criteria. The system has been implemented as a Web application and designed in a generic way so that it can be applied to any decision making problem. The system is composed by two separate modules: a module to rate and rank all alternatives received by the system according to the current interests of the user, and a module to adapt the current user profile in an unsupervised fashion collecting implicit information about the user interaction with the system. The paper, focused in the second step, presents some results and discusses the influence of several parameters introduced in the algorithm.
Submitted to the "5th Multidisciplinary Workshop on Advances in Preference Handling" within the "19th European Conference on Artificial Intelligence", Lisbon, Portugal, 2010.

B. 4 A Generic User Profile Adaptation Framwork..... 121

Abstract: In this work we propose a recommender system that permits to represent a user's profile including qualitative and quantitative criteria. The system has been implemented as a Web application. It is designed in a generic way so that it can be applied to any decision making problem. The alternatives are aggregated using a linguistic averaging operator. Then, they are ranked and presented to the user. The system includes some predefined sets of linguistic terms to be used to represent qualitative preferences. Some unbalanced sets have been defined. The ULOWA operator is proposed to make the aggregation in balanced and unbalanced sets of terms. The recommender has been designed to permit the update of the user's preferences in an unsupervised way. The evolution of the user's profile is based on the information provided by his selections, taking into account both the alternative selected and the set of discarded alternatives.

Submitted to the "13è Congrés Internacional de l'Associació Catalana d'Intel·ligència Artificial", Tarragona, Spain, 2010.

The Unbalanced Linguistic Ordered Weighted Averaging Operator

David Isern, Lucas Marin, Aida Valls and Antonio Moreno

Abstract— **Aggregation operators for linguistic variables usually assume a uniform and symmetrical distribution of the linguistic terms that define the variable. A well-known aggregation operator is the Linguistic Ordered Weighted Average (LOWA), which has been extensively applied. However, there are some problems where an unbalanced set of linguistic terms is more appropriate to describe the objects. In this paper we define the Unbalanced Linguistic Ordered Weighted Average (ULOWA) on the basis of the LOWA operator. ULOWA takes into account the fuzzy membership functions of the terms during the aggregation process. There is no restriction on the form of the membership functions of the terms, which can be triangular or trapezoidal, non symmetrical and non equally distributed. The paper demonstrates the properties of ULOWA. Finally, a comparison of this operator with some other aggregation operators for unbalanced sets of terms is done.**

I. Introduction

The fuzzy linguistic approach has been widely used for making assessments in a qualitative setting. In this paper, we will concentrate on the aggregation of linguistic terms, which is part of many other procedures, such as decision making [19].

In the last decades, various linguistic aggregation operators have been proposed, including linguistic median operator [24], linguistic weighted median operator [23], median-based aggregation operator [8], 2-tuple weighted averaging operator [12], 2-tuple OWA operator [12], linguistic OWA operators [6], [10], etc. In most of those aggregation operators, a set of equally-informative (balanced) terms is assumed. That is, the membership functions of the fuzzy terms are symmetric and the terms are uniformly distributed around a mid (*i.e.*, neutral) term. For example, when evaluating the users' satisfaction on some topic, linguistic labels like "none", "very low", "low", "medium", "high", "very high", "perfect" can be used [20].

However, recently many authors have noticed that there are some problems that require a more flexible definition of the set of linguistic terms [11], [21]. In some cases, fuzzy sets should be asymmetric or more positive than negative terms are needed (or vice-versa), leading to an unbalanced set of labels. An example is the set of terms used in grading evaluations $S = \{F, D, C, B, A\}$ (being D the neutral; A, B and C, positive marks, and F - fail - the unique negative value). Many other real-life applications also make use of unbalanced terms, such as [21], [18], [16].

Some extensions of well-known aggregation operators have been defined to deal with unbalanced sets of terms. Proposals of [2] and [11] are based on the linguistic 2-tuple representation model, consisting on attaching to each label a deviation value that permits to work with more precision. In [21] the Uncertain Linguistic Weighted Average is proposed. It restricts to the case of sets of labels where the absolute value of the deviation between the indices of two adjacent labels increases as the indices increase. An additional number is attached to each label, indicating its degree of separation from the mid term. The aggregation operator uses those additional numbers to calculate the result. In this paper, we assume the classic fuzzy linguistic model that associates a fuzzy membership function to each linguistic value and operates in terms of those fuzzy sets.

The LOWA operator defined in [10] is the basis of the new operator, called ULOWA (Unbalanced Linguistic Ordered Weighted Average). LOWA has been chosen because it has been extensively used in many applications and it is computationally efficient. LOWA relies on a symbolic model of the set of linguistic terms, and the aggregation is done considering only the positions of the labels in the set of terms, but without taking into account any fuzzy set associated with the terms. The aggregation procedure of ULOWA is the same to the one used in LOWA. However, in the convex combination of two labels, ULOWA exploits the knowledge given by the membership functions during the aggregation procedure. This approach, although it is focused on managing unbalanced label sets, can also be used with balanced label sets giving the same results than the LOWA operator.

The rest of the paper is organised as follows. Section II gives some preliminary definitions. Section III defines the proposed aggregation method, ULOWA. Section IV introduces the aggregation operator properties. Section IV gives some examples to illustrate the behaviour of the operator. Section VI compares ULOWA with other previous operators. Finally, section VII lists some conclusions and proposes some lines of future work.

David Isern is with Universitat Rovira i Virgili (URV), Departament d'Enginyeria Informàtica i Matemàtiques (DEIM), Escola Tècnica Superior d'Enginyeria (ETSE), Intelligent Technologies for Advanced Knowledge Acquisition (ITAKA) research group, Av Països Catalans, 26. 43007, Tarragona, Catalonia (Spain) (corresponding author, phone: +34977256563; fax: +34977559710; e-mail: david.isern@urv.cat)

Lucas Marin is with Universitat Rovira i Virgili (URV), DEIM, ETSE, ITAKA Research Group (e-mail: lucas.marin@urv.cat)

Aida Valls is with Universitat Rovira i Virgili (URV), DEIM, ETSE, ITAKA Research Group (e-mail: aida.valls@urv.cat)

Antonio Moreno is with Universitat Rovira i Virgili (URV), DEIM, ETSE, ITAKA Research Group (e-mail: antonio.moreno@urv.cat)

II. Preliminaries

A. Linguistic values

Let us consider a set of linguistic labels $S = \{s_i\}, i \in \{0, .., T\}$. Let us consider a set of linguistic labels [1], [11], [20]. taking S as a finite and totally ordered term set on a reference domain $X = [0, 1]$, with an odd cardinal, where one of the labels corresponds to the neutral value and the remaining terms are placed around it. The cardinality of the set must be small enough so as not to impose useless precision and rich enough in order to allow an appropriate discrimination level. The usual cardinality values are 7 or 9.

The semantics of the linguistic labels are given by a trapezoidal membership function $\mu : X \rightarrow [0, 1]$ that is represented with a tuple $A = (a_1, a_2, a_3, a_4)$, where a_1, a_2, a_3 and a_4 are the points in the reference domain X which define the trapezoid (see Fig. 1). Some special cases can be defined. If $a_1 = a_2$ and $a_3 = a_4$ then A corresponds to a crisp interval. If $a_2 = a_3$ the fuzzy set A is triangular. If $a_1 = a_2 = a_3 = a_4$, then A is called a crisp real number. This last case will be especially relevant for the aggregation method that we propose.

B. Centre of gravity

Let us denote the centre of gravity (COG, [3]) of a trapezoidal fuzzy set A as $COG(A) = (x^*_A, y^*_A)$, which can be calculated as follows :

$$y^*_A = \begin{cases} \frac{1}{6}\left(\frac{a_3 - a_2}{a_4 - a_1} + 2\right) & \text{if } a_1 \neq a_4, \\ \frac{1}{2} & \text{if } a_1 = a_4. \end{cases}$$
$$x^*_A = \frac{y^*_A(a_3 + a_2) + (a_4 + a_1)(1 - y^*_A)}{2} \quad (1)$$

C. Similarity between fuzzy sets

Several functions to measure the degree of similarity between two fuzzy sets have been defined. In our case, as it will be seen in section III, we require a function that measures the similarity between a crisp number and a trapezoidal or triangular fuzzy set. For this reason, we will consider the ones referenced in Table 1 [3-5, 14, 26]. These functions are based on the different properties of the membership function of the fuzzy set, such as the centre of gravity, the radius of gyration, or the geometric mean. For any similarity function, the following three properties should be satisfied [4]. Given two fuzzy sets A and B:

1) A and B are identical if and only if $Sim(A, B) = 1$,
2) $Sim(A, B) = Sim(B, A)$
3) If $A = (a, a, a, a)$ and $B = (b, b, b, b)$ denote two crisp numbers in [0..1], then $Sim(A, B) = 1 - |a - b|$.

Fig. 1 shows different combinations where there is a crisp number and a fuzzy set (triangular or trapezoidal). Table I gives the results obtained with different similarity functions in those 8 cases. For our purpose, some relations between the similarity values given to each of those cases should be satisfied. In the sequences defined by the cases 1, 2, 3, and 4, 5, 6, the crisp value has different degrees of overlapping with the fuzzy set. In those cases, the similarity value should increase as the crisp value gets inside the fuzzy set. Moreover, the similarity in case 3 should be greater than in cases 4 and 5, and even 6. It can be noted that the results obtained with our similarity measure permit to distinguish all cases shown in Fig. 1, with independence of the particular values.

Existing similarity measures do not fulfil these conditions (marked in Table I as anomalous results). In the case of Hsieh and Chen [14], the cases 1 and 5 cannot be distinguished. The Chen [4] function cannot distinguish the cases 1 and 6.

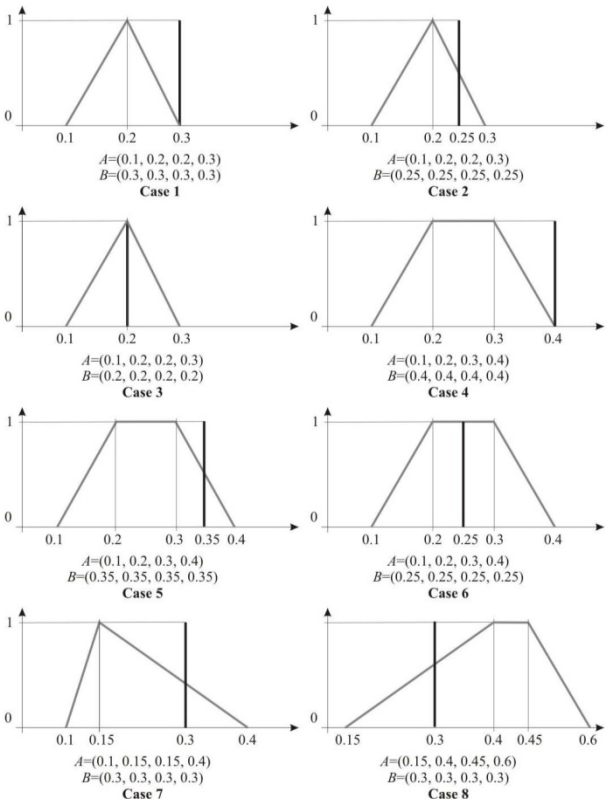


Fig. 1. Combinations used to analyse the similarity functions

TABLE I
COMPARISON OF SIMILARITY FUNCTIONS BETWEEN FUZZY SETS

Set	Hsieh & Chen [14]	Chen [4]	Chen & Chen [3]	Yong [25]	Chen [5]	[5] modified
1	0.9091	0.9000	0.5400	0.5854	0.5991	0.8987
2	0.9524	0.9250	0.5858	0.6344	0.6163	0.9245
3	1.0000	0.9500	0.6333	0.6584	0.6329	0.9494
4	0.8696	0.8500	0.5619	0.6387	0.6585	0.8466
5	0.9091	0.8750	0.6125	0.6705	0.6791	0.8731
6	1.0000	0.9000	0.7000	0.7656	0.6995	0.8993
7	0.8955	0.8500	0.5194	0.5821	0.5664	0.8497
8	0.9023	0.8250	0.5268	0.5620	0.5794	0.8234

"" means anomalous results

The rest of functions give a lower similarity in the case 3 when compared with cases 4, 5 or 6. Therefore, we chose the similarity function between two fuzzy sets A and B defined by Chen [5], which was proven to be more adequate than the rest (details in [5]), and we have adapted it. In particular, we avoid the scale factor designed to compare two general fuzzy numbers with different heights as, in our scenario, the height of the membership function is always 1. The new function is given in (2). The results obtained with this similarity function are given in the last column of Table I.

$$Sim(A,B) = \sqrt[4]{\prod_{i=1}^{4}(2-|a_i-b_i|)} - 1 \qquad (2)$$

III. The Unbalanced Linguistic Weighted Average

In this section, the LOWA operator [10] is modified in order to be applicable to unbalanced linguistic terms. First of all, we present the LOWA definition. Afterwards, the ULOWA modification on this operator is given. Let be a set of labels, where , that have to be aggregated with respect to a set of weights such that and . Those weights specify the decision-maker policy, so that we can emphasize different arguments based upon their ordered position. Weights are used to define different conjunction/disjunction aggregation models [22]. The LOWA operator, ϕ_{LOWA}, is defined as:

$$\phi_{LOWA}(a_1,\ldots,a_m) = W \cdot B^T = \\ = C^m\{w_k, b_k, k=1,\ldots,m\} = \\ = w_1 \otimes b_1 \oplus (1-w_1) \otimes C^{m-1}\{\beta_h, b_h, h=2,\ldots,m\} \quad (3)$$

Where $\beta_h = \frac{w_h}{\sum_2^m w_h}, h = 2, ..., m$, and $B = (b_1, \ldots, b_m)$ is a permutation of the elements of A, such that $B = \sigma(A) = \{a_{\sigma(1)}, ..., a_{\sigma(m)}\}$, where $a_{\sigma(j)} < a_{\sigma(i)}\ \forall i \leq j$.

C^m is the convex combination operator of m labels. If $\exists j \in 1,\ldots,m,\ w_j = 1$ (so $\forall k \in 1,\ldots,m,\ k \neq j\ w_j = 0$), then $C^m\{w_i, b_i, i = 1,\ldots,m\} = b_j$.

When $m = 2$, with $b_1 = s_j;\ b_2 = s_i;\ s_j, s_i \in S\ (j \geq i)$ then:

$$C^2\{w_i, b_i, i = 1,2\} = w_1 \otimes s_j \oplus (1-w_1) \otimes s_i = s_k$$

such that $k = min\{T, i + round(w_1(j-i))\}$.

Notice that, in this definition, the resulting label s_k is obtained considering the position of the labels s_j and s_i in the linguistic domain, but without considering the form of the membership function associated with each term.

Considering only the order between the terms implies the assumption that terms are defined with symmetric membership functions and uniformly distributed. For this reason, as it has been said, the LOWA operator is not able to deal with unbalanced sets of linguistic terms.

The new operator, called ULOWA (Unbalanced Linguistic Ordered Weighted Average), is defined on the basis of the LOWA operator. In fact, the definition is the same until we have to make the aggregation of two labels ($m = 2$). The convex combination of two terms $b_1 = s_j$ and $b_2 = s_i$, with $s_j, s_i \in S\ (j \geq i)$ is calculated taking into account the membership functions of labels s_j and s_i.

$$C^2\{w_i, b_i, i = 1,2\} = w_1 \otimes s_j \oplus (1-w_1) \otimes s_i = s_k \\ \text{such that } k = argmax_{i \leq p \leq j}\{Sim(s_p, \delta)\} \qquad (4)$$

where δ is an intermediate crisp number defined as $\delta = (x_k, x_k, x_k, x_k)$ with $x_k = x^*_{s_i} + w_1(x^*_{s_j} - x^*_{s_i})$.

Note that we use the COG of labels s_j and s_i (1), and the similarity function for two fuzzy sets (2) described above. Fig. 2 illustrates the aggregation procedure of the two extreme labels (VL and P) with a "mean" policy. The figure shows the COGs of both labels and their intermediate crisp number δ used to find the result of the aggregation. After the application of the similarity function (2), the resulting label is the neutral term M. Using the LOWA operator, the result would have been AH, which is not the intuitive expected result of the mean average of these two opposite evaluations, considering the meaning of the terms.

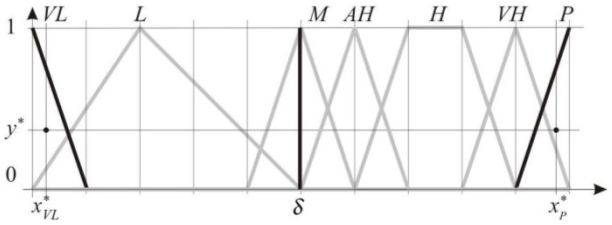


Fig. 2. Graphical example of ULOWA when aggregating two labels (VL and P) with a mean average

IV. Properties of the ULOWA operator

As stated in [23], an OWA operator should satisfy the following properties: *monotonicity*, *commutativity*, *idempotency*, and *to be an "orand" operator*.

Property 1: *The ULOWA operator ϕ is increasing monotonous with respect to the argument values, in the following sense:*

Let $A = [a_1, \ldots, a_m]$ and $B = [b_1, \ldots, b_m]$ be two ordered argument vectors, such that $\forall j, a_j \geq b_j$ then:

$$\phi(A) \geq \phi(B)$$

Proof: By induction over the number of arguments to aggregate.

a) For $m = 2$. Let $x^*_{s_j}, x^*_{s_i}, x^*_{s_q}, x^*_{s_p}$ the x coordinate of the centre of gravity of the ordered labels s_j, s_i, s_q, s_p corresponding to a_1, a_2, b_1, b_2, respectively; and δ_{ji} as the intermediate point between s_j and s_i, and δ_{qp} as the intermediate point between s_q and s_p. In this case, $x^*_{s_j} \geq x^*_{s_q}$ and $x^*_{s_i} \geq x^*_{s_p}$; therefore, given any $w_1 \in [0,1]$ then $x^*_{s_j} w_1 \geq x^*_{s_q} w_1$ and $x^*_{s_i}(1-w_1) \geq x^*_{s_p}(1-w_1)$, and thus $x^*_{s_j} w_1 + x^*_{s_i}(1-w_1) \geq x^*_{s_q} w_1 + x^*_{s_p}(1-w_1)$.

