Computing and Informatics, Vol. 29, 2008, 453–481

A MODEL OF USER PREFERENCE LEARNING FOR CONTENT-BASED RECOMMENDER SYSTEMS

Tomáš Horváth

Institute of Computer Science
Faculty of Science
Pavol Jozef Šafárik University
Moyzesova 16
041 54 Košice, Slovakia
e-mail: tomas.horvath@upjs.sk

Revised manuscript received 24 February 2009

Abstract. This paper focuses to a formal model of user preference learning for content-based recommender systems. First, some fundamental and special requirements to user preference learning are identified and proposed. Three learning tasks are introduced as the exact, the order preserving and the iterative user preference learning tasks. The first two tasks concern the situation where we have the user's rating available for a large part of objects. The third task does not require any prior knowledge about the user's ratings (i.e. the user's rating history). Local and global preferences are distinguished in the presented model. Methods for learning these preferences are discussed. Finally, experiments and future work will be described.

Keywords: Content-based recommender systems, user preference learning, induction of fuzzy and annotated logic programs

Mathematics Subject Classification 2000: 68T05, 03B52, 03B80

1 INTRODUCTION

Imagine the problem of finding the appropriate accomodation – for our holiday or business travel – from the offer of several hotels (*objects*) with several properties (*attributes*). The values of these attributes and their combination have influence on

our choice[1]. E.g., while a student prefers cheap hotel with internet connection, not necessarily close to the city centre, a professor prefers hotels in the centre of the city with room service, even if these are a bit expensive. We often meet this type of classification in our everyday life, e.g. choice of car, tool, job, food, bank and even the partners, too.

The most important question is: What attributes of objects, what values of these attributes and in what importance are preferable for a given user or a type of user? In other words, what are the *user's preferences*?

The fact is that users often have just a slight idea about their preferences and/or they are not able to express them exactly (e.g. by an equation or a set of rules). Because of the mentioned reasons, it should be better to obtain preferences by some methods of data mining. We call this process *user preference learning*, and this is what our paper is adressed to.

A popular area of research, dealing with user preference learning are *recommender systems* [7, 40]. These are a specific type of information filtering technique that attempts to present to the user the objects the user is interested in.

This paper concerns *content-based* recommender systems [4, 33], in which the learning of the user's interests (preferences) is based on the features (attributes) of objects *rated*[2] by the user.

Hotel name	**Distance**	**Price**	**Equipment**	**User's evaluation**
Apple	100	99	nothing	1
Danube	1 300	120	tv	2
Cherry	500	99	internet	2
Iris	1 100	35	internet, tv	3
Lemon	500	149	nothing	1
Linden	1 200	60	internet, tv	3
Oak	500	149	internet, tv	2
Pear	500	99	tv	2
Poplar	100	99	internet, tv	2
Rhine	500	99	nothing	1
Rose	500	99	internet, tv	3
Spruce	300	40	internet	2
Themse	100	149	internet, tv	1
Tulip	800	45	internet, tv	3

Table 1. Illustrative example of the input to user preference learning

An input to user preference learning is illustrated in Table 1, where user rates hotels with *grades 1*, *2* and *3* (corresponding to levels *poor, good* and *excellent*,

[1] Similar to decathlon, where results from several disciplines aggregate the final result.

[2] Often a *Likert scale* is used as rating scale, where usually five or seven categories are distinguished from "strongly acceptable" to "strongly unacceptable". A comparison of ranking and rating is presented in [22].

respectively). Attributes of hotels are the name, the price for a room per person per night (in US dollars), the distance from the city center (in meters) and the room furnishings (internet connection, TV, both, or none of them).