From this expression, we can derive that:

$$x^*_{s_i} + w_1(x^*_{s_j} - x^*_{s_i}) \geq x^*_{s_p} + w_1(x^*_{s_q} - x^*_{s_p})$$

Knowing that $\delta_{ji} = (x_k, x_k, x_k, x_k)$ with $x_k = x^*_{s_i} + w_i(x^*_{s_j} - x^*_{s_i})$ and $\delta_{qp} = (x_r, x_r, x_r, x_r)$ with $x_r = x^*_{s_p} + w_p(x^*_{s_q} - x^*_{s_p})$, we can deduce $x_k \geq x_r$.

Given a fuzzy number A, its similarity with the previous values using the similarity function (2) are:

$$Sim(A, \delta_{ji}) = \sqrt[4]{\prod_{i=1}^{4}(2 - |a_i - x_k|)} - 1$$

$$Sim(A, \delta_{qp}) = \sqrt[4]{\prod_{i=1}^{4}(2 - |a_i - x_r|)} - 1$$

Notice that the relation between $Sim(A, \delta_{ji})$ and $Sim(A, \delta_{qp})$ only depends on the distances between the values that define the fuzzy number A and x_k and x_r respectively. Thus, the smaller the distances ($|a_i - x_k|$) and ($|a_i - x_r|$), the bigger the similarities are.

So if we calculate a s_k and a s_r that

$$k = argmax_{i \leq \eta \leq j}\{Sim(s_\eta, \delta_{ji})\},\ s_k = \phi(a_1, a_2)$$
$$r = argmax_{p \leq \tau \leq q}\{Sim(s_\tau, \delta_{qp})\},\ s_r = \phi(b_1, b_2)$$

then s_k is closer (or equal) to δ_{ji} than to δ_{qp} and s_r is closer (or equal) to δ_{qp} than to δ_{ji}, so if $x_k \geq x_r$, then $s_k \geq s_r$ and this proves that $\phi(a_1, a_2) \geq \phi(b_1, b_2)$.

b) Suppose that for $m = 1$, $\phi(a_1, \ldots, a_{m-1}) \geq \phi(b_1, \ldots, b_{m-1})$. Consider the aggregation of m values on A and B:

$$\phi(A) = \phi(a_1, \ldots, a_m) =$$

$$= w_1 \otimes a_1 \oplus (1 - w_1) \otimes C^{m-1}\{\beta_h, a_h, h = 2, \ldots, m\}$$

with $C^{m-1}\{\beta_h, a_h, h = 2, \ldots, m\} = \phi(a_2, \ldots, a_m) = s_j$.

$$\phi(B) = \phi(b_1, \ldots, b_m) =$$

$$= w_1 \otimes b_1 \oplus (1 - w_1) \otimes C^{m-1}\{\beta_h, b_h, h = 2, \ldots, m\}$$

with $C^{m-1}\{\beta_h, b_h, h = 2, \ldots, m\} = \phi(b_2, \ldots, b_m) = s_i$.

By induction hypothesis $\phi(a_1, a_2, \ldots, a_m) = \phi(a_1, s_j)$ and $\phi(b_1, b_2, \ldots, b_m) = \phi(b_1, s_i)$, which reduces it to the case $m = 2$.

As $s_j \geq s_i$ and the following relations are true: $a_1 \geq b_1$, $a_1 \geq s_j$ and $b_1 \geq s_i$, we conclude that $\phi(a_1, s_j) \geq \phi(b_1, s_i)$. Therefore $\phi(a_1, a_2, \ldots, a_m) \geq \phi(b_1, b_2, \ldots, b_m)$. □

Property 2: *The ULOWA operator is commutative, i.e.,* $\phi(a_1, \ldots, a_m) = \phi(a'_1, \ldots, a'_m)$ *where* $(a'_1, \ldots, a'_m)$ *is any permutation of the elements in* $(a_1, \ldots, a_m)$

Proof: The ULOWA operator makes an ordered weighted average of the arguments. If A and A' are the ordered argument vectors of $(a_1, \ldots, a_m)$ and $(a'_1, \ldots, a'_m)$ respectively, then $A = A'$. So $\phi(A) = \phi(A')$. □

Property 3: *The ULOWA operator* ϕ *is idempotent in the sense that if* $\forall j, a_j = a$, *then* $\phi(a_1, \ldots, a_m) = a$

Proof: Following the definition of the ULOWA operator, the final step consists in $\phi(a_{m-1}, a_m) = s_k$ where

$$k = argmax_{i \leq p \leq j}\{Sim(s_p, (x_k, x_k, x_k, x_k)))\}$$

In this case, $p = i = j$ and $s_k = s_i = a$, and recursively we obtain that $\phi(a_1, \ldots, a_m) = a$. □

Property 4: *The ULOWA operator* ϕ *is an "orand" operator. That is, for any weighting vector* W *and ordered labels* $A = [a_1, \ldots, a_m]$ $(a_1 \geq a_2 \geq \cdots \geq a_m)$, *then:*

$$min(a_1, \ldots, a_m) \leq \phi(a_1, \ldots, a_m) \leq max(a_1, \ldots, a_m)$$

Proof: Being $s_j, s_i \in S$ $(j \geq i)$, we have defined the convex combination of these two terms as $s_k = w_1 \otimes s_j \oplus (1 - w_1) \otimes s_i$ [7]. According to (4) $k = argmax_{i \leq p \leq j}\{Sim(s_p, \delta)\}$. Therefore, $i \leq k \leq j$, that is, the resulting label from the combination of two labels ($\phi(s_i, s_j)$) is s_k with $i \leq k \leq j$, proving that $a_m \leq \phi(a_1, \ldots, a_m) \leq a_1$ □

V. Examples

In this section two examples of the ULOWA operator are analysed. The first is a numerical example, and the second illustrates the behaviour of the operator when the fuzzy membership functions of the terms change.

A. Numerical example

Let us take five options to aggregate using the ULOWA operator. The granularity of the fuzzy set is 7 and the definition of the membership functions is shown in Fig. 3b. The weighting vector is $W = (0.2, 0.2, 0.2, 0.2, 0.2)$, which corresponds to the "mean" average. The ordered set of linguistic labels to aggregate is P, H, M, L and VL.

The ULOWA aggregation begins following (3):

$$\phi(P, H, M, L, VL) = W * B^T =$$
$$= C^5\{w_k, b_k, k = 1, \ldots, 5\} =$$
$$= 0.2 \otimes P \oplus (1 - 0.2) \otimes C^4\{\beta_h, b_h, h = 2, \ldots, 5\}$$

Now, we have to evaluate C^4 as:

$$C^4\{\beta_h, b_h, h = 2, \ldots, 5\} =$$
$$= \frac{0.2}{0.8} \otimes H \oplus \left(1 - \frac{0.2}{0.8}\right) \otimes C^3\{\beta_h, b_h, h = 3, \ldots, 5\}$$

,where

$$C^3\{\beta_h, b_h, h = 3, \ldots, 5\} =$$
$$= \frac{0.2}{0.6} \otimes M \oplus \left(1 - \frac{0.2}{0.6}\right) \otimes C^2\{\beta_h, b_h, h = 4, 5\}$$

In the last step, C^2 is calculated as follows:

$$C^2\{\beta_h, b_h, h = 4, 5\} = \\ = \frac{0.2}{0.4} \otimes L \oplus \left(1 - \frac{0.2}{0.4}\right) \otimes VL = s_k$$

Now, the two terms to aggregate are $s_j = L$ and $s_i = VL$. The corresponding centers of gravity (see (1)) of L and VL are $COG(L) = (0.23, 0.33)$ and $COG(VL) = (0.03, 0.33)$. The intermediate point δ is calculated as follows:

$$x_k = x^*_{VL} + w_1(x^*_L - x^*_{VL}) = \\ = 0.03 + 0.5(0.23 - 0.03) = \\ = 0.13$$

The similarities of VL and L with $\delta = (0.13, 0.13, 0.13, 0.13)$ are 0.89 and 0.83, respectively (using (2)). According to (4), the label with maximum similarity is $s_k = VL$, because it is quite near the point 0.13 and has a more precise meaning than L, which is more vague (this is penalized in (4)).

Then, we continue with the same approach to obtain:

$$C^3\{\beta_h, b_h, h = 3, \ldots, 5\} = \\ = \frac{0.2}{0.6} \otimes M \oplus \left(1 - \frac{0.2}{0.6}\right) \otimes VL = L$$

The intermediate point δ is $\delta = (0.19, 0.19, 0.19, 0.19)$. The similarities of VL, L and M with δ are 0.83, 0.86 and 0.68 respectively. Notice that, in this case, the right side of the membership function of L moves its centre of gravity to the right, increasing its similarity with δ. So, the result is L.

The next step is to evaluate C^4:

$$C^4\{\beta_h, b_h, h = 2, \ldots, 5\} = \\ = \frac{0.2}{0.8} \otimes H \oplus \left(1 - \frac{0.2}{0.8}\right) \otimes L = M$$

In this step, $\delta = (0.36, 0.36, 0.36, 0.36)$. The similarities of δ with L, M, AH, and H are 0.79, 0.86, 0.76, and 0.60 respectively. Now, the most similar label to δ is M.

Finally, the last step gives the final result:

$$\phi(P, H, M, L, VL) = \\ = C^5\{w_k, b_k, k = 1, \ldots, 5\} = \\ = 0.2 \otimes P \oplus (1 - 0.2) \otimes M = AH$$

The resulting intermediate point δ is $\delta = (0.59, 0.59, 0.59, 0.59)$. The similarities of δ with M, AH, H, VH and P are 0.90, 0.94, 0.83, 0.69, and 0.61 respectively. The resulting label of the aggregation is AH.

B. Qualitative analysis

In this section we show an example to illustrate the influence of the membership function of the terms in the result of the ULOWA aggregation. Fig. 4 shows three cases aggregating the same labels (VL and AH) with the mean average policy. In each case, we have a different fuzzy membership function for the term L. Notice that in case 1 the result of the aggregation given by the ULOWA operator is

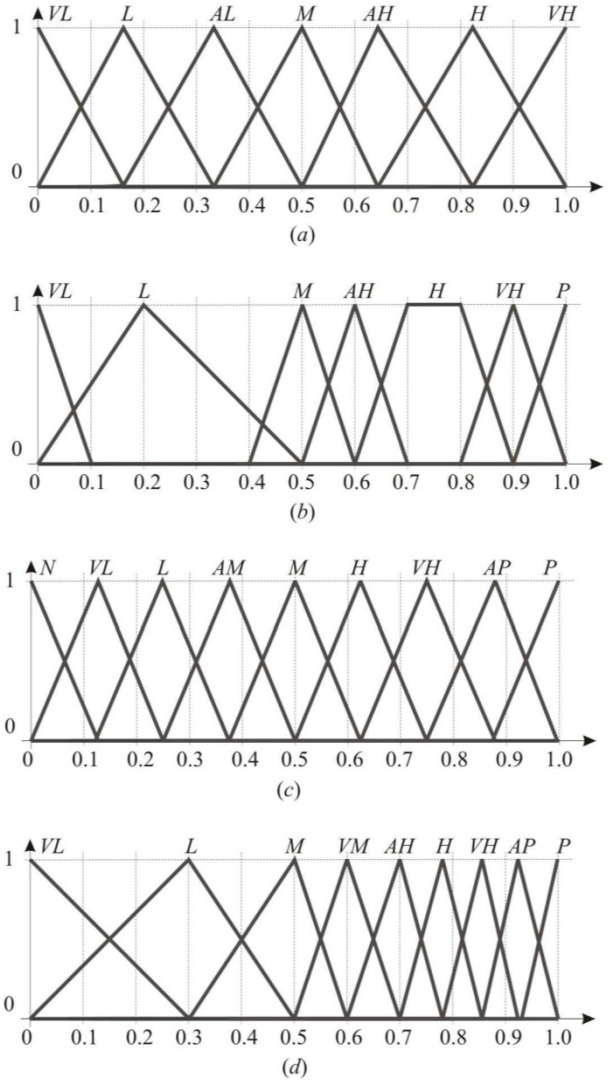


Fig. 3. Fuzzy term sets with 7 and 9 labels: (a) and (c) balanced; (b) and (d) unbalanced.

M, whereas in cases 2 and 3 the result is L. This difference is due to the influence of the membership function of L.

In the first case, the closest label to δ (the intermediate point between the centres of gravity of both labels) is M, because the label L is asymmetric and its centre of gravity is located in the left hand side, that is, it has a more negative meaning than the term L defined in cases 2 and 3. On the other two cases, we obtain the label L because the point of the label with maximum membership is close to δ.

This example shows that ULOWA is sensible to the definition of the semantics of the fuzzy terms, which is especially appropriate when we want to deal with unbalanced sets of terms.

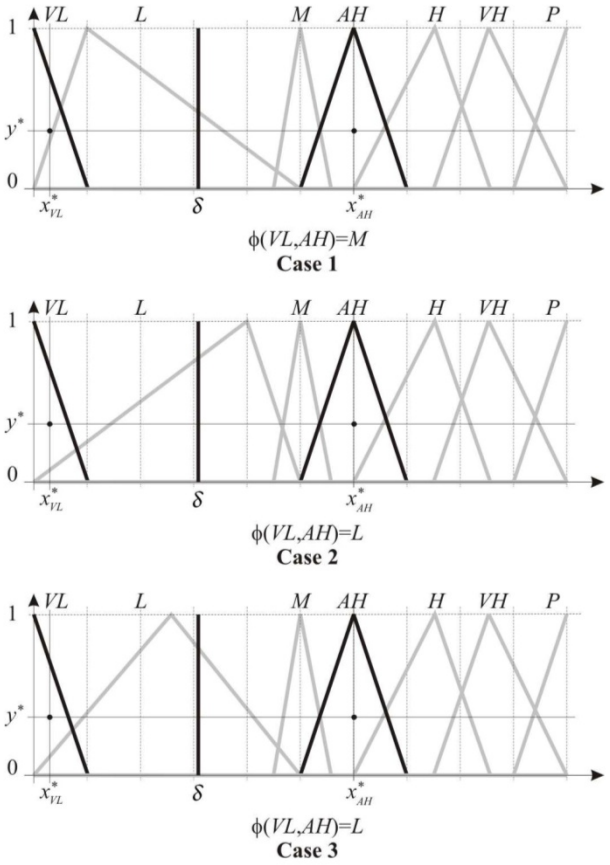


Fig. 4. Examples of ULOWA when aggregating VL and AH with an average weight.

VI. Comparison with other linguistic operators

A. LOWA vs ULOWA

Dealing with linguistic labels is usually made with symbolic algorithms that use the order of terms and negation functions. These approaches do not use the membership functions associated with the labels and, with unbalanced label sets, they exhibit undesirable (or non logical) results.

As stated in section III, the LOWA operator deals with linguistic information taking into account any balanced distribution of terms [10]. The operations are defined at a symbolic level, considering only the position of terms in the set S. For this reason it cannot be applied in the case of unbalanced linguistic term sets.

On the contrary, ULOWA has been specially designed for the unbalanced case. However, it should behave correctly also with balanced sets of terms.

In Table 2 we present the results obtained with LOWA and ULOWA in different situations. This table shows the aggregation of different sequences of labels, with different policies (e.g., mean, as many as possible), as well as taking into account different fuzzy sets (shown in Fig. 3). When applying ULOWA on balanced sets (term sets a and c in Fig. 3), both operators exhibit a very similar behaviour, with the exception of entry 18. This is due to the fact that when dealing with boundary labels on the aggregations (such as VL or VH with 7-label sets), their membership functions are different from the rest, and their COGs are closer to the adjacent label than the rest. Since ULOWA uses the information of the centre of gravity, the result obtained with ULOWA when the boundary labels are aggregated can be different than the result with the LOWA operator.

When dealing with unbalanced sets of terms (sets b and d on Fig. 3), the behaviour of both operators is different. The influence of membership functions, as shown in the previous section, permits to obtain more logical results. For instance, entry 11 aggregates the best (P) and the worst (VL) options with nine labels (set d). The LOWA operator just returns their intermediate label AH, but this result is not logical attending to the used fuzzy set. In this case, ULOWA returns M, the neutral term.

The influence of the decision-makers policy is also compared in Table 2. The policies are sorted according to its corresponding orness [23]: at least half (0.8), mean (0.5), most (0.45), and as many as possible (0.25). With those different aggregation weights and an unbalanced set of terms, the LOWA and ULOWA give 7 different results out of 24 cases (entries from 23 to 38 and from 43 to 46). For example, for entry 46 that aggregates (L, M, M, VH, P, P) with "as many as possible", the results of ULOWA is M instead of L (given by LOWA), because it is able to recognise that M is the neutral value and there are 3 very high evaluations.

B. Unbalanced aggregation operators

Recently, some unbalanced aggregation operators have appeared. In [11] a new model for representing unbalanced linguistic information is defined. It is based on the 2-tuple fuzzy model for semantically interpreting the terms, as well as on a hierarchy of balanced linguistic term sets with different granularity. With the combination of parts of this linguistic hierarchy, one can define the unbalanced linguistic term set by means of the appropriate transformation functions. Then, any of the aggregation operators of linguistic 2-tuples can be used [12].

Although this methodology permits to deal with information from unbalanced sets of terms, its most important drawbacks are the computational complexity of the method and the definition of a new representation model. Moreover, it is assumed that the fuzzy sets are subsumed in the hierarchy of labels (usually with a high number of labels). This assumption is complex to fulfil in some unbalanced and asymmetric sets such as [9] and [13]. On the contrary, the ULOWA operator uses standard membership functions to define each of the linguistic terms, requiring only an order between the terms and the complete coverage of the reference domain. Moreover, our approach supports a wide range of situations, working with any combination of triangular and trapezoidal functions, with different degrees of overlapping, as shown in Figs. 3 and 4.