The user's preferences (the output of user preference learning) corresponding to rating of objects in Table 1 are as follows:

- *IF* Distance ≥ 500 *AND* Price ≤ 99 *AND* equipment $= \{\text{tv, internet}\}$ *THEN* evaluation ≥ 3
- *IF* Distance ≥ 500 *AND* equipment $= \{\text{tv}\}$ *THEN* evaluation ≥ 2
- *IF* Price ≤ 99 *AND* equipment $= \{\text{internet}\}$ *THEN* evaluation ≥ 2
- *In other (resp. all) cases* evaluation ≥ 1

The aim of this paper is to propose a flexible *model* of user preference learning in the case when user rates objects. The proposed model should be applicable to arbitrary (type of) user and domain of objects. The structure of the paper is as follows: In the next section, consideration about needs and circumstances leading to creation of a new formal model for preference learning are discussed. At the end of the section, two types of requirements, namely fundamental and special requirements for user preference learning are set. Then, the proposed formal model of user preference learning follows which is the main goal of the paper. The exact, the order preserving and the iterative user preference learning tasks will be introduced. Following, statistical approaches are present for learning the so-called local preferences. Then, the method of induction of generalized annotated programs (IGAP) and the respective φ-GAP algorithm will be described for learning the so-called global preferences. Since this section is a short summary of [26] and [15], we will not discuss IGAP and φ-GAP in details. Finally, some experiments and further work will be discussed.

2 REQUIREMENTS FOR A FLEXIBLE MODEL OF USER PREFERENCE LEARNING

Some issues of user preference learning from the point of view of *representation* and *learning* will be discussed in this section.

2.1 Data Model

The first thing we have to consider are data models. We can take three data representation models into consideration: *object-attribute* model (one relation[3]), *multi-relational* model (several relations connected by key attributes) and the *RDF* model (triples of type object-attribute-value). Even if these models can be transformed into one another, each of them has its own particularities. So we have to consider all

[3] As in our illustrative example in Table 1.

of them. The multi-relational model[4] is appropriate for representing complex structures. On the other hand, from the view of computability, it is not very effective (we have to deal with joins of relations). An example of user preference learning from multi-relational data is introduced in [26]. This model can be transformed to object-attribute model by *propositionalisation*, but the resulting relation is often large and contains redundant data. It is mainly convenient for statistical approaches. A flexible model is the RDF which is used in the semantic web.

2.2 User's Rating

Now, let's have a look at user's ratings, e.g. from Table 1. It is clear that there are no clearly good and bad hotels. Rather, there is a *hierarchy* of hotels (in our case, excellent, good and poor). The users overall preference is monotonically dependent on the grade of fulfilment of single features of objects (hotels). Such an *ordinal* (monotone graded) *classification* is quite common, e.g. we classify students in school by grades, hotels by stars, investment safety of countries ranging from AAA to FFF, cars by categories, etc. In all these cases an object with higher classification should fulfil requirements for one with lower classification. For example, the hotel marked by *** fulfils requirements for hotels with grades ** and * (worse) and does not fulfil all requirements for hotels **** and ***** (better). Note that because the mentioned ordinality, we do not need to find rules for the lowest grade (poor) because every hotel fulfils requirements of the lowest grade (i.e. every hotel is at least poor).

2.3 Attribute Domain Ordering

As mentioned before, the specific property of the classification of hotels from our example is the natural order among the different classes. Moreover, there is an (partial) ordering in attribute domains, too. These orderings determine which values of the attribute are preferable to the user. In case of numerical attributes we distinguish four main types of such orderings. These are called *higher-best*, *lower-best*, *middle-best* or *marginal-best* according to which values (higher, lower, middle, marginal) of an attribute the given user prefers[5]. These main types of orderings are illustrated in Figure 1.

Thus, an attribute domain ordering can be viewed as a mapping $f : D \longrightarrow [0, 1]$, where D is an attribute domain and the mapped value from the unit interval $[0, 1]$ represents the impact (importance) of an attribute value.

Note that in case of non-numeric types of attributes we can use this concept, too.

[4] A multi-relational data model is not usual in recommender systems but a general formal model for preference learning should consider this model.

[5] Note that, in fact, these types of orderings are common.

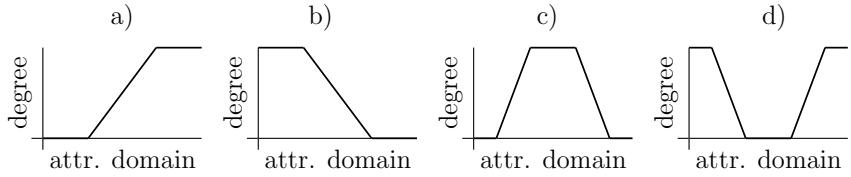

Fig. 1. Four main types of orderings of numerical attribute domains: a) *higher-best*, b) *lower-best*, c) *middle-best*, d) *marginal-best*

2.4 Attribute Type

The attributes can be of two basic types: *nominal* (without any ordering in the domain) and *ordinal* (with ordered domain). However, in our case we deal mainly with ordinal types; it can happen that some nominal attributes have big impact on the classification. An interesting attribute is the Boolean type attribute which can behave as nominal or ordinal (i.e. that an ordering between the true and false values is present). It means that the true values are better, or, conversely, the false values have positive impact to the classification[6]. From our point of view the most interesting attributes are the ordinal ones. These can be divided into two classes: *discrete* and *continuous*. In the case of discrete attributes an (partial) ordering of individual values or a disjunctive sets of values (disjunctive subsets of the domain) are present. In the case of continuous attributes we have a partial ordering of disjunctive intervals of values (disjunctive subintervals of the domain).

2.5 Imperfection

In a standard logical framework, we are restricted to represent only facts that are absolutely true. Thus, this framework is unable to represent and reason with imperfect – uncertain, vague, noisy, ranked/preferenced – information. This is a significant gap in the expressive power of the framework, and a major barrier to its use in many real-world applications. We use imperfection in the generic sense of uncertainty. Imperfection is unavoidable in the real world: our information (and particularly our classification) is often inaccurate and always incomplete, and only a few of the "rules" that we use for reasoning are true in all (or even most) of the possible cases. Furthermore, it is hard to represent the notions of a natural language just with two values (true, false). If we consider the concept "cheap", in a standard two-valued logic we can say that it holds (an object is cheap) or not (an object is not cheap). However, in two valued logic we cannot express easily that one item is cheaper than another. So, it is convenient to use several degrees of truth to the facts, in order to represent e.g. the concepts "more" or "less" cheap. For this purpose we can use the *multi-valued logical framework*, in which *a truth value* (accuracy, trustworthiness or

[6] It depends on the nature of the domain.

preference) is assigned to information. Beside probabilistic models[7] [45, 46] there is an extensive study of these phenomena in multiple-valued logic [44, 46].

Definition 1. *Fundamental requirements* for a flexible formal model for user preference learning are:

- Capability to represent and learn *ordinal ratings* (classification) of objects.
- Working with an arbitrary combination of *ordinal* and *nominal* attributes.
- Consideration of several types of *orderings of attribute domains* (at least, the above-mentioned higher-best, lower-best, middle-best and marginal-best types).

Definition 2. *Special requirements* for a flexible formal model for user preference learning are:

- Ability to deal with *several data models* (Object-Attribute, Multi-Relational, RDF).
- Representation of *imperfection* (uncertainty, vagueness, imprecision, ...) and the notions of a *natural language* (e.g. vague concepts, as "cheap", "near").

Notice that none of the recent approaches[8] to user preference learning cover all of these fundamental resp. special requirements at once.

3 THE USER PREFERENCE LEARNING MODEL

As stated in the previous chapter, there are some fundamental requirements, which a flexible formal model should fulfil (Definition 1). Assume that we have extensional data in relation $\mathbb{R}$ having attributes $\mathcal{A}, \mathcal{A}_1, \ldots, \mathcal{A}_n$ with domains $\mathbb{D}, \mathbb{D}_1, \ldots, \mathbb{D}_n$ in which the $\mathcal{A}$ attribute is a key, i.e. an identifier of an object. In other words, it is an object-attibute model where an object of interest is represented by values uniquely assigned to its attributes. For a tuple $(x, \mathbf{x}) \in \mathbb{R}$, where $x \in \mathbb{D}$ is an identifier of an object and $\mathbf{x} = (x_1, \ldots, x_n) \in \prod_{i \in \{1,\ldots n\}} \mathbb{D}_i$, this generates a mapping $data_{\mathbb{R}} : \mathbb{D} \longrightarrow \prod_{i \in \{1,\ldots n\}} \mathbb{D}_i$ defined as $data_{\mathbb{R}}(x) = \mathbf{x}$. In the following text, $(data_{\mathbb{R}}(x))_i$ will denote the ith component x_i of $\mathbf{x} = (x_1, \ldots, x_n)$.