In [21], the sets of unbalanced terms only consider the case of having the labels uniformly distributed around the

TABLE II
COMPARISON OF LOWA AND ULOWA OPERATORS

Entry	Set[a]	Labels to Aggregate	Policy[b]	LOWA	ULOWA
1	(a)	*VL VH*	*AV*	*M*	*M*
2	(a)	*VL H*	*AV*	*M*	*M*
3	(a)	*VL AH*	*AV*	*AL*	*AL*
4	(a)	*VL M*	*AV*	*AL*	*AL*
5	(a)	*L M*	*AV*	*AL*	*AL*
6	(b)	*VL P*	*AV*	*AH*	*M*
7	(b)	*VL VH*	*AV*	*AH*	*M*
8	(b)	*VL H*	*AV*	*M*	*M*
9	(b)	*VL AH*	*AV*	*M*	*M*
10	(b)	*VL M*	*AV*	*L*	*L*
11	(d)	*VL P*	*AV*	*AH*	*M*
12	(d)	*VL AP*	*AV*	*AH*	*M*
13	(d)	*L AP*	*AV*	*AH*	*VM*
14	(d)	*M AP*	*AV*	*H*	*AH*
15	(a)	*VL L AL AH VH*	*ALH*	*AH*	AH
16	(a)	*VL L AL AH VH*	*AV*	*M*	M
17	(a)	*VL L AL AH VH*	*Most*	*AL*	*AL*
18	(a)	*VL L AL AH VH*	*AMAP*	*L*	*VL*
19	(a)	*VL VL L VH VH*	*ALH*	*H*	*H*
20	(a)	*VL VL L VH VH*	*AV*	*M*	*M*
21	(a)	*VL VL L VH VH*	*Most*	*AL*	*AL*
22	(a)	*VL VL L VH VH*	*AMAP*	*VL*	*VL*
23	(b)	*VL L M H P*	*ALH*	*H*	*H*
24	(b)	*VL L M H P*	*AV*	*AH*	*AH*
25	(b)	*VL L M H P*	*Most*	*M*	*M*
26	(b)	*VL L M H P*	*AMAP*	*L*	*VL*
27	(b)	*VL VL M P P*	*ALH*	*VH*	*P*
28	(b)	*VL VL M P P*	*AV*	*AH*	*AH*
29	(b)	*VL VL M P P*	*Most*	*M*	*M*
30	(b)	*VL VL M P P*	*AMAP*	*VL*	*VL*
31	(b)	*L M M H P P*	*ALH*	*VH*	*VH*
32	(b)	*L M M H P P*	*AV*	*H*	*AH*
33	(b)	*L M M H P P*	*Most*	*AH*	*AH*
34	(b)	*L M M H P P*	*AMAP*	*L*	*M*
35	(d)	*VL VL M P P*	*ALH*	*AP*	*VH*
36	(d)	*VL VL M P P*	*AV*	*AH*	*VH*
37	(d)	*VL VL M P P*	*Most*	*M*	*M*
38	(d)	*VL VL M P P*	*AMAP*	*VL*	*VL*
39	(c)	*VL L L VH P P*	*ALH*	*AP*	*AP*
40	(c)	*VL L L VH P P*	*AV*	*M*	*M*
41	(c)	*VL L L VH P P*	*Most*	*AM*	*AM*
42	(c)	*VL L L VH P P*	*AMAP*	*VL*	*VL*
43	(d)	*L M M VH P P*	*ALH*	*AP*	*AP*
44	(d)	*L M M VH P P*	*AV*	*AH*	*AH*
45	(d)	*L M M VH P P*	*Most*	*VM*	*VM*
46	(d)	*L M M VH P P*	*AMAP*	*L*	*M*

[a] Fuzzy sets depicted in Fig. 3

[b] Decision-maker policies: *Most*; *ALH*="At least half"; *AMAP*="As many as possible"; *AV*="Average"

neutral term, but increasing the separation between two adjacent labels as they are more distant from the neutral term (which leads to asymmetric fuzzy sets). They propose a new representation model for the linguistic labels, where each label is attached a numerical value β that depends on its position in the domain of reference. Some linguistic averaging operators are defined on the basis of numerical operations on the β values.

Those two approaches do not use membership functions when combining labels, because they are operating at a symbolic level, taking into account only the positions of the values. In [26] an algorithm to make the aggregation of n fuzzy sets using the OWA operator is proposed, where the weights are also defined as fuzzy sets. This method, called type-1 OWA, is based on the extension principle and performs the aggregation by making a discretization of the domain of reference of both values and weights. The result is a new fuzzy set, whose membership function is calculated using a T-norm to combine the values for each point. However, the paper does not address the linguistic interpretation of the new fuzzy set. That is, the new fuzzy set may not correspond to any of the original ones, being necessary to define some procedure to create new linguistic terms that could describe the new fuzzy set. This is not necessary in the other approaches, because the result is also in the initial set of terms.

VII. CONCLUSIONS

The reader can find some linguistic approaches to aggregate information, but they do not operate with the fuzzy membership functions of terms. In this paper, we have focused on the LOWA operator, which works at a pure symbolic level, taking into account only the position of terms in the set, assuming a set of balanced terms. Our proposal extends the LOWA definition, including the information provided by fuzzy sets. This is used to decide in each step of the aggregation which label is the most appropriate result according to its semantics. The proposed algorithm is able to work with both balanced and unbalanced fuzzy sets.

The ULOWA operator uses a similarity function to compare fuzzy labels and decide the resulting label of each aggregation step. Some existing similarity functions defined for fuzzy sets do not fit our requirements, so we proposed a new one, based on previous work by Chen [5].

Combining these elements, we have shown how ULOWA is sensitive to the fuzzy membership functions of the labels. This fact gives the user more freedom when defining the sets according to his/her requirements. We have illustrated how this operator works and how it reacts to the change of one of the membership functions.

As seen on section V, in some extreme cases, the results can be slightly different to the ones obtained with LOWA when using a balanced set of terms. Due to this fact, the election between LOWA and ULOWA for aggregating labels in balanced sets should not be tough. It depends on whether one wants to work at a symbolic or semantic level.

In addition, the ULOWA aggregator is based on the OWA operator and permits to customize the results using different well-known decision-makers policies. We also analysed the influence of the weight in the final result, and compared the results obtained in using both aggregators.

Finally, a line of future research is the use of this operator with other kinds of fuzzy sets, for instance Gaussian fuzzy sets. In this sense, other similarity functions should be used [17]. Then, we want to apply it to different problems. On one hand, following a previous work [15], we plan to use this operator in different recommendation systems to include linguistic preferences of the users. On the other hand, it will be studied how this operator can improve the existing techniques for privacy preserving in linguistic data [19].

ACKNOWLEDGEMENTS

This work has been supported by the Universitat Rovira i Virgili (a post-doc grant of D. Isern, a predoctoral grant of L. Marin, the project 2009AIRE-04, and a grant of the Master of Computer Security and Intelligent Systems), Spanish Government (PlanẼ, Spanish Economy and Employment Stimulation Plan) and the Spanish Ministry of Science and Innovation projects: DAMASK (TIN2009-11005) ARES (CONSOLIDER-INGENIO 2010 CSD2007-00004) and eAEGIS (TSI2007-65406-C03-02).

REFERENCES

[1] P. Bonissone and K. Decker, "Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-Off Precision and Complexity," in *Proceedings of the First Conference Annual Conference on Uncertainty in Artificial Intelligence, UAI-85*, L. N. Kanal and J. F. Lemmer, Eds. New York, NY: Elsevier Science, 1985, pp. 217–248.

[2] F. J. Cabrerizo, S. Alonso, and E. Herrera-Viedma, "A Consensus Model for Group Decision Making Problems with Unbalanced Fuzzy Linguistic Information," *International Journal of Information Technology and Decision Making*, vol. 8, no. 1, pp. 109–131, 2009.

[3] S. J. Chen and S. M. Chen, "Fuzzy risk analysis based on similarity measures for generalized fuzzy numbers," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 1, pp. 45–56, 2003.

[4] S. M. Chen, "New methods for subjective mental workload assessment and fuzzy risk analysis," *Cybernetics and systems, an International Journal*, vol. 27, no. 5, pp. 449–472, 1996.

[5] S.-J. Chen, "A New Similarity Measure for Generalized Fuzzy Numbers Based on Geometric-Mean Averaging Operator," in *Proceedings of the 16th IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2006*. Vancouver, CA: IEEE Press, 2006, pp. 1879–1886.

[6] M. Delgado, F. Herrera, E. Herrera-Viedma, and L. Martínez, "Combining Numerical and Linguistic Information in Group Decision Making," *Information Sciences*, vol. 107, no. 1-4, pp. 177–194, 1998.

[7] M. Delgado, J. Verdegay, and M. Vila, "On aggregation operations of linguistic labels," *International Journal of Intelligent Systems*, vol. 8, no. 3, pp. 351–370, 1993.

[8] J. Domingo-Ferrer and V. Torra, "Median-Based Aggregation Operators for Prototype Construction in Ordinal Scales," *International Journal of Intelligent Systems*, vol. 18, no. 6, pp. 633–655, 2003.

[9] J. M. Garibaldi and E. C. Ifeachor, *Fuzzy Systems in Medicine*, ser. Studies in Fuzziness and Soft Computing. Physica-Verlag Heidelberg, 2000, vol. 41, ch. The Development of a Fuzzy Expert System for the Analysis of Umbilical Cord Blood, pp. 652–668.

[10] F. Herrera, E. Herrera-Viedma, and J. Verdegay, "Direct approach processes in group decision making using linguistic OWA operators ," *Fuzzy Sets and Systems*, vol. 79, no. 2, pp. 175–190, 1996.

[11] F. Herrera, E. Herrera-Viedma, and L. Martínez, "A Fuzzy Linguistic Methodology to Deal With Unbalanced Linguistic Term Sets," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 2, pp. 354–370, 2008.

[12] F. Herrera and L. Martínez, "A 2-tuple fuzzy linguistic representation model for computing with words," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 6, pp. 746 – 752, 2000.

[13] T.-P. Hong and J.-B. Chen, "Finding relevant attributes and membership functions," *Fuzzy Sets and Systems*, vol. 103, no. 3, pp. 389–404, 1999.

[14] C. H. Hsieh and S. H. Chen, "Similarity of generalized fuzzy numbers with graded mean integration representation," in *Proceedings of the 8th International Fuzzy Systems Association World Congress, IFSA 99*, vol. 2, Taipei, Taiwan, Republic of China, 1999, pp. 551–555.

[15] D. Isern, A. Valls, and A. Moreno, "Using aggregation operators to personalize agent-based medical services," in *Proceedings of the 10th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, KES 2006*, ser. Lecture Notes in Artificial Intelligence, B. Gabrys, R. J. Howlett, and L. C. Jain, Eds., vol. 4252. Bournemouth, UK: Springer Berlin / Heidelberg, 2006, pp. 1256–1263, cP.

[16] J. S. Kim, B. A. Sohn, and B. G. Whang, "A tolerance approach for unbalanced economic development policy-making in a fuzzy environment," *Information Sciences*, vol. 148, no. 1-4, pp. 71–86, 2002.

[17] D.-C. Lin and M.-S. Yang, "A similarity measure between type-2 fuzzy sets with its application to clustering," in *Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007*, J. Lei, J. Yu, and S. Zhou, Eds., vol. 1. Haikou, Hainan, China: IEEE Computer Society, 2007, pp. 726–731.

[18] L. Martínez, M. Espinilla, and L. Pérez, "Sensory Evaluation Model with Unbalanced Linguistic Information," in *Proceedings of the International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2007*, T. Li, Y. Xu, and D. Ruan, Eds. Chengdu, China: Atlantis Press, 2007.

[19] V. Torra, "OWA operators in data modeling and re-identification," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 5, pp. 652–660, 2004.

[20] Z. Xu, *Fuzzy Sets and Their Extensions: Representation, Aggregation and Models - Intelligent Systems from Decision Making to Data Mining, Web Intelligence and Computer Vision*, ser. Studies in Fuzziness and Soft Computing. Springer-Verlag Berlin Heidelberg, 2008, vol. 220, ch. Linguistic Aggregation Operators: An Overview , pp. 163–182.

[21] Z. Xu, "An Interactive Approach to Multiple Attribute Group Decision Making with Multigranular Uncertain Linguistic Information," *Group Decision and Negotiation*, vol. 18, no. 2, pp. 119–145, 2009.

[22] R. R. Yager, "Including Importances in OWA Aggregations Using Fuzzy Systems Modeling," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 2, pp. 286–294, 1988.

[23] R. R. Yager, "On Ordered Weighted Averaging Aggregation. Operators in Multicriteria Decision making," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 183–190, 1988.

[24] R. R. Yager, "Fusion of ordinal information using weighted median aggregation ," *International Journal of Approximate Reasoning*, vol. 18, no. 1-2, pp. 35–52, 1998.

[25] D. Yong, S. Wenkang, D. Feng, and L. Qi, "A new similarity measure of generalized fuzzy numbers and its application to pattern recognition," *Pattern Recognition Letters*, vol. 25, no. 8, pp. 875–883, 2004.

[26] S. Zhou, F. Chiclana, R. John, and J. Garibaldi, "Type-1 OWA operators for aggregating uncertain information with uncertain weights induced by type-2 linguistic quantifiers," *Fuzzy Sets and Systems*, vol. 159, no. 24, pp. 3281–3296, 2008.

Web-based Recommender using Linguistic Preferences

Lucas Marín, David Isern, Antonio Moreno, Aida Valls

Intelligent Technologies for Advanced Knowledge Acquisition Research Group (ITAKA)
Departament d'Enginyeria Informàtica i Matemàtiques
Universitat Rovira i Virgili, Tarragona, Catolonia (Spain)
{lucas.marin,david.isern,antonio.moreno,aida.valls}@urv.cat

Abstract. In this work we propose a recommender system that permits to represent a user's profile including qualitative and quantitative criteria. The system has been implemented as a Web application. It is designed in a generic way so that it can be applied to any decision making problem. The alternatives are aggregated using a linguistic averaging operator. Then, they are ranked and presented to the user. The system includes some predefined sets of linguistic terms to be used to represent qualitative preferences. Some unbalanced sets have been defined. The ULOWA operator is proposed to make the aggregation in balanced and unbalanced sets of terms. The recommender has been designed to permit the update of the user's preferences in an unsupervised way. The evolution of the user's profile is based on the information provided by his selections, taking into account both the alternative selected and the set of discarded alternatives.

Keywords: Web-based recommender; linguistic preferences; linguistic aggregation operators; automatic user's profile maintenance.

1 Introduction

The user preferences may evolve during time due to changes in the personal conditions. To adapt the user profile over time, some information about how the user is interacting with the system is needed. The two most common ways to obtain this information, known as *relevance feedback* [1] are (1) to use information given *explicitly* by the user and (2) to get the information *implicitly* by observing the user's behaviour. Some *hybrid* systems combine implicit and explicit approaches [2].

Explicit feedback is obtained when users are required to explicitly evaluate items, indicating how relevant or interesting an item is to the user.

Implicit feedback is obtained by monitoring the user's actions in order to automatically infer the user's preferences, following the idea that user's preferences can be inferred from his behaviour.

Explicit feedback is sometimes not desirable because it has some serious limitations. One of those is that numeric scales may not be adequate for describing the reactions humans have to items. Moreover, as studied in [3], users are reluctant to

spend time giving explicit feedback and only 15% of the users would supply it even they were encouraged to do so.

For this reason, we are designing a recommender system based on MCDA techniques, which adapts the personal profile of the users in an implicit way. We assume that the system is used regularly, so we can learn, from the usage history, how the preferences of the decision maker evolve. The goal is to obtain the appropriate recommendation over time by adapting the preference functions.

In this paper, it is presented the example of a tourist that uses our recommender system to find a suitable destination for holidays. In this case, the recommender system knows the preferences of the user with respect to different subjects (*i.e.*, interest on cities that have an airport, preference for the different continents, preferred climate). The user profile includes a set of value functions that permit to rate the values of the attributes according to the user preferences. Preferences could be given in a numerical or linguistic scale.

The recommendation is displayed, in a Web interface, as a list of alternatives ordered according to the user's preferences. The score of each alternative is obtained from the aggregation of the partial preferences given by each criterion, by means of the Unbalanced-LOWA operator (ULOWA). ULOWA is an extension of the Linguistic Ordered Weighted Averaging Operator (LOWA) [4], specially suitable when unbalanced sets of linguistic terms are used.

Once the list of recommendations is presented to the user, the proposal that best fits with the current interests is selected. If the user has not selected the first proposal, an adaptation mechanism is activated in order to learn from the selection made by the user in an unsupervised way. The rationale is that when the user does not choose the first alternative, the profile is not modelling correctly the real preferences of the user. The architecture of the system is presented in Fig. 1.