Since we are interested in the case when user rates objects, take a look at user's ratings. Ratings can be expressed in natural language (poor, good, excellent) or by real numbers (e.g. normalized to the unit interval $[0, 1]$). Here the *preference degrees*[9] are assumed to be *totally (linearly) ordered*[10]. In general, user's preference degrees can be represented by (or transformed to) real numbers from the unit interval $[0, 1]$.

[7] Probability theory models uncertainty by assigning a probability to each of the states of the world that an agent considers possible.

[8] Discussed later in this work.

[9] User's ratings are often called preference degrees, too.

[10] The position is that partially ordered ratings – without contradictions, cycles – can be realised by a linear extension of these ratings.

Thus, if a user u rates objects, he/she determines a linear ordering of these objects, i.e. an ordering $\preceq^u_{\mathbb{D}}$ on the domain $\mathbb{D}$. Note that an ordering $\preceq^u_{\mathbb{D}}$ is distinctive (and thus subjective and unique) for an individual user u. In case of m users we denote their orderings as $\preceq^{u_1}_{\mathbb{D}}, \ldots, \preceq^{u_m}_{\mathbb{D}}$.

Partial orderings $\preceq^u_{\mathbb{D}_1}, \ldots, \preceq^u_{\mathbb{D}_n}$ on the attribute domains $\mathbb{D}_1, \ldots, \mathbb{D}_n$ are an important part of our model. Notice that these orderings have the same meaning as the above-mentioned four main types of orderings of attribute domains, illustrated in Figure 1. Again, these orderings are distinctive for an individual user u. In case of m users, we denote these orderings as $\preceq^{u_1}_{\mathbb{D}_1}, \ldots, \preceq^{u_1}_{\mathbb{D}_n}, \ldots, \preceq^{u_m}_{\mathbb{D}_1}, \ldots, \preceq^{u_m}_{\mathbb{D}_n}$.

Definition 3. Suppose an ordering $\preceq$ on the set $\mathbb{X}$ and a function $p : \mathbb{X} \longrightarrow [0, 1]$. Then we say that $\preceq$ is *generated* by p if the following holds: $(\forall x, y \in \mathbb{X}) \quad x \preceq y$ iff $p(x) \leq p(y)$. In this case, we can denote $\preceq_p$.

Definition 4 (Monotone dataset)**.** Suppose a dataset $\mathbb{R} \subseteq \mathbb{D} \times \prod_{i\in\{1,\ldots,n\}} \mathbb{D}_i$, a linear ordering $\preceq^u_{\mathbb{D}}$ on $\mathbb{D}$ and partial orderings $\preceq^u_{\mathbb{D}_1}, \ldots, \preceq^u_{\mathbb{D}_n}$ on $\mathbb{D}_1, \ldots, \mathbb{D}_n$, and the mapping $data_{\mathbb{R}} : \mathbb{D} \longrightarrow \prod_{i\in\{1,\ldots,n\}} \mathbb{D}_i$.
If for any two objects $(x, \mathbf{x}), (y, \mathbf{y}) \in \mathbb{R}$ it holds

$$(\forall i \in \{1, \ldots, n\})(data_{\mathbb{R}}(x))_i \preceq^u_{\mathbb{D}_i} (data_{\mathbb{R}}(y))_i \Longrightarrow x \preceq^u_{\mathbb{D}} y \tag{1}$$

we call the dataset[11] $\mathbb{R}$ *monotone w.r.t. orderings* $\preceq^u_{\mathbb{D}}, \preceq^u_{\mathbb{D}_1}, \ldots, \preceq^u_{\mathbb{D}_n}$.

Definition 5 (User's preferences)**.** Suppose a dataset $\mathbb{R} \subseteq \mathbb{D} \times \prod_{i\in\{1,\ldots,n\}} \mathbb{D}_i$ having attributes $\mathcal{A}, \mathcal{A}_1, \ldots, \mathcal{A}_n$ with domains $\mathbb{D}, \mathbb{D}_1, \ldots, \mathbb{D}_n$. Suppose that $\mathbb{R}$ is monotone w.r.t. orderings $\preceq^u_{\mathbb{D}}, \preceq^u_{\mathbb{D}_1}, \ldots, \preceq^u_{\mathbb{D}_n}$, where $\preceq^u_{\mathbb{D}}$ is generated by an $p^u_G : \mathbb{D} \longrightarrow [0, 1]$ and $\preceq^u_{\mathbb{D}_i}$ are generated by mappings $p^u_{L_i} : \mathbb{D}_i \longrightarrow [0, 1]$.

Then the mapping p^u_G is called *user's global preferences* and the mappings[12] $p^u_{L_i}$ are called *user's local preferences on* $\mathbb{R}$ *for attributes* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *compatible to user's global preferences* p^u_G. In this case, we say that $\mathbb{R}$ is monotone w.r.t. p^u_G and $p^u_{L_1}, \ldots, p^u_{L_n}$.

As can be seen from Definition 5, we consider user's preferences on the whole domain of objects. The reason is the following: we assume that users have their fixed preferences at certain time, valid for all objects, even if these are hard to determine exactly. For example, if one prefers cheap hotels, then this preference is valid for all hotels, and doesn't depend on concrete objects (hotels). Of course, preferences can change with time, but are stable at a certain moment; e.g. if a user gets good job, he/she is assumed to prefer medium price hotels.

[11] In the following text, we will use shorter notation "monotone dataset", instead of "monotone dataset w.r.t. orderings $\preceq^u_{\mathbb{D}}, \preceq^u_{\mathbb{D}_1}, \ldots, \preceq^u_{\mathbb{D}_n}$".

[12] In the following text, we will use shorter notation "user's local preferences", instead of "user's local preferences on $\mathbb{R}$ for attributes $\mathcal{A}_1, \ldots, \mathcal{A}_n$ compatible to user's global preferences p^u_G".

Note that some exceptions can occur, when user likes objects, which don't fulfil his/her known preferences. Usually this is because of other, previously not considered attributes. For example, user can prefer a hotel which has a lovely receptionist, although it is a little bit expensive. In this case, "loveliness" of receptionist can be considered as an additional (or "hidden") attribute.

An important question arises: Do user's local preferences always exist? An example of dataset, where user's local preferences cannot be found is given in the next example.

Example 1. Suppose the dataset $\mathbb{R} = \{(a,0,0),(b,0,1),(c,1,0),(d,1,1)\} \subseteq \mathbb{D} \times \mathbb{D}_1 \times \mathcal{D}_2$ where $\mathbb{D} = \{a,b,c,d\}, \mathbb{D}_1 = \mathbb{D}_2 = \{0,1\}$. The mapping p^u_G (generating an ordering $\preceq^u_{\mathbb{D}}$ is defined as $p^u_G(a) = p^u_G(d) = 0, p^u_G(b) = p^u_G(c) = 1$. The situation is similar to classical XOR operator, as can be seen in Figure 2.

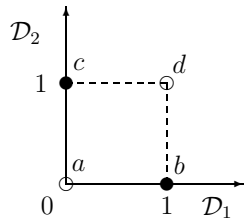

Fig. 2. Example of a XOR-typed dataset

There are three possible cases of mappings $p^u_{L_1}$ for the domain $\mathbb{D}_1$, namely, when $p^u_{L_1}(0) < p^u_{L_1}(1), p^u_{L_1}(0) > p^u_{L_1}(1)$ or $p^u_{L_1}(0) = p^u_{L_1}(1)$. Similar situation arises in case of mapping $p^u_{L_2}$ for the domain $\mathbb{D}_2$. Note that the concrete values of these functions (which are from the unit interval $[0,1]$) do not matter. Thus, there are nine different combinations of mappings $p^u_{L_1}$, $p^u_{L_2}$, which generate orderings $\preceq^u_{\mathbb{D}_1}, \preceq^u_{\mathbb{D}_2}$, respectively. None of these combinations is appropppriate to fulfil Condition 1 for a monotone dataset (and thus for local preferences, too). Note that for the dataset $\mathbb{R}$ with user's global preferences p^u_G, local preferences do not exist.

Notice that a non-monotone dataset, i.e. in which local preferences do not exist can be denoted as *XOR-type dataset*.