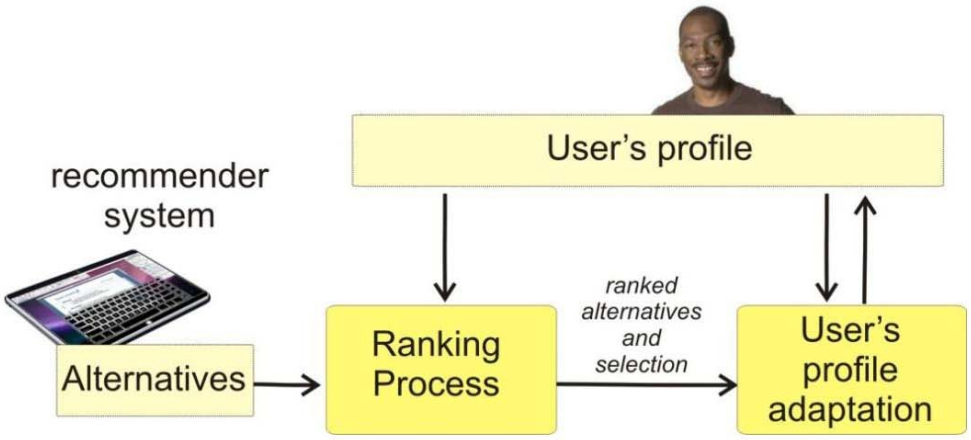


Fig. 1. Recommender System including Ranking and Adaptation

The rest of the paper is organised as follows. The next section explains the design of the recommender system from a Multiple Criteria Decision Aid perspective. Section 3 explains the aggregation operator defined to be used for dealing with

balanced and unbalanced sets of linguistic preferences. Section 4 summarises how the developed recommendation web platform works. Finally, section 5 gives some conclusions and devises the research lines we are working on.

2 Recommender system

The goal of the recommendation process is to rate and rank a set of alternatives introduced by the user, in order to permit him to distinguish between the good and bad options. Each alternative is defined by a set of values for some attributes. Those values must be evaluated with respect to the user's preferences in order to obtain the corresponding preference scores.

In our system we consider two types of attributes:

- **Quantitative measurement attributes**: take numerical values within a given range. For example, the distance from the city centre to the nearest airport, measured in kilometres.
- **Categorical attributes**: they have a predefined set of possible categories (i.e. values) that describe the alternative according to some property or feature. For example, the continent or the climate type of the city (with values such as *humid-continental*, *humid-subtropical*, *tropical-savanna*, *alpine*, *mediterranean*, ...).

To build preference criteria over those attributes, some preference value functions are needed. The preference functions of each user are stored in his personal profile, which is managed by the recommender system.

When an alternative must be evaluated, the preference functions are used to obtain the preference score of each attribute according to some particular user profile. Once the value of the preference criteria has been obtained, an aggregation operator is used to calculate a global preference score (following the Multi-attribute Utility Theory approach [5]). These scores are used to rank the options.

Two types of criteria have been defined:

- **Numerical criteria:** preference values are given in the [0..1] range.
- **Linguistic criteria:** preferences are given in a set of ordered linguistic terms.

The next subsections give more details about how criteria are built from the different types of attributes.

2.1 Categorical attributes

A fuzzy linguistic approach is proposed for making assessments in a qualitative setting [6]. Linguistic preferences are defined as a set of fuzzy numbers, each one

representing a value of preference. A linguistic preference scale permits to deal with uncertainty in the evaluation of the preferences in categorical variables.

A common fuzzy linguistic scale, *S*, is defined and the decision maker can attach each of the available linguistic terms to each of the possible values of the attribute. Fig. 2. depicts an unbalanced set of preferences with seven possible values of preference (*VL*: Very Low, *L*: Low, *M*: Medium, *AH*: Almost High, *H*: High, *VH*: Very High, *P*: Perfect).

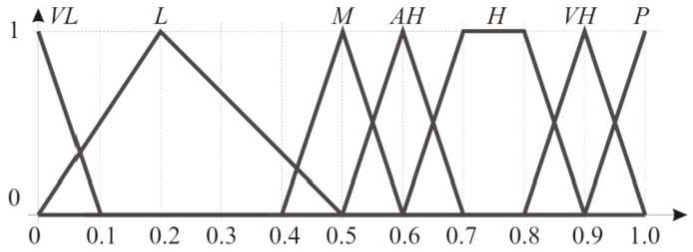


Fig. 2. Linguistic preferences label set

The linguistic scale $S = \{s_i\}, i\epsilon\{0, \ldots, T\}$ is a finite and totally ordered set of terms on a reference domain X=[0,1], with an odd cardinality. One of the labels corresponds to the neutral value and the remaining terms are placed around it. The cardinality of the set must be small enough so as not to impose useless precision and rich enough in order to allow an appropriate discrimination level. The usual cardinality value is 7 or 9.

In our case, we consider that all labels are represented by linear functions, which are defined by four points (a, b, c, d). Then, we can distinguish four different cases: a crisp value if $a = b = c = d$, a trapezoid if $a \neq b \neq c \neq d$ (for example *H* in Fig. 2), a triangular if $a \neq b = c \neq d$ (for example *L* in Fig. 2), and a rectangular if $a = b, c = d$ and $b \neq c$.

Then, the preference function for categorical values consists on a mapping from the set of possible values of a categorical variable, *X*, into the set of preference terms *S*.

$$g_{Ling} : X \rightarrow S$$

The mapping is initially given by the user, and then it can be adapted using the learning algorithm included into the recommender system.

So, in the "Tourist destination" example case, a user may express a "Low" preference for the "Alpine" value and "Very High" for the "Mediterranean" value in the "Destination climate" criterion.

2.2 Quantitative criteria

Quantitative attributes are evaluated using a numerical value function that gives a score in the [0,1] range, with 1 representing the maximum preference degree and 0 the minimum [7]. The value function assumes that the decision maker has a maximum

preference for a single value of the reference domain of the variable, which is denoted as vpref. The preference function is defined symmetrically, that is, the increasing and decreasing rate (left and right sides of vpref) are equal. The parameter d indicates when the preference must have a 0 value. Finally, the form of the function is defined by the parameter m, as shown in Fig. 3.

Formally, let x be the criteria value, v_{pref} the most preferred value for this attribute, m the slope of the utility function and d its width. Then, the preference function u for the value x can be expressed as:

$$v(x) = 1 - \left(\frac{|x - v_{pref}|}{d}\right)^m$$

A graphical representation of the numerical value function can be observed on Fig. 3.

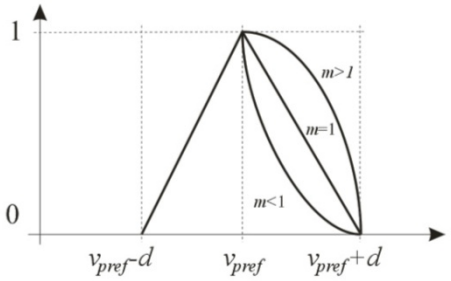


Fig. 3. Numerical utility function representation

Following the example of recommending touristic destinations, the user can define that he prefers a place located in the mountain, indicating that the preferred altitude is 1000 meters. Using the defined model, the preferences will decrease as the altitude is below 1000m and also if it is higher than 1000m. If d=700, this means that cities that are at 300m from the sea level, will be evaluated with $v(300)=0$, and cities above 1700m too, $v(1700)=0$.

Although the users' preference profile stores and manages this numerical value functions, during the aggregation stage, the numerical preference scores are translated to a common linguistic domain (S). This translation is needed because the aggregation operator requires that all the values belong to the same scale. This translation is made using a numeric-to-linguistic function defined in [8]. However, in the adaptation phase, the numerical preference values are used.

3 Unbalanced Linguistic Ordered Aggregation Average Operator (ULOWA)

As commented before, the decision making process has been designed following the Multi-attribute Utility Theory Approach. So, the process is divided in two stages: rating and ranking. There exist several methods to aggregate numerical values as well

as linguistic [6][9]. The family of Ordered Weighted Average (OWA) operators are in the class of mean operators, which are idempotent, monotonic and commutative [10] [11]. The LOWA aggregation operator [4] is an extension of the OWA operator to deal with linguistic attributes defined by Yager [10]. OWA considers a set of weights that specify the decision-maker policy, so that we can emphasize different arguments based upon their ordered position. Weights are used to define different conjunction/disjunction aggregation models [12].

In LOWA, a set of equally-informative (*balanced*) terms is assumed. That is, the membership functions of the fuzzy terms are symmetric and the terms are uniformly distributed around a mid (*i.e.*, neutral) term. For example, when evaluating the users' satisfaction on some topic, linguistic labels like "very low", "low", "almost low", "medium", "high", "very high", "perfect" can be used [13]. However, recently many authors have noticed that there are some problems that require a more flexible definition of the set of linguistic terms [14][15]. In some cases, fuzzy sets should be asymmetric or more positive than negative terms are needed (or vice-versa), leading to an *unbalanced* set of labels. An example is the set of terms used in grading evaluations S ={*F,D,C,B,A*} (being *D* the neutral; *A*, *B* and *C*, positive marks, and *F* – fail – the unique negative value). Many other real-life applications also make use of unbalanced terms, such as [14][16][17]. In this work, we are interested on defining scales with more positive than negative values. This fact permits to be more precise in the positive evaluations, so that we can model better the preferences of the user.

To deal with the unbalanced case, we have defined an adaptation of the LOWA operator, called ULOWA (Unbalanced Linguistic Ordered Weighted Average). ULOWA exploits the knowledge given by the membership functions during the aggregation procedure. This approach, although it is focused on managing unbalanced label sets, can also be used with balanced label sets giving the same results than the LOWA operator.

Let $A = \{a_1, \dots, a_m\}$ be a set of labels, where $a_j \in S$, that have to be aggregated with respect to a set of weights $W = \{w_1, \dots, w_m\}$ such that $w_i \in [0,1]$ and $\sum w_i = 1$.

The ULOWA operator, ϕ, is defined as:

$$\phi(a_1, \dots, a_m) = W \cdot B^T = C^m\{w_k; b_k; k = 1, \dots, m\} = w_1 \otimes b_1 \oplus (1 - w_1) \otimes C^{m-1}\{\beta_h; b_h; h = 2, \dots, m\}$$

Where $\beta_h = w_h / \sum_2^m w_h$, $h = \{2, \dots, m\}$, and $B = \{b_1, \dots, b_m\}$ is a permutation of the elements of A, such that $B = \sigma(A) = \{a_{\sigma(1)}, \dots, a_{\sigma(m)}\}$, where $a_{\sigma(j)} \leq a_{\sigma(i)}, \forall\, i \leq j$.

C^m is the convex combination operator of m labels. If $\exists j \in \{1, \dots, m\}, w_j = 1$ and $\forall k \in \{1, \dots, m\}, k \neq j, w_k = 0$, then $C^m\{w_i; b_i; i = 1, \dots, m\} = b_j$.

When m=2, with $b_1 = s_j$ and $b_2 = s_i$ such that $s_j, s_i \in S(j \geq i)$ then

$$C^2\{w_i; b_i; i = 1,2\} = w_1 \otimes s_j \oplus (1 - w_1) \otimes s_i = s_k,$$

$$\text{with } k = \text{argmax}_{i \leq p \leq j}\{Sim(s_p, \delta)\}$$

Where δ is an intermediate crisp number defined as $\delta = (x_k, x_k, x_k, x_k)$ with $x_k = x^*_{s_i} + w_1(x^*_{s_j} - x^*_{s_i})$.

This x^* value is obtained from the centre of gravity (COG)[18]. It is represented as $COG(A) = (x_A^*, y_A^*)$ as follows:

$$y_A^* = \begin{cases} \frac{1}{6}\left(\frac{a_3 - a_2}{a_4 - a_1} + 2\right) & \text{if } a_1 \neq a_4 \\ \frac{1}{2} & \text{if } a_1 = a_4 \end{cases}$$

$$x_A^* = \frac{y_A^*(a_3 + a_2) + (a_4 + a_1)(1 - y_A^*)}{2}$$

The similarity function needed to compare two fuzzy numbers is the following one:

$$Sim(A, B) = \left[\sqrt[4]{\prod_{i=1}^{4}(2 - |a_i - b_i|)} - 1\right]$$

Fig. 4 illustrates the aggregation of the two extreme labels (VL and P) with a "mean" policy. The figure shows the COGs of both labels and their intermediate crisp number δ used to find the result of the aggregation. After the application of the similarity function, the resulting label is the neutral term M.

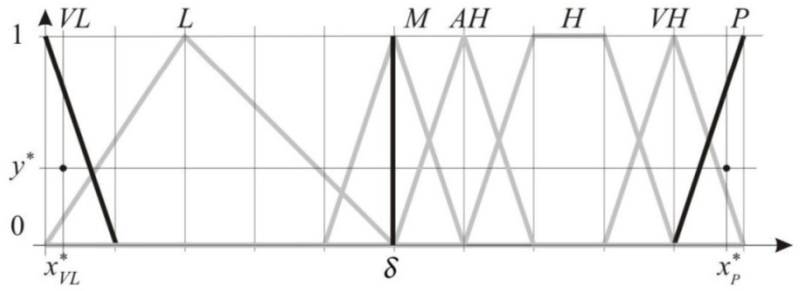


Fig. 4. Graphical example of ULOWA when aggregating two labels (VL and P) with a mean average

After using the ULOWA operator to aggregate the information of the whole set of alternatives, all of them are ranked according to the final value of preference obtained and shown to the user. If the profile contains the correct information about the user's preferences, the first ranked alternative will be the user's selection. However, the user may choose another alternative. This fact would indicate that the profile does not contain accurate information or that the user has changed his interests. This is when the learning process needs to adapt the user profile in order to give a more accurate recommendation the next time that the user employs the recommender system.

4 Web-based recommender

In order to test and evaluate our recommender method, we implemented a web platform. The access to the platform is regulated by a user name and a password, allowing the maintenance of multiple users with different profiles for each recommendation domain.

Users can upload data files with information about the alternatives that must be analysed in order to make a decision. The platform stores that information in a database, letting users to define a context which uses it.

The whole recommendation process consists in the following steps:

- Definition of the problem
- Definition of the initial profile
- Alternatives ranking
- Results evaluation and profile adaptation

4.1 Definition of the context

Each user of this platform has his identifier and password and uses it to log into the system. The first step of the recommendation process is to define all the variables and data that should be used (e.g., recommendation of touristic places, or restaurants, or medical centres). The context includes the following parameters:

- Name of the context.
- Data file: the file previously uploaded to the platform which contains the alternatives data (e.g. "touristic destinations" or "medical centres").
- Numerical variables parameters m and d: parameters of the numerical utility function described previously (see Fig. 3).
- Categorical variables parameters: the number of labels to express the degree of preference and their distribution (see Fig. 2).

This structure enables using the same data file on multiple contexts with different parameters.

4.2 Definition of the initial profile

After the previous step it is necessary for the user to have a profile associated to this context. On the profile creation page (see Fig. 5), all the attributes of the context alternatives are shown in order to be initialised.

The user must declare a preference value for the numerical attributes (*e.g.* preference value of "0" for the attribute "Distance to the nearest airport") as well as select a preference label for each possible value of the linguistic attributes (*e.g.* preference label of "Very high" for the value "Italian" and "Very low" for the value "German" on the attribute "Language").

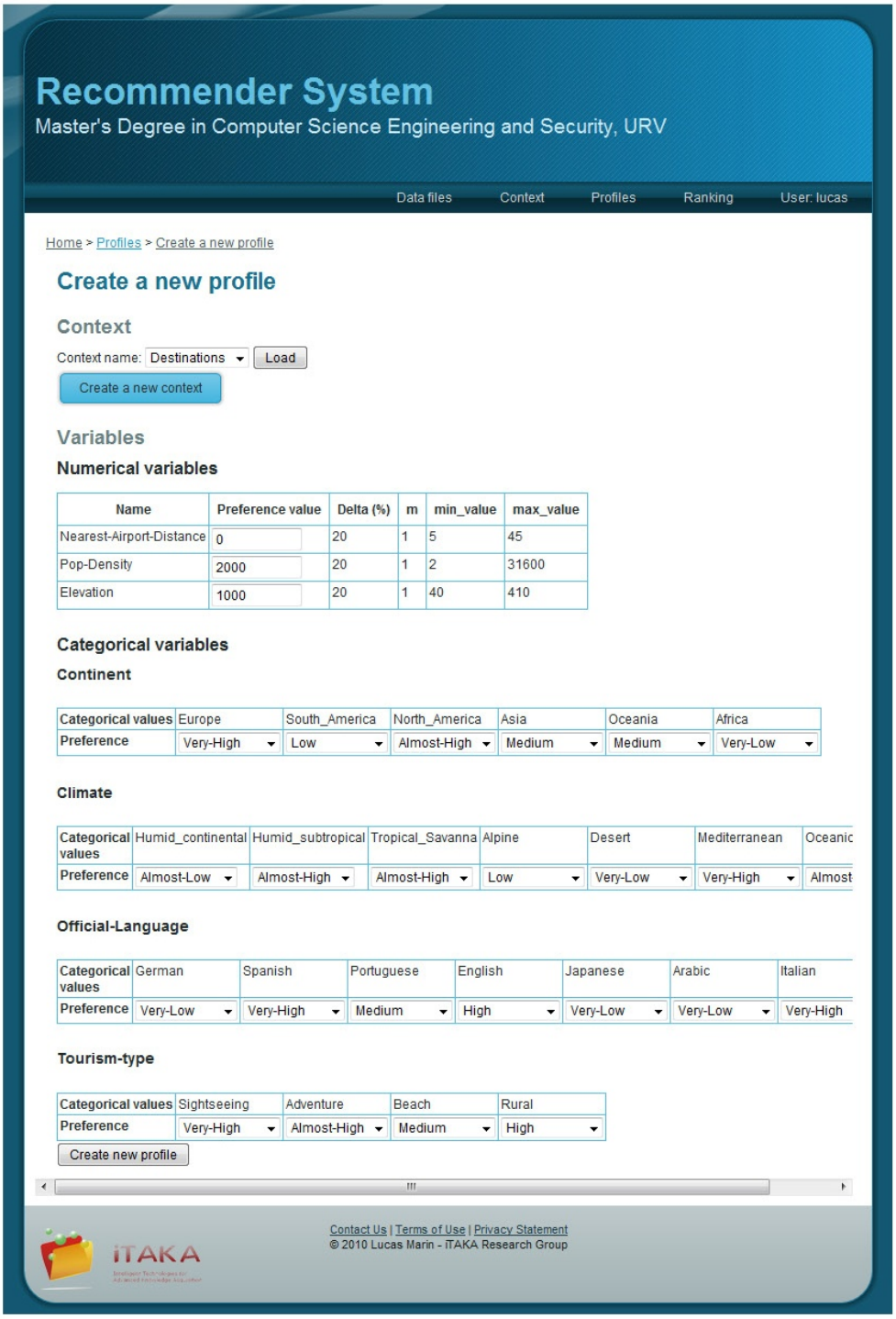


Fig. 5. Profile creation page screenshot

When the profile is created, the user associates it with its identifier and can start using it in the recommendation process.

4.3 Ranking of alternatives

The recommender system uses the ULOWA aggregation process explained before to aggregate the alternatives. Afterwards, a ranking based on the final linguistic ratings is done.

Using this aggregation operator it is crucial to establish correctly the *aggregation policy* to apply. Aggregation policies are related to the degree of *orness/andness* that will be considered in relation to the simultaneous fulfillment of the different criteria. If a high degree of *andness* is considered, this means that we are taking a *risk averse* policy, because an alternative will only obtain a high score if it simultaneously fulfills most of the criteria. Otherwise, if a high level of *orness* is selected, we are in an *optimistic* policy and we consider that only fulfilling some of the criteria is enough to obtain a high global preference score.
In our system, the following policies are available: *most*, *at least half*, *as many as possible*, *average* and *worst* [10].

4.4 Results evaluation and profile adaptation

After selecting the aggregation policy, the recommender system ranks all the alternatives of the context and shows them all in order to the user (see Fig. 6). Here the user may choose which is his final decision. If this selection is not the first one, it means that the recommender has not given an accurate result. Then is when the learning process is triggered and adapts the profile in order to give a better result next time the system is used.

The goal is to modify the user's profile, so that if the ranking receives alternatives previously selected by the user, these alternatives will become better ranked than in the past. Our approach considers as relevance feedback the selection the user does with the results of the recommender, inferring why the alternatives ranked above where overvalued by the recommender or why the selected alternative was undervalued.

The learning process is still in development but the main idea is to separate the final ranking of alternatives in three groups: the alternatives that were ranked over the selected alternative, the selected alternative itself, and the alternatives ranked under the selected one.

The user's profile contains numerical and linguistic preferences, which are managed in a similar way. The adaptation is based on finding the characteristics that are repeated in many alternatives of the first group but not in the selected alternative. For example, we are recommending a tourist destination to the user; from the set of overrated alternatives we find that the value "Rural" of the "Tourism type" attribute is repeated many times, but in the selected alternative, the attribute "Tourism type" has the value "Sightseeing". This is probably indicating that the user wants to do

sightseeing tourism rather than visiting a rural zone, so preference label for those values may need to be adjusted, probably reducing the "Rural" level of preference.

Furthermore, attribute values that only appear on the selected alternative are potentially the ones which made the user to choose that alternative and may need to be considered to modify the profile.

Another aspect that is being considered is to maintain a history of changes of the profile. Initially, changes to the profile will have a low impact but if over the user interaction these changes are done repeatedly, the profile change intensity will be greater.

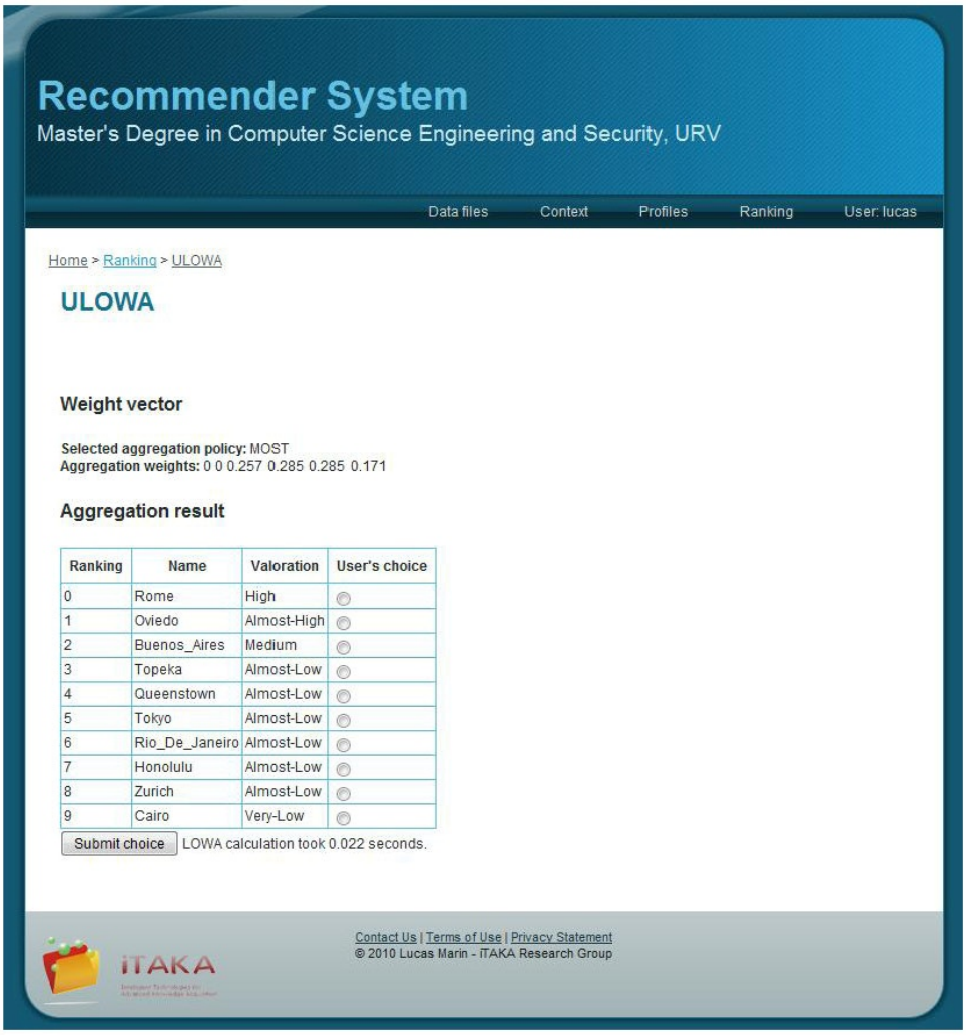


Fig. 6. Ranking result page screenshot

5 Conclusions and future work

The web-based recommender system proposed in this paper has been designed as a general framework that permits to solve decision problems using MCDA techniques. It allows working with numeric and categorical variables at the same time.

The separation between the domain-dependant (*e.g.*, context, and profile) and the domain-independent data (*e.g.*, aggregation policies, value functions parameters) permits to different users re-use data files changing their profile.

The system has been prepared to make automatic simulations that will permit to study the influence of each parameter or the behaviour of a user.

The creation of this platform is the first step in a project that aims to develop techniques to adapt the user profile according to the selections made in the last step of the recommendation process. We consider that having automatic algorithms to update the preferences is required in many domain applications, where the preferences may change dynamically over time.

We are currently studying how we can extract information from the user selected alternative and the better ranked ones and how this information can affect the user profile and to what extent.

As stated before, this adaptation is focused on working in environments in which the user requires recommendations quite often, such as recommendation of films or restaurants, or even recommendations for tourists. Probably, this framework is not so adequate for punctual decisions such as “buying a car”.

Acknowledgements

This work has been supported by the Universitat Rovira i Virgili (predoctoral grant of L. Marin, post-doctoral grant of D. Isern, and the project 2009AIRE-04), the Spanish Government (*PlanE*, Spanish Economy and Employment Stimulation Plan) and the Spanish Ministry of Science and Innovation (DAMASK project, TIN2009-11005).

References

[1] B. Roy, "Paradigms and Challenges," in *Multiple Criteria Decision Analysis.*: Springer New York, 2005.

[2] D. M. Nichols, "Implicit Rating and Filtering," in *Proceedings of 5th DELOS Workshop of Filtering and Collaborative Filtering*, 1997, pp. 31-36.

[3] M. Pazzani and D. Billsus, "Learning and Revising User Profiles: The Identification of Interesting Web Sites," *Machine Learning*, vol. 27, pp. 313-331, 1997.

[4] F. Herrera and E. Herrera-Viedma, "Linguistic decision analysis: steps for solving decision problems under linguistic information," *Fuzzy Sets and Systems*, vol. 115, no. 1, pp. 67-82, 2000.

[5] R. L. Keeney and H. Raiffa, "Decision with multiple objectives: preference and value tradeoffs.," Cambridge University Press, New York, 1993.

[6] M. Grabisch, S. A. Orlovski, and R. R. Yager, "Fuzzy aggregation of numerical preferences," in *The Handbook of Fuzzy Sets Series*, R. Slowinski, Ed.: Kluwer Academic, 1998, pp. 31-68.

[7] J. S. Dyer, "MAUT - Multiattribute utility theory," in *Multiple Criteria Decision Analysis.*: Springer, 2008.

[8] M. Delgado, F. Herrera, E. Herrera-Viedma, J. L. Verdegay, and M. A. Vila, "Aggregation of Linguistic Information Based on a Symbolic Approach," in *Computing with Words in Information/Intelligent Systems I. Foundations*, L. A. Zadeh and J. Kacpryzk, Eds. Heidelberg: Physica-Verlag, 1999, pp. 428-440.

[9] T. Calvo, A. Kolesárová, M. Komorníkova, and R. Mesiar, "Aggregation Operators: Basic Concepts, Issues and Properties," in *Aggregation Operators: New Trends and Applications*, T. Calvo, G. Mayor, and R. Mesiar, Eds.: Springer Studies in Fuzziness and Soft Computing, 2002, pp. 3-106.

[10] R. R. Yager, "On Ordered Weighted Averaging Aggregation. Operators in Multicriteria Decisionmaking," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 183-190, 1988.

[11] V. Torra, "The Weighted OWA Operator," *International Journal of Intelligent Systems*, vol. 12, no. 2, pp. 153-166, 1997.

[12] R. R. Yager, "Including Importances in OWA Aggregations Using Fuzzy Systems Modelling," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 286-294, 1988.

[13] Z. Xu, "Linguistic Aggregation Operators: An Overview," in *Fuzzy Sets and Their Extensions: Representation, Aggregation and Models - Intelligent Systems From Decision Making to Data Mining, Web Intelligence and Computer Vision.*: Springer-Verlag Berling Heidelberg, 2008, pp. 163-182.

[14] Z. Xu, "An Interactive Approach to Multiple Attribute Group Decision Making with Multigranular Uncertain Linguistic Information," *Group Dec. Negotiation*, vol. 18, pp. 119-145, 2009.

[15] F. Herrera, E. Herrera-Viedma, and L. Martínez, "A Fuzzy Linguistic Methodology to Deal With Unbalanced Linguistic Term Sets," *IEEE Trans. Fuzzy Syst.*, vol. 16, pp. 354-370, 2008.

[16] L. Martínez, M. Espinilla, and L. G. Pérez, "Sensory Evaluation Model with Unbalanced Linguistic Information," in *Proc. of International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2007*, Chengdu, China, 2007.

[17] J. S. Kim, B. A. Sohn, and B. G. Whang, "A tolerance approach for unbalanced economic development policy-making in a fuzzy environment.," *Inf. Sciences*, vol. 148, pp. 71-86, 2002.

[18] S. J. Chen and S. M. Chen, "Fuzzy risk analysis based on similarity measures for generalized fuzzy numbers," *IEEE Trans. Fuzzy Syst.*, vol. 11, pp. 45-56, 2003.

[19] B. S. Ahn, "The Uncertain OWA Aggregation with Weighting Functions Having a Constant Level of Orness," *International Journal of Intelligent Systems*, vol. 21, no. 5, pp. 469-483, 2006.

Unsupervised Adaptation of a User Profile

Lucas Marin and David Isern and Antonio Moreno [1]

Abstract. The paper shows an algorithm to deal with user preferences using linguistic criteria. The system has been implemented as a Web application and designed in a generic way so that it can be applied to any decision making problem. The system is composed by two separate modules: a module to rate and rank all alternatives received by the system according to the current interests of the user, and a module to adapt the current user profile in an unsupervised fashion collecting implicit information about the user interaction with the system. The paper, focused in the second step, presents some results and discusses the influence of several parameters introduced in the algorithm.

1 INTRODUCTION

To create and maintain the user profile, the system needs relevant information about the user interests. When users interact with a computer, they provide a great deal of information about themselves. Successful interpretation of these data streams is required to tailor systems to each individual's behaviour, habits and knowledge [5]. In addition, users interests about an issue may evolve during time due to changes in their personal conditions (*e.g.*, change of location, having a child).

Learning and adapting profiles are widely studied research areas. The user profile may be learnt by extracting patterns from training data, *e.g.*, using clustering techniques. Concerning the adaptation of the profile, some information about how the user is interacting with the system is needed. The two most common ways to obtain this information, known as relevance feedback [10], are to use information given *explicitly* by the user and to get the information *implicitly* by observing the user behaviour. Some *hybrid* systems combine implicit and explicit approaches [6].

In the one hand, explicit feedback is obtained when users are required to explicitly evaluate items, indicating how relevant or interesting an item is to the user in a numeric or linguistic scale. These systems, such as [7] and [8] offer high performance and simplicity. On the other hand, implicit feedback is obtained by monitoring the user actions and automatically inferring the user preferences. The amount of data collected is very large, the computation needed to derive recommendations for adaptations is extensive, and the confidence in the suitability of these adaptations is likely to be relatively low. This approach has been less explored, although some existing works (*e.g.* [1], [3]) show promising results [5].

Explicit feedback is sometimes not desirable because it has some serious limitations. One of those is that numeric scales may not be adequate for describing the reactions humans have to items. Moreover, as studied in [9], users are reluctant to spend time giving explicit feedback and only 15% of the users would supply it even if they were encouraged to do so. For these reasons, we present an algorithm that adapts the personal profile of the users in an implicit way. We assume that the system is used regularly, so that it can learn, from the usage history, how the preferences of the decision maker evolve. The main goal is to obtain the appropriate recommendation over time by adapting the preference functions.

The recommender system receives a set of alternatives composed by several criteria. The user profile stores the preferences of the user over all those criteria. The first step of the recommender is to rate all the alternatives taking into account the user profile in order to prioritize them according to his particular circumstances (this step is out of the focus of this paper but the reader can find more information in [4]). Then, the user selects the most appropriate alternative from this sorted list. Afterwards, the adapting algorithm collects the information provided by this selection, the set of non selected alternatives, and all past selections, to infer which changes can be made to the current profile. As it will be described later, we have implemented a Web-based platform that permits to simulate all those stages and extract some conclusions.

The user preferences have been modelled using fuzzy linguistic variables [11]. An example is the set of terms used in grading evaluations $S = \{F, D, C, B, A\}$ (being D the neutral; A, B and C, positive marks, and F - fail - the unique negative value). This kind of data has been widely used for making assessments in a qualitative setting due to the vagueness of the managed information and to facilitate their readability and usage.

The rest of the paper is organised as follows. Section 2 briefly explains the information stored in a user profile. Section 3 explains the algorithm of adaptation of the user profile. Then, section 4 presents the Web-based platform implemented to test all recommender settings and discusses the performance of the system, paying special attention on how the user profile evolves. Finally, section 5 gives some conclusions and comments the research lines we are working on.

2 REPRESENTATION OF THE USER PREFERENCES

The user profile stores the user preferences about some criteria using linguistic variables (S). The selection of the set of labels S and its membership functions is an important decision that influences the final results. We tested different scenarios with both balanced and unbalanced sets of terms (see [4]). Fig. 1 shows a fuzzy term set S with nine unbalanced labels using triangular membership functions, that we use in the examples shown in this paper. In this case, the ordered set $S = \{VL, L, M, VM, AH, H, VH, AP, P\}$ permits to distinguish with precision positive examples (*e.g.*, *almost perfect* - AP- is slightly better than *very high* -VH-).

[1] Universitat Rovira i Virgili, Departament d'Enginyeria Informàtica i Matemàtiques, ITAKA Research Group, Catalonia (Spain). Emails: {lucas.marin, david.isern, antonio.moreno}@urv.cat

Each criteria is composed by several values that are handled independently. An example of user profile can be the interests of a patient about some healthcare centres around his town. In this case, the recommender system is used to prioritize sets of proposals of medical visits with specialists that a user (patient) can receive when his practitioner requires a medical test. In this case of study, each alternative contains the day of the week where the appointment should be made, the period of the day (morning, afternoon, night), the existence or not of a parking area (blue zone -payment area-, no parking, free parking), the distance from the patient's home to the medical centre (very near, near, far, very far), and finally, we have a global evaluation of the medical centre's doctors (excellent, notable, good, normal, bad, very bad). These preferences are shown in the screenshot of the recommender platform depicted in Fig. 2.

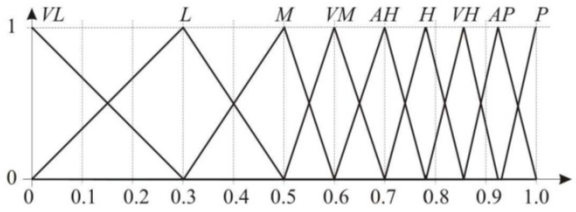


Figure 1. Fuzzy term set S with 9 unbalanced labels

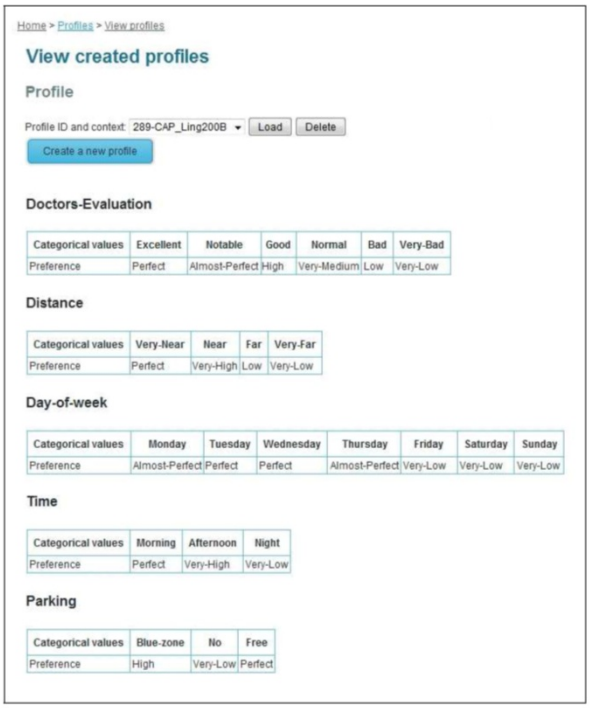


Figure 2. Example of user profile

3 UNSUPERVISED ADAPTATION ALGORITHM

As said before, in order to maintain an updated and accurate profile of a user, it is necessary to extract important information of the feedback generated from his interaction with the recommender system. Our unsupervised adaptation approach extracts this information from the set of alternatives that were ranked and shown to the user when using the ranking module of the recommender system. Two units of information are extracted from here: the selected alternative, which is the final decision of the user, and the set of alternatives that were ranked above the selected one. The main idea is that the adaptation process must be able to internally find an answer to questions such as "why are those alternatives better ranked that the one the user really wants?" or "why the alternative the user really wants does not have a score good enough to be ranked in the first place?".

The 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 just evaluates the information that can be extracted of the current ranked alternatives. The second one, called *off-line* adaptation, is triggered after some user recommendations have been served, and uses the information of the previous ranked alternatives and selections.

Adaptation processes modify the user profile by increasing or decreasing the level of preference of the attribute values according to the reasoning explained below. The algorithm accumulates evidences to update all values, decides which values should be updated, and finally changes them by adding/subtracting one label of S. For instance, a value with the label *high* can change to *very-high* or *almost-high*.

3.1 On-line adaptation process

The on-line profile adaptation process tries to maintain the user profile updated by evaluating each recommendation and the final decision the user makes individually. For each recommendation the system makes, two sources of information are evaluated: the selected alternative, which is the final decision of the user, and the alternatives that were ranked above the selected one. Many conclusions can be extracted by evaluating this information. Those conclusions are found by extracting *characteristics* from the available data. A characteristic is a tuple composed by a linguistic attribute, a concrete label from the set of the possible values of said linguistic attribute, and the times it is repeated among a concrete set of alternatives. It is necessary to have a minimum number of alternatives to allow the extraction of characteristics since characteristics extracted from a reduced number of alternatives do not give useful information.

The main objective of this stage is to decrease the preference of the attribute values that are causing non desired alternatives to be ranked with higher scores, and also to increase the preference of the attribute values which are important for the user but are not well judged on the current user profile.

We call *over ranked characteristics* the features extracted from the set of alternatives that were ranked above the user final selection. In order to obtain relevant characteristics, a *repetition threshold* is defined: just the characteristics whose repetition value is equal or over that threshold will be considered. In our tests, we used a threshold defined as a percentage on all the over ranked alternatives. Moreover, there is another threshold that determines that if there are not enough over ranked alternatives the system can not extract any characteristic. Finally, an over ranked characteristic is only taken into account if it does not appear on the selected alternative.

Consider the set of alternatives shown in Table 1, in which the sixth alternative is the one selected by the user, and a repetition threshold of 40%. The tuple ('Day of Week', 'Sat', '2') is an example of over ranked characteristic, being 'Day of Week' the attribute name, 'Saturday' a possible attribute value, and '2' the times it ap-

pears among the set of over ranked alternatives. Note that ('Day of Week', 'Thu', '1') can not be considered as an over ranked characteristic since it only appears once, thus it just represents a 20% of the over ranked alternatives. ('Parking', 'Free', '2') is not an over ranked characteristic either, since this value appears on the selected alternative.

Table 1. Example set of alternatives.

Rank	*Distance*	*Doctors Evaluation*	*Day of Week*	*Time*	*Parking*
1	Far	Excellent	Sat	Afternoon	Blue Zone
2	Far	Bad	Sat	Night	Blue Zone
3	Very-Far	Good	Fri	Afternoon	No
4	Very-Far	Good	Fri	Afternoon	Free
5	Near	Very-Bad	Thu	Morning	Free
6	**Very-Near**	**Excellent**	**Mon**	**Morning**	**Free**
7	Far	Notable	Wed	Afternoon	Free

Over ranked characteristics are used to *decrease* the level of preference of those attribute values pointed out by the characteristics. The intensity of that decrease is regulated by the number of repetitions: if a characteristic is repeated many times, we have more evidence to decrease the preference on that attribute value.

We call *selection characteristics* the features extracted from the user final selection that do not appear on the set of over ranked alternatives more than a given number of times. As in the over ranked characteristics, a repetition threshold is also defined here, but its use is opposite as when extracting over ranked characteristics. The adaptation process will only consider as selection characteristics the ones which appear on the over ranked alternatives equal or fewer times of said threshold. Following the values of Table 1, the data structures ('Day of Week', 'Mon', '0') and ('Time', 'Morning', '1') are two examples of correct selection alternatives.

Following a similar procedure, selection characteristics are also used to *increase* the level of preference of the attribute values pointed out by the characteristics. The intensity of that increase is greater the fewer times that characteristic appears among the over ranked characteristics. In other words, using the previous example, we have more evidence to increase the preference of the value 'Monday' of the attribute 'Day of week' than the one of the value 'Morning' of the attribute 'Time', since the first does not appear among the over ranked alternatives and the second appears once.

3.2 Off-line adaptation process

The on-line adaptation process can give an immediate response to a single user interaction with the recommender system. However, there are cases in which the information extracted from a single interaction is not enough to gather enough evidence to make changes in the profile. In those cases, data from past recommendations is stored and accumulated to be analysed later. The off-line adaptation process manages over ranked characteristics and selections differently as is explained below.

We defined the minimum number of over ranked alternatives as a requirement to start extracting characteristics. However, this causes the following problem: when the profile is near to the ideal profile, the user selection is among the first ranked alternatives, so there are not enough over ranked alternatives to analyse. This leads to the system being unable to determine which preferences it has to increase or decrease to reach the ideal profile. A solution to this problem can be to store those small pieces of data and evaluate them as a whole when there is enough information. Since this information comes from different recommendations, it is not appropriate to compare over ranked alternatives with the selected alternatives as in the on-line process. However, evidences for preference decrease can be extracted by evaluating the over ranked alternatives separately, and evidences for preference increase can be extracted by evaluating the selected alternatives separately.

When the system faces cases when the number of over ranked alternatives is not enough to extract reliable characteristics, it stores that small number of over ranked alternatives into the database. After some cases of insufficient number of over ranked alternatives to evaluate, the system gathers enough alternatives to start evaluating them. When the number of saved over ranked alternatives is adequate, characteristics are extracted from that set of accumulated alternatives. Those characteristics are used to decrease the preferences in the same way as it is done in the on-line process: characteristics with a repetition value over the defined threshold are decreased. After the stored over ranked alternatives have been treated, they are erased from the database. The number of stored over ranked alternatives to reach before evaluating them is the same than the one used in the on-line process to determine if there are enough over ranked alternatives to extract characteristics.

On the other hand, user selections can be stored and, after a certain number of choices have been made, they can be evaluated. By extracting characteristics from that set of stored selections, the most repeated attribute values, which are the ones of most interest to the user, can have its preference increased. After evaluating them, they are erased from the database. The number of selections stored before evaluating them is another parameter of the algorithm.

3.3 Adaptation mechanism

After a single recommendation and the posterior selection of the final decision of the user, many characteristics can be found and the system can deduce a large amount of changes to do in the profile. Many of those changes, also named 'adaptations', although deduced using the reasoning techniques defined before, can be incorrect.

We have addressed this problem by restricting the number of adaptations that can be made to the profile for each execution of the adaptation process. A parameter limits the number of increases and decreases which can be made to a profile per adaptation step (a value of '2' allows a total of 4 changes: 2 increases and 2 decreases).

When a large number of adaptations are being considered, only the more evident ones are performed. For the decreases, that 'level of evidence' is measured by the number of the repetitions of that characteristic: the ones with more repetitions will be decreased. For the increases, it is measured by counting the number of over ranked alternatives which do not have those characteristics, so the ones with a greater counter will be increased. This mechanism reduces the errors that can be done by increasing/decreasing wrong preferences, although it may be inefficient when those levels of evidence are equal among many possible adaptations, including the wrong ones.

Our approach to solve this problem involves a *signed counter* which regulates the final increase/decrease of the preference of the profile. When the adaptation process detects an evidence for a possible adaptation, it increases/reduces a counter, initially set to 0, associated to that attribute value (for example the 'Morning' value of the attribute 'Time'). This amount increased or decreased is the level of evidence explained before. When this counter reaches a certain value defined by a parameter, the increase/reduction is finally performed over the preference in the profile and the counter is reset to 0. Con-

sidering said parameter as '+/-5' and following the example of Table 1, the system finds out that the 'Morning' preference value of the attribute 'Time' needs to be increased so it increases its counter by '4', since it only appears once among the over ranked alternatives. So, the counter does not reach the necessary number to increase the preference value on the profile and it will be necessary another positive evidence to finally change it.

This mechanism was motivated by the observation of the fact that the great majority of erroneous adaptations have low levels of evidence, so with this mechanism several consecutive erroneous evidences need to be treated before performing a wrong change in the profile.

4 EVALUATION

4.1 Web-based platform

In order to test and evaluate our recommender method, we implemented a Web platform[2] that permits the definition of a problem (set of user criteria, aggregation and adaptation parameters), the definition of an initial profile, the definition of an ideal profile that we aim to learn, the random generation of a corpus of alternatives for the problem, the simulation of tests of user interaction with the system, and finally, the visualisation of the evolution of the user profile.

The accesses to the platform are regulated by a user name and a password, allowing the maintenance of multiple users with different profiles for each recommendation domain. This approach permits to analyse the variability of results under different circumstances in the aggregation and adaptation stages.

Users can upload data files with information about the alternatives that must be analysed in order to make a decision. The platform stores that information in a database, letting users to define a decision making problem on those alternatives.

As shown in the pseudocode of Fig. 3, the algorithm permits an automatic validation by simulating how the system adapts an initial random profile using the information extracted from choices calculated using an ideal profile. The platform also permits a real user to perform the selection. The algorithm requires several parameters such as the corpus of alternatives initialised *a priori*, some thresholds analysed in detail in the next subsections, and the profiles to run a simulation. The first important task is to rate and rank the corresponding block of alternatives in each iteration. This sorted set using the current profile (P) is used later in the adaptation process. In this automatic simulation mode, the alternative selected u is the best rated alternative of R over the ideal profile I. The current selection ($R(a'_u)$) and the set of over ranked alternatives ($R(a'_0, \ldots, a'_{u-1})$) are used to find the evidences in the on-line adaptation step. Evidences count if attributes should be increased or decreased. The off-line stage complements the on-line one in order to compose the set of evidences used to adapt the current profile.

4.2 General performance

In order to evaluate the framework, we propose to iteratively measure the distance between the current user profile and the ideal profile that we want to learn. Iteratively and automatically, the system selects the alternative that best fits the ideal profile, which is the one that the user which we want to have its profile learnt would choose. Then, the selection and the over ranked alternatives are used by the adaptation processes to evaluate which changes should be performed. To

[2] Website: http://itaka2-deim.urv.cat/recommender/

ADAPTATION-ALGORITHM(
$E(a_0, \ldots, a_{max})$, *//corpus of alternatives*
k, *//level of evidence*
t, *//characteristics extraction threshold*
pc, *//maximum preferences changes threshold*
ai, *//alternatives per iteration shown*
$MaxIter$, *//iterations to simulate*
I, *//ideal profile*
P *//initial profile*)
begin
$iter, beg, end = 0$;
$ovr, sel = \emptyset$;
while $(iter < MaxIter)$ do
$beg = iter * ai$;
$end = (iter + 1) * ai - 1$;
$R(a'_0, \ldots, a'_{ai-1}) =$
rating-and-ranking$(P, E(a_{beg}, \ldots, a_{end}))$;
$u =$ *calculate-ideal-selection*(R, I);
$sel = sel \cup R(a'_u)$;
if $(u > 0)$ then $ovr = ovr \cup R(a'_0, \ldots, a'_{u-1})$;
$evidences =$ *on-line*$(P, R(a'_u), R(a'_0, \ldots, a'_{u-1}), t)$;
$evidences = evidences +$ *off-line*(P, sel, ovr, t);
$P =$ *adaptation*$(P, evidences, pc, k)$;
$iter = iter + 1$;
end while;
end;

Figure 3. Pseudocode of the simulator

compare the results of different tests, we initialize the user profile randomly, but we maintain the ideal user profile. Distances between both profiles decrease over time.

All tests have been made using the same corpus of 3000 alternatives initialised randomly, with blocks of 15 alternatives per iteration. Profiles (ideal and initial) have been initialised manually (using the attributes explained in section 2) in order to obtain comparable results, and using the term set shown in Fig. 1. Although we will analyse the influence of the main parameters later, these tests have been performed using the extraction characteristics threshold of 30%, with the level of evidence +/-5 to consider a change in a value, and allowing only 2 changes in each sense in one iteration. Under these circumstances, we analyse the distance between the current (P) and the ideal (I) profiles, which is calculated as follows:

$$dist(I, P) = \tag{1}$$
$$= \frac{1}{n} \sum_{j=1}^{n} \left(\frac{1}{m_j} \sum_{k=1}^{m_j} \frac{|COG_x(v(P, j, k)) - COG_x(v(I, j, k))|}{|COG_x(s_0) - COG_x(s_r)|} \right)$$

Here, n is the number of attributes, m_j is the cardinality of the attribute j, COG_x evaluates the coordinate x of the centre of gravity of a linguistic label [2], v is a function that returns the linguistic value of a profile, an attribute and a given value. With this function we obtain an average of the distances between all pairs of labels. The distance is normalised by using the extreme values of the fuzzy set $S = \{s_0, \ldots, s_r\}$.

The distance is 0 when both profiles are identical. The maximum distance is 1 when all values contained in the user and the ideal profiles have just the opposite labels (s_0 and s_r). Moreover, this function is commutative ($dist(P, I) = dist(I, P)$).

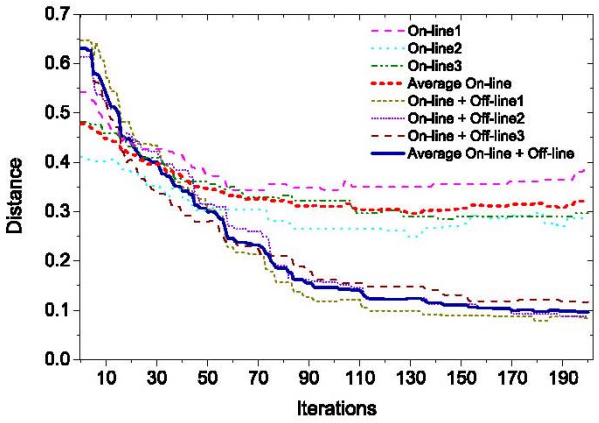


Figure 4. Distances between user and ideal profile

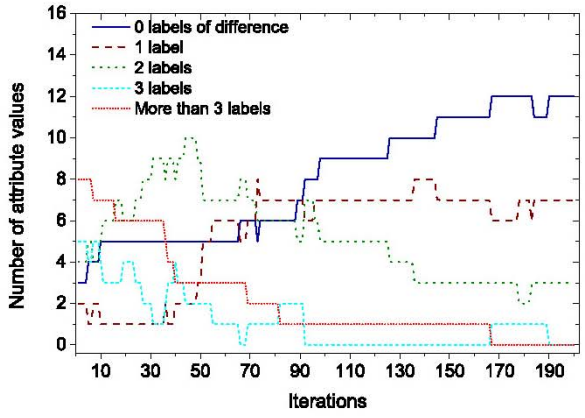


Figure 5. Quantitate study of the labels misclassified in the comparison of the current profile with the ideal profile

Fig. 4 compares the general performance of the adaptation algorithm using the on-line process, and using the on-line combined with the off-line stage. The figure shows the distances between the current and the ideal profile of three examples of each case and their average through 200 iterations. The three examples have different initial profiles but share the same ideal profile. The general performance accomplished when both processes are working together to adapt the profile is much better than when just using the on-line process. The parameters used in this simulation are the ones which gave the best results in the parameter's analysis conducted in the following section. Eq. (1) gives a general evaluation of the performance of the adaptation algorithm. In addition, Fig. 5 focuses the study on one simulation, and shows the labels of distance between the values of current and ideal profiles. For instance, if an attribute of the current profile is H and the ideal is P, the distance is 3. The figure compares how many attributes there are with difference 0 (correctly classified), 1, 2, 3 and more than 3 labels. It can be seen how the number of the most misclassified labels (distance 3 and more) decreases drastically, while the set of well classified attributes (distances 0 and 1) increases.

4.3 Analysis of the parameters

The proposed implicit adaptation algorithm introduces several parameters that should be properly customised. This section explains the influence of these parameters in the final result, although exact parameter values may vary depending on the context in which the adaptation is taking place (number of attributes, number of values per attribute, etc.).

Over ranked characteristics are extracted by considering a threshold defined as a percentage over the number of over ranked alternatives. Fig. 6 shows that after several tests (using the threshold values of 15%, 30%, 50%, and 80%), a threshold of 30% was the one that got better results.

Having a greater threshold implies that the extracted characteristics need to have a greater repetition value, thus fewer characteristics are extracted, reducing the number of possible changes in the profile. On the contrary, the extraction of valuable information is compromised as the threshold decreases, because incorrect characteristics (which usually have a low number of repetitions) are extracted more easily.

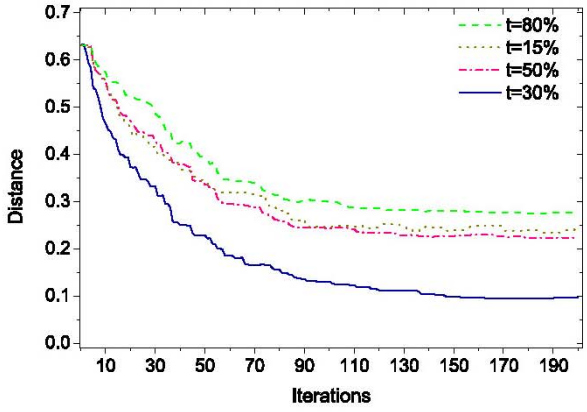


Figure 6. Influence of the characteristics extraction threshold during tests

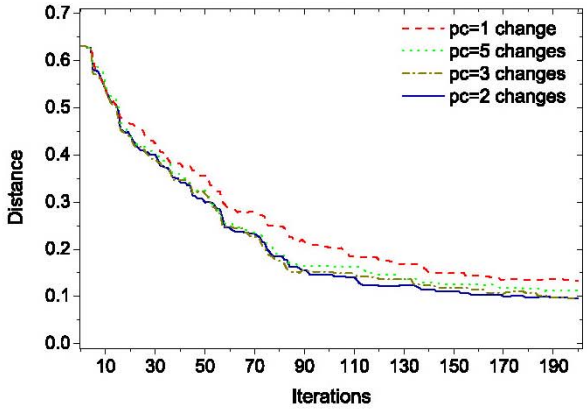


Figure 7. Influence of the maximum number of preference changes at each iteration

On the adaptation mechanism section, a parameter for controlling how many increases/decreases of preference can be done at each adaptation process iteration was introduced. Fig. 7 compares the performance of the system with different values. Setting that value to '2' proved to give better results than using a lower or greater number, letting the system do up to two increases and two decreases on any attribute value preference of the profile.

Moreover, section 3.3 also introduces the parameter that varies the necessary number of evidences to increase/decrease a preference on the profile. In this case, with lower values, the number of required evidences is lower, and the quality of these changes is also compromised. On the other hand, higher values for this parameter difficult to make changes on preferences, so it requires more iterations (selections) to reach a near-ideal profile. Those differences, however, tend to decrease as the number of iterations increases. Fig. 8 shows these behaviours and how an intermediate value such as +/-5 exhibits a good performance.

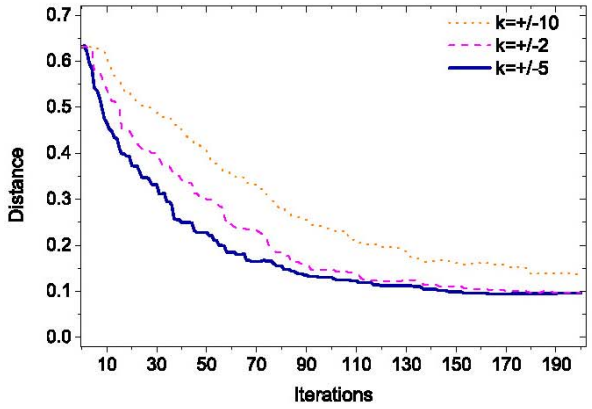


Figure 8. Influence of the level of evidence during tests

5 CONCLUSIONS

The Web-based recommender system proposed in this paper has been designed as a general framework that permits to solve decision problems using MCDA techniques. The separation between the domain-dependant (*e.g.*, adaptation thresholds) and the domain-independent data (*e.g.*, rating, ranking and adaptation processes) permits to understand the evolution of values, as well as to reuse the same methods in a wide range of problems.

The system has been prepared to make automatic simulations, which permit to study the influence of several parameters of the proposed adaptation algorithm. Further studies should be performed to analyse the influence of the fuzzy term set S, and to determine the minimum number of over ranked alternatives to be taken into account in the off-line adaptation stage.

The creation of this platform is the first step in a project that aims to develop techniques to adapt the user profile according to the selections made in the last step of the recommendation process. We consider that having automatic algorithms to update the preferences is required in many domain applications, where the preferences may change dynamically over time. The paper is focused in the implicit adaptation of the user profile using linguistic variables, but we made preliminary attempts to deal also with numerical attributes. The final goal is to combine these two kinds of data into a single algorithm.

As stated before, this adaptation is focused on working in environments in which the user requires recommendations quite often, such as recommendation of films or restaurants, or even recommendations for tourists. Probably, this framework is not so adequate for punctual decisions such as 'buying a car'. However, under real settings like buying a car, the algorithm can be optimised with an initial profile learnt *a priori*. This set up stage can be performed through the creation of stereotyped profiles, such as 'teenager', 'family man', 'traveller', which permit to start the adaptation algorithm near the ideal profile. Another way to create the initial profile is requesting the user to fill in a form.

ACKNOWLEDGEMENTS

This work has been supported by the Universitat Rovira i Virgili (predoctoral grant of L. Marin, post-doctoral grant of D. Isern, and the project 2009AIRE-04), the Spanish Government (PlanE, Spanish Economy and Employment Stimulation Plan) and the Spanish Ministry of Science and Innovation (DAMASK project, TIN2009-11005).

REFERENCES

[1] L. Baltrunas and X. Amatriain, 'Towards Time-Dependant Recommendation based on Implicit Feedback', in *Proceedings of the Workshop Context-aware Recommender Systems in conjunction with the 3rd ACM Conference on Recommender Systems, RecSys 09*, eds., G. Adomavicius and F. Ricci, pp. 25–30, New York, USA, (2009).

[2] S. J. Chen and S. M. Chen, 'Fuzzy risk analysis based on similarity measures for generalized fuzzy numbers', *IEEE Transactions on Fuzzy Systems*, **11**(1), 45–56, (2003).

[3] E. Cheng, F. Jing, M. Li, W. Ma, and H. Jin, 'Using Implicit Relevance Feedback to Advance Web Image Search', in *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME 2006*, pp. 1773–1776, Toronto, ON, Canada, (2006). IEEE Computer Society.

[4] D. Isern, L. Marin, A. Valls, and A. Moreno, 'The Unbalanced Linguistic Ordered Weighted Averaging Operator', in *Proceedings of the 2010 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2010*, p. (to appear), Barcelona, Catalonia, (2010). IEEE Computer Society.

[5] M. Montaner, B. López, and J. L. de la Rosa, 'A Taxonomy of Recommender Agents on the Internet', *Artificial Intelligence Review*, **19**(3), 285–330, (2003).

[6] D. M. Nichols, 'Implicit Rating and Filtering', in *Proceedings of the 5th DELOS Workshop of Filtering and Collaborative Filtering*, ed., L. Kovács, pp. 31–36, Budapest, Hongary, (1997). European Research Consortium for Informatics and Mathematics.

[7] O. Noppens, M. Luther, T. Liebig, M. Wagner, and M. Paolucci, 'Ontology-supported Preference Handling for Mobile Music Selection', in *Proceedings of the Multidisciplinary Workshop on Advances in Preference Handling in conjunction with ECAI 2006*, eds., U. Junker and W. Kießling, pp. 96–101, Riva del Garda, Italy, (2006).

[8] S.-F. Pan and J.-H. Lee, 'eDAADe: An Adaptive Recommendation System for Comparison and Analysis of Architectural Precedents', in *Proceedings of the 4th International Conference of Adaptive Hypermedia and Adaptive Web-Based Systems, AH 2006*, eds., V. Wade, H. Ashman, and B. Smyth, volume 4018 of *Lecture Notes in Computer Science*, pp. 370–373, Dublin, Ireland, (2006). Springer Berlin / Heidelberg.

[9] M. Pazzani and D. Billsus, 'Learning and Revising User Profiles: The Identification of Interesting Web Sites', *Machine Learning*, **27**(3), 313–331, (1997).

[10] B. Roy, *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *International Series in Operations Research and Management Science. Lecture Notes in Economics and Mathematical Systems*, chapter Paradigms and Challenges, 3–24, Springer New York, 2005.

[11] V. Torra and Y. Narukawa, *Modeling Decisions: Information Fusion and Aggregation Operators*, Springer Berlin Heidelberg, 2007.

A Generic User Profile Adaptation Framework

Lucas MARIN [a] and David ISERN [a,1], and Antonio MORENO [a]
[a] *Universitat Rovira i Virgili, Departament d'Enginyeria Informàtica i Matemàtiques, ITAKA Research Group, Tarragona (Catalunya)*

Abstract. The paper presents a recommender system that permits to manage user preferences using linguistic criteria and, after collecting information about selections made by the user, it performs an unsupervised adaptation of the user profile. It has been implemented as a Web application and designed in a generic way so that it can be applied to any decision making problem. It includes two separate modules: a module to rate and rank all alternatives received by the system according to the current interests of the user, and a module to adapt the current user profile in an unsupervised fashion collecting implicit information about the user interaction with the system. The paper presents some preliminary results and discusses the performance of the adaptation algorithm.

Keywords. Preferences, Implicit adaptation of profile, Recommending system

1. Introduction

Nowadays, users are constantly confronted with situations in which they have to select one alternative from a set of possibilities. Those daily scenarios, such as deciding which radio station to listen, or which program to watch in TV, require the inclusion of user preferences to sort all the alternatives according to the user interests. Then, the next step is to interpret the actual choice made by the user in order to dynamically adapt the user profile over time. To achieve these goals, the system needs to create and maintain the user profile including relevant information about some criteria. When users interact with a computer, they provide a great deal of information about themselves. Successful interpretation of these data streams is required to tailor systems to each individual's behaviour, habits and knowledge [5]. In addition, users interests about an issue may evolve due to changes in their personal conditions (*e.g.*, change of location, having a child).

This paper presents a recommender system designed in two separate modules (see Fig. 1): rating and ranking a set of alternatives, and the adaptation of the user profile. The recommender system receives a set of alternatives composed by several criteria. The user profile stores the preferences of the user over all those criteria. The first step of the recommender is to rate all the alternatives taking into account the user profile in order to prioritize them according to his particular circumstances. Then, the user selects the most

[1]Corresponding Author: Universitat Rovira i Virgili, ITAKA Research Group. Av. Països Catalans, 26 (Campus Sescelades), 43007 Tarragona (Catalunya); E-mail: david.isern@urv.cat

appropriate alternative from this sorted list. Afterwards, the adapting algorithm collects the information provided by this selection, the set of non selected alternatives, and all past selections, to infer which changes can be made to the current profile. As it will be described later, we have implemented a Web-based platform that permits to simulate all those stages and extract some conclusions.

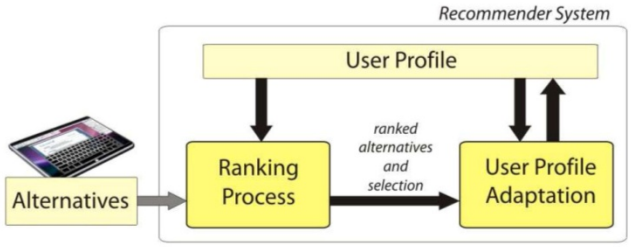


Figure 1. Architecture of the recommender system

Learning and adapting profiles are widely studied research areas. The user profile may be learnt by extracting patterns from training data, *e.g.*, using clustering techniques. Concerning the adaptation of the profile, some information about how the user is interacting with the system is needed. The two most common ways to obtain this information, known as relevance feedback [10], are to use information given *explicitly* by the user and to get the information *implicitly* by observing the user behaviour. Some *hybrid* systems combine implicit and explicit approaches [6].

On the one hand, explicit feedback is obtained when users are required to explicitly evaluate items, indicating how relevant or interesting an item is to the user in a numeric or linguistic scale. These systems, such as [7] and [8], offer high performance and simplicity. On the other hand, implicit feedback is obtained by monitoring the user actions and automatically inferring the user preferences. The amount of data collected is very large, the computation needed to derive recommendations for adaptations is extensive, and the confidence in the suitability of these adaptations is likely to be relatively low. This approach has been less explored, although some existing works (*e.g,* [1], [3]) show promising results [5].

Explicit feedback is sometimes not desirable because it has some serious limitations. One of those is that numeric scales may not be adequate for describing the reactions humans have to items. Moreover, as studied in [9], users are reluctant to spend time giving explicit feedback and only 15% of the users would supply it even if they were encouraged to do so. For these reasons, we present an algorithm that adapts the personal profile of the users in an unsupervised way. We assume that the system is used regularly, so that it can learn, from the usage history, how the preferences of the decision maker evolve. The main goal is to obtain adequate recommendations by adapting the preference functions over time.

The rest of the paper is organised as follows. Section 2 briefly explains the information stored in a user profile, which is used to rate and rank the alternatives (Section 3). Section 4 explains the algorithm of adaptation of the user profile. Then, section 5 presents the Web-based platform implemented to test all recommender settings and discusses the performance of the system, paying special attention on how the user profile evolves. Finally, section 6 gives some conclusions and comments the research lines we are working on.

2. Representation of the user preferences

In our case, the user profile stores the user preferences about some criteria using linguistic variables (S). The selection of the set of labels S and their membership functions is an important decision that influences the final results. We tested different scenarios with both balanced and unbalanced sets of terms (see [4]).

Fig. 2 shows a fuzzy term set S with nine unbalanced labels defined by triangular membership functions, which we use in the examples shown in this paper. In this case, the ordered set $S = \{VL, L, M, VM, AH, H, VH, AP, P\}$ (translated as *very low*, *low*, *medium*, *very medium*, *almost high*, *high*, *very high*, *almost perfect* and *perfect*) permits to distinguish with precision positive examples (*e.g.*, AP is slightly better than VH).

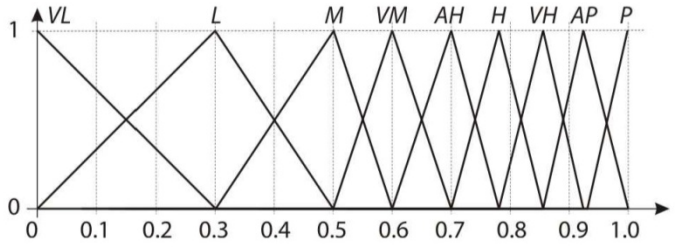


Figure 2. Fuzzy term set S with 9 unbalanced labels

An example of user profile can be the interests of a patient about some healthcare centres. The recommender system is used to prioritize sets of proposals of medical visits with specialists that a user (patient) can receive when his practitioner requires a medical test. In this case of study, each alternative contains the day of the week where the appointment should be made, the period of the day (morning, afternoon, night), the existence or not of a parking area (blue zone -payment area-, no parking, free parking), the distance from the patient's home to the medical centre (very near, near, far, very far), and a global evaluation of the medical centre's doctors (excellent, notable, good, normal, bad, very bad). Table 1 shows an example of user profile.

Table 1. Example of user profile including linguistic variables

Criteria	$\langle$*attribute,preference*$\rangle$*
Distance	$\langle$Very-far,$VL\rangle$, $\langle$Far,$L\rangle$, $\langle$Near,$VH\rangle$, $\langle$Very-near,$P\rangle$
Doctors Evaluation	$\langle$Excellent,$P\rangle$, $\langle$Notable,$AP\rangle$, $\langle$Good,$VH\rangle$, $\langle$Normal,$H\rangle$, $\langle$Very-bad,$VL\rangle$
Day of Week	$\langle$Mon,$AP\rangle$, $\langle$Tue,$P\rangle$, $\langle$Wed,$P\rangle$, $\langle$Thu,$AP\rangle$, $\langle$Fri,$P\rangle$, $\langle$Sat,$VL\rangle$, $\langle$Sun,$VL\rangle$
Time	$\langle$Morning,$P\rangle$, $\langle$Afternoon,$VH\rangle$, $\langle$Night,$VL\rangle$
Parking	$\langle$Blue-zone,$H\rangle$, $\langle$No,$VL\rangle$, $\langle$Free,$P\rangle$

* Preferences use the term set S shown in Fig. 2.

3. Aggregation-based preference rating

As Fig. 1 shows, the recommender system receives a collection of alternatives. First of all, an utility function translates the attribute values into linguistic labels, taking into account the user interests. This function, defined implicitly by the user profile, consists on a mapping from the set of possible values of a categorical variable into the set of preference terms S.

A multi-criteria decision making process using an aggregation policy evaluates all the criteria and finally obtains a single value used to rank the alternatives. We apply an aggregation operator called ULOWA (Unbalanced Linguistic Ordered Weighted Average), which exploits membership functions to calculate the final aggregated value (for more details about this operator and its properties, see [4]).

The ULOWA operator ϕ aggregates a set of labels $A = \{a_1, \ldots, a_m\}$, where $a_i \in S$, with respect to a set of weights $W = \{w_1, \ldots, w_m\}$ such that $w_i \in [0, 1]$ and $\sum_i w_i = 1$. Those weights specify the decision-maker policy [11].

$$\phi(a_1, \ldots, a_m) = W \cdot B^T = C^m\{w_k, b_k, k = 1, \ldots, m\} =$$
$$= w_1 \otimes b_1 \oplus (1 - w_1) \otimes C^{m-1}\{\beta_h, b_h, h = 2, \ldots, m\}$$

C^m is the convex combination operator of m labels. If $\exists j \in 1, \ldots, m,\ w_j = 1$ (so $\forall k \in 1, \ldots, m,\ k \neq j\ w_j = 0$), then $C^m\{w_i, b_i, i = 1, \ldots, m\} = b_j$.

Here, β_h is the corresponding weight vector $\beta_h = w_h / \sum_2^m w_h, h = 2, ..., m$, and $B = (b_1, \ldots, b_m)$ is a permutation of the elements of A, such that $B = \sigma(A) = \{a_{\sigma(1)}, ..., a_{\sigma(m)}\}$, where $a_{\sigma(j)} < a_{\sigma(i)}\ \forall i \leq j$.

The convex combination of two terms (C^2) taking $b_1 = s_j$ and $b_2 = s_i$, with $s_j, s_i \in S\ (j \geq i)$ is calculated taking into account the membership functions of the labels s_j and s_i, using the following expression:

$$C^2\{w_i, b_i, i = 1, 2\} = w_1 \otimes s_j \oplus (1 - w_1) \otimes s_i = s_k \quad (1)$$
$$\text{such that } k = \underset{i \leq p \leq j}{\text{argmax}} \{Sim(s_p, \delta)\}$$

where, δ is an intermediate crisp number between s_i and s_j defined as:

$$\delta = (x_k, x_k, x_k, x_k) \text{ with } x_k = COG_x(s_i) + w_1(COG_x(s_j) - COG_x(s_i)) \quad (2)$$

Here, COG_x evaluates the coordinate x of the centre of gravity of a linguistic label [2]. The similarity function (Sim) for two fuzzy sets is described as:

$$Sim(A, B) = \sqrt[4]{\prod_{i=1}^{4} (2 - |a_i - b_i|)} - 1$$

The behaviour of the aggregation operator ϕ depends on the weighting vector W. The reader may find more details on the ULOWA operator, its properties, examples of use and a comparison with related similarity functions in [4].

4. Unsupervised adaptation algorithm

Our unsupervised adaptation approach extracts the feedback information from the set of alternatives that were ranked and shown to the user. Two units of information are extracted: the selected alternative, which is the final decision of the user, and the set of alternatives that were ranked above the selected one. The main idea is that the adaptation process must be able to find an answer to questions such as "why are those alternatives better ranked that the one the user really wants?" or "why the alternative the user really wants does not have a score good enough to be ranked in the first place?".

The 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 just evaluates the information that can be extracted from the current set of ranked alternatives. The second one, called *off-line* adaptation, is triggered after some user recommendations have been served, and uses the information of the ranked alternatives and selections in that set of recommendations.

The adaptation processes modify the user profile by increasing or decreasing the level of preference of the attribute values according to the reasoning explained below. The algorithm accumulates evidences to update all values, decides which values should be updated, and finally changes them by adding/subtracting one label of S. For instance, a value with the label *high* can change to *very high* or *almost high*.

4.1. On-line adaptation process

The on-line profile adaptation process tries to maintain the user profile updated by evaluating individually each recommendation and each decision the user makes . For each recommendation the system makes, two sources of information are evaluated: the selected alternative, which is the final decision of the user, and the alternatives that were ranked above the selected one. With these two elements, we extract *characteristics*. A characteristic is a tuple composed by a linguistic attribute, a concrete label from the set of the possible values of said linguistic attribute, and the times it is repeated among a concrete set of alternatives. It is necessary to have a minimum number of alternatives to allow the extraction of characteristics since characteristics extracted from a reduced number of alternatives do not give useful information. The main goal of this stage is to decrease the preference of the attribute values that are causing non-desired alternatives to be ranked with higher scores, and also to increase the preference of the attribute values which are important for the user but are not currently well rated.

We call *over ranked characteristics* the features extracted from the set of alternatives that were ranked above the user final selection. In order to obtain relevant characteristics, a *repetition threshold* is defined: just the characteristics whose repetition value is equal or over that threshold will be considered. In our tests, we used a threshold defined as a percentage on all the over ranked alternatives. An over ranked characteristic is only taken into account if it does not appear on the selected alternative.

Consider the set of alternatives shown in Table 2. This table shows seven alternatives sorted according to the rating obtained with the user profile (the rate is depicted in parenthesis in the first column). Imagine that the user selects the sixth alternative. Taking into account a repetition threshold of 40%, the tuple $\langle$*Day of Week, Sat, 2*$\rangle$ is an example of over ranked characteristic, being *Day of Week* the attribute name, *Sat* a possible

attribute value, and *2* the times it appears among the set of five over ranked alternatives. Note that ⟨*Day of Week, Thu, 1*⟩ can not be considered as an over ranked characteristic since it only appears once, thus it just represents a 20% of the over ranked alternatives. ⟨*Parking, Free, 2*⟩ is not an over ranked characteristic either, since this value appears on the selected alternative.

Table 2. Example set of rated and sorted alternatives

Rank	*Distance*	*Doctors Evaluation*	*Day of Week*	*Time*	*Parking*	*Selection*
1 (*VH*)	Far	Excellent	Sat	Afternoon	Blue Zone	○
2 (*H*)	Far	Bad	Sat	Night	Blue Zone	○
3 (*H*)	Very-Far	Good	Fri	Afternoon	No	○
4 (*AH*)	Very-Far	Good	Fri	Afternoon	Free	○
5 (*AH*)	Near	Very-Bad	Thu	Morning	Free	○
6 (*M*)	Very-Near	Excellent	Mon	Morning	Free	●
7 (*L*)	Far	Notable	Wed	Afternoon	Free	○

Over ranked characteristics are used to *decrease* the level of preference of those attribute values pointed out by the characteristics. The intensity of that decrease is regulated by the number of repetitions: if a characteristic is repeated many times, we have more evidence to decrease the preference on that attribute value.

We call *selection characteristics* the features extracted from the user final selection that do not appear on the set of over ranked alternatives more than a given number of times. As in the over ranked characteristics, a repetition threshold is also defined here, but its use is opposite as when extracting over ranked characteristics. The adaptation process will only consider as selection characteristics the ones which appear on the over ranked alternatives equal or fewer times of a given threshold. Following the values of Table 2, the data structures ⟨*Day of Week, Mon, 0*⟩ and ⟨*Time, Morning, 1*⟩ are two examples of correct selection characteristics.

Following a similar procedure, selection characteristics are also used to *increase* the level of preference of the attribute values pointed out by the characteristics. The intensity of that increase is greater the fewer times that characteristic appears among the over ranked characteristics. In other words, using the previous example, we have more evidence to increase the preference of the value *Mon* of the attribute *Day of week* than the one of the value *Morning* of the attribute *Time*, since the first does not appear among the over ranked alternatives and the second appears once.

4.2. Off-line adaptation process

A minimum number of over ranked alternatives is required to start extracting characteristics. However, this causes the following problem: when the profile is near to the ideal profile, the user selection is among the first ranked alternatives, so there are not enough over ranked alternatives to analyse. This leads to the system being unable to determine which preferences it has to increase or decrease to reach the ideal profile. A solution to this problem can be to store those small pieces of data and evaluate them as a whole when there is enough information. After some cases of insufficient number of over ranked alternatives to evaluate, the system gathers enough alternatives to start evaluating them. Since this information comes from different recommendations, it is not appropriate to compare

over ranked alternatives with the selected alternatives as in the on-line process. However, evidences for preference decrease can be extracted by evaluating the over ranked alternatives separately, and evidences for preference increase can be extracted by evaluating the selected alternatives separately.

When the number of saved over ranked alternatives is adequate, characteristics are extracted from that set of accumulated alternatives. Those characteristics are used to decrease the preferences in the same way as it is done in the on-line process: characteristics with a repetition value over the defined threshold are decreased. After the stored over ranked alternatives have been treated, they are erased from the database. The number of stored over ranked alternatives to reach before evaluating them is the same than the one used in the on-line process to determine if there are enough over ranked alternatives to extract characteristics.

On the other hand, user selections can be stored and, after a certain number of choices have been made, they can be evaluated. By extracting characteristics from that set of stored selections, the most repeated attribute values, which are the ones of most interest to the user, can have its preference increased. After evaluating them, they are erased from the database. The number of selections stored before evaluating them is another parameter of the algorithm.

4.3. Adaptation mechanism

After the execution of the on-line and off-line stages, many characteristics can be found and the system can deduce a large amount of potential changes to do in the profile. Many of those changes, also named *adaptations*, although deduced using the reasoning techniques defined before, can be incorrect.

We have addressed this problem by restricting the number of adaptations that can be made to the profile for each execution of the adaptation process. A parameter limits the number of increases and decreases which can be made to a profile per adaptation step. For instance, a value of '2' allows a total of 4 changes: 2 increases and 2 decreases.

The set of adaptations is filtered according to the *level of evidence*, which is associated to all attribute values. In the case of decreases, this level is measured by counting the number of times that an attribute value is located in the over ranked alternatives. In the case of increases, the level measures the number of over ranked alternatives which do not have the associated attribute value. Finally, only the adaptations with the greatest levels will be performed over the current profile.

Moreover, some errors can be done by increasing/decreasing wrong preferences, because those levels of evidence are equal among many possible adaptations, including the wrong ones. Also, these errors can be repeated in different iterations.

To solve these problems, we introduced a *signed counter*, which regulates the final increase/decrease of the preference of the profile. When the adaptation process detects an evidence for a possible adaptation, it increases/reduces a counter, initially set to 0, associated to that attribute value (for example the *Morning* value of the attribute *Time*). This amount increased or decreased is the level of evidence explained before. When this counter reaches a certain value defined by a parameter, the increase/reduction is finally performed over the preference in the profile and the counter is reset to 0. Considering said parameter as +/-5 and following the example of Table 2, the system finds out that the *Morning* preference value of the attribute *Time* needs to be increased so it increases

its counter by *4*, since it only appears once among the five over ranked alternatives. So, the counter does not reach the necessary number to increase the preference value on the profile and it will be necessary another positive evidence to finally change it.

5. Evaluation

We have implemented a Web platform[2] that permits the definition of a problem (set of user criteria, aggregation and adaptation parameters), the definition of an initial profile, the definition of an ideal profile that we aim to learn, the random generation of a corpus of alternatives for the problem, the simulation of tests of user interaction with the system, and finally, the visualisation of the evolution of the user profile. The accesses to the platform are regulated by a user name and a password, allowing the maintenance of multiple users with different profiles for each recommendation domain. This approach permits to analyse the variability of results under different circumstances in the aggregation and adaptation stages. Users can upload data files with information about the alternatives that must be analysed in order to make a decision. The platform stores that information in a database, letting users to define a decision making problem on those alternatives.

In order to evaluate the framework, we proposed to iteratively measure the distance between the current user's profile and the ideal profile that we want to learn. The system selects the best rated alternative using the ideal profile. Then, the same alternatives are ranked using the current profile and, the ones that get better valuation than the one which was selected by the ideal profile, are considered as the over ranked alternatives. The selection and the over ranked alternatives are used by the adaptation user's profile to evaluate which changes should be performed on the user profile. To compare the results of different tests, we initialise the user profile randomly, but we maintain the ideal user profile.

All tests have been made using the same corpus of 3000 alternatives initialised randomly, with blocks of 15 alternatives per iteration. Profiles (ideal and initial) have been initialised manually (using the attributes explained in section 2) in order to obtain comparable results, and using the term set shown in Fig. 2. These tests have been performed using the extraction characteristics threshold of 30%, with the level of evidence +/-5 to consider a change in a value, and allowing only 2 changes in each sense in one iteration. Under these circumstances, we analyse the distance between the current (P) and the ideal (I) profiles, which is calculated as follows:

$$dist(I,P) = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{1}{m_j}\sum_{k=1}^{m_j}\frac{|COG_x(v(P,j,k)) - COG_x(v(I,j,k))|}{|COG_x(s_0) - COG_x(s_r)|}\right) \quad (3)$$

Here, n is the number of attributes, m_j is the cardinality of the attribute j, COG_x evaluates the coordinate x of the centre of gravity of a linguistic label [2], v is a function that returns the linguistic label associated to a profile, an attribute and a given value. With this function we obtain an average of the distances between all pairs of labels. The distance is normalised by using the extreme values of the fuzzy set $S = \{s_0, \ldots, s_r\}$. The distance is 0 when both profiles are identical. The maximum distance is 1 when all

[2]Website: http://itaka2-deim.urv.cat/recommender/

values contained in the user and the ideal profiles have just the opposite labels (s_0 and s_τ). Moreover, this function is commutative ($dist(P, I) = dist(I, P)$).

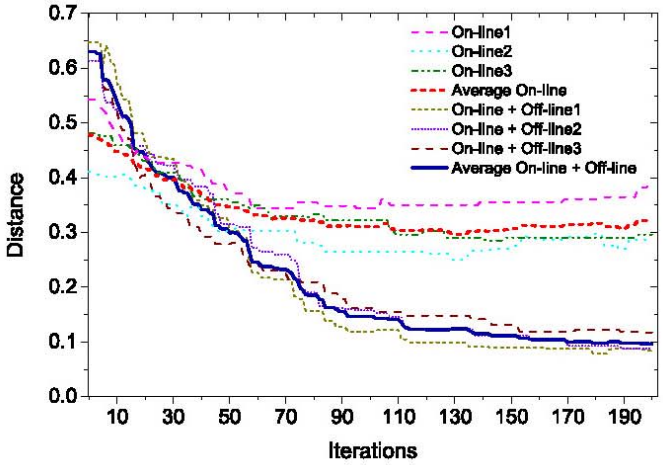


Figure 3. Distances between user and ideal profile

Fig. 3 compares the general performance of the adaptation algorithm using the on-line process, and using the on-line combined with the off-line stage. The figure shows the distances between the current and the ideal profile of three examples of each case and their average through 200 iterations. The three examples have different initial profiles but share the same ideal profile. The general performance accomplished when both processes are working together to adapt the profile is much better than when just using the on-line process. Eq. 3 gives a general evaluation of the performance of the adaptation algorithm.

6. Conclusions

The Web-based recommender system proposed in this paper has been designed as a general framework that permits to solve decision problems using multi-criteria decision aiding techniques. The separation between the domain-dependant (*e.g.*, adaptation thresholds) and the domain-independent data (*e.g.*, rating, ranking and adaptation processes) permits to understand the evolution of values, as well as to reuse the same methods in a wide range of problems.

The system has been prepared to make automatic simulations, which permit to study the influence of several parameters of the proposed adaptation algorithm. Further studies should be performed to analyse the influence of the fuzzy term set S, to determine the minimum number of over ranked alternatives to be taken into account in the off-line adaptation stage, and to study the percentage of elements to consider before to consider a characteristic.

The creation of this platform is the first step in a project that aims to develop techniques to adapt the user profile according to the selections made in the last step of the recommendation process. We consider that having automatic algorithms to update the preferences is required in many domain applications, where the preferences may change dynamically over time. The paper is focused in the implicit adaptation of the user profile

using linguistic variables, but we made preliminary attempts to deal also with numerical attributes. The final goal is to combine these two kinds of data into a single algorithm.

As stated before, this adaptation is focused on working in environments in which the user requires recommendations quite often, such as recommendation of films or restaurants, or even recommendations for tourists. Probably, this framework is not so adequate for punctual decisions such as 'buying a car'. However, under real settings like buying a car, the algorithm can be optimised with an initial profile learnt *a priori*. This set up stage can be performed through the creation of stereotyped profiles, such as 'teenager', 'family man', 'traveller', which permit to start the adaptation algorithm near the ideal profile. Another way to create the initial profile is requesting the user to fill in a form.

Acknowledgements

This work has been supported by the Universitat Rovira i Virgili (pre-doctoral grant of L. Marin, post-doctoral grant of D. Isern, and the project 2009AIRE-04), the Spanish Government (PlanÊ, Spanish Economy and Employment Stimulation Plan) and the Spanish Ministry of Science and Innovation (DAMASK project, TIN2009-11005).

References

[1] L. Baltrunas and X. Amatriain, 'Towards Time-Dependant Recommendation based on Implicit Feedback', in *Proceedings of the Workshop Context-aware Recommender Systems in conjunction with the 3rd ACM Conference on Recommender Systems, RecSys 09*, pp. 25–30, New York, USA, (2009).

[2] S. J. Chen and S. M. Chen, 'Fuzzy risk analysis based on similarity measures for generalized fuzzy numbers', *IEEE Transactions on Fuzzy Systems*, **11**(1), 45–56, (2003).

[3] E. Cheng, F. Jing, M. Li, W. Ma, and H. Jin, 'Using Implicit Relevance Feedback to Advance Web Image Search', in *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME 2006*, pp. 1773–1776, Toronto, ON, Canada, (2006). IEEE Computer Society.

[4] D. Isern, L. Marin, A. Valls, and A. Moreno, 'The Unbalanced Linguistic Ordered Weighted Averaging Operator', in *Proceedings of the 2010 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2010*, p. (to appear), Barcelona, Catalonia, (2010). IEEE Computer Society.

[5] M. Montaner, B. López, and J. L. de la Rosa, 'A Taxonomy of Recommender Agents on the Internet', *Artificial Intelligence Review*, **19**(3), 285–330, (2003).

[6] D. M. Nichols, 'Implicit Rating and Filtering', in *Proceedings of the 5th DELOS Workshop of Filtering and Collaborative Filtering*, pp. 31–36, Budapest, Hongary, (1997). European Research Consortium for Informatics and Mathematics.

[7] O. Noppens, M. Luther, T. Liebig, M. Wagner, and M. Paolucci, 'Ontology-supported Preference Handling for Mobile Music Selection', in *Proceedings of the Multidisciplinary Workshop on Advances in Preference Handling in conjunction with ECAI 2006*, pp. 96–101, Riva del Garda, Italy, (2006).

[8] S.-F. Pan and J.-H. Lee, 'eDAADe: An Adaptive Recommendation System for Comparison and Analysis of Architectural Precedents', in *Proceedings of the 4th International Conference of Adaptive Hypermedia and Adaptive Web-Based Systems, AH 2006*, eds., V. Wade, H. Ashman, and B. Smyth, volume 4018 of *Lecture Notes in Computer Science*, pp. 370–373, Dublin, Ireland, (2006). Springer Berlin / Heidelberg.

[9] M. Pazzani and D. Billsus, 'Learning and Revising User Profiles: The Identification of Interesting Web Sites', *Machine Learning*, **27**(3), 313–331, (1997).

[10] B. Roy, *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *International Series in Operations Research and Management Science. Lecture Notes in Economics and Mathematical Systems*, chapter Paradigms and Challenges, 3–24, Springer New York, 2005.

[11] R. R. Yager, 'Including Importances in OWA Aggregations Using Fuzzy Systems Modeling', *IEEE Transactions on Fuzzy Systems*, **6**(2), 286–294, (1988).