Let's modify Example 1 by adding an attribute to the data model as can be seen in the following example:

Example 2. The dataset $\mathbb{R} = \{(a,0,0,\diamondsuit),(b,0,1,\heartsuit),(c,1,0,\heartsuit),(d,1,1,\diamondsuit)\} \subseteq \mathbb{D} \times \mathbb{D}_1 \times \mathbb{D}_2 \times \mathbb{D}_3$ where $\mathbb{D} = \{a,b,c,d\}, \mathbb{D}_1 = \mathbb{D}_2 = \{0,1\}$ and $\mathbb{D}_3 = \{\heartsuit,\diamondsuit\}$. The mapping p^u_G is equal to that in Example 1, i.e. $p^u_G(a) = p^u_G(d) = 0, p^u_G(b) = p^u_G(c) = 1$.
Now, define the following mappings $p^u_{L_i} : \mathbb{D}_i \longrightarrow [0,1]$ as follows: $p^u_{L_1}(x) = p^u_{L_2}(y) = c$ for every $x \in \mathbb{D}_1, y \in \mathbb{D}_2$, where $c \in [0,1]$ is a constant and $p^u_{L_3}(\diamondsuit) = \frac{1}{3}$, $p^u_{L_3}(\heartsuit) = \frac{2}{3}$. This situation is illustrated in Figure 3.

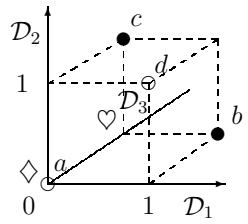

Fig. 3. Illustration of data from Example 2

Now, the mappings $p^u_{L_i}$ and p^u_G generate orderings which fulfil Condition 1 and makes the dataset $\mathbb{R}$ monotone. Thus, $p^u_{L_1}, p^u_{L_2}$ and $p^u_{L_3}$ can be declared as user's local preferences.

Definition 6. A user's local preference $p^u_{L_i}$ defined as $(\forall x \in \mathbb{D}_i)\ p^u_{L_i}(x) = c$, where $c \in [0,1]$ is called *constant user's local preference for attribute $\mathcal{A}_i$ of the domain $\mathbb{D}_i$.*

Note that a constant user preference indicates that an attribute to which it belongs has no certain ordering on its domain, i.e. all values have the same importance.

Definition 7 (Exact user preference learning task)**.** Suppose a monotone dataset $\mathbb{R} \subseteq \mathbb{D} \times \mathbb{D}_1 \times \ldots \times \mathbb{D}_n$ w.r.t. orderings $\preceq^u_{\mathbb{D}}, \preceq^u_{\mathbb{D}_1}, \ldots, \preceq^u_{\mathbb{D}_n}$, where $\preceq^u_{\mathbb{D}}$ is generated by $p^u_G : \mathbb{D} \longrightarrow [0,1]$.

The exact task of user preference learning has a sample set (training set) $\mathbb{S} \subseteq \mathbb{R}$ and $p^u_G \restriction \mathbb{S}$ on input. The task is to find mappings $p^u_{L_i} : \mathbb{D}_i \longrightarrow [0,1]$ which generate $\preceq^u_{\mathbb{D}_i}$, and find a mapping $@ : [0,1]^n \longrightarrow [0,1]$, such that

$$(\forall o \in \mathbb{D} \restriction \mathbb{R}) \quad @(p^u_{L_1}((data_{\mathbb{R}}(o))_1), \ldots, p^u_{L_n}((data_{\mathbb{R}}(o))_n) = p^u_G(o) \tag{2}$$

where @ is called *aggregation function.*

As can be seen, a user preference learning task consists of two individual *learnig* steps: to find user's local preferences and an aggregation function @. We will call the value of $@(p^u_{L_1}((data_{\mathbb{R}}(o))_1), \ldots, p^u_{L_n}((data_{\mathbb{R}}(o))_n)$ *the learned value for an object o by an user preference learning method.*

Note that the presented user preference learning tasks do not depend on a concrete learning method. It can be chosen arbitrarily, depending on the concrete domain and application.

Naturally, if we demand effective (resp. on-line) computation, the mappings $p^u_{L_1}, \ldots, p^u_{L_n}$ and @ have to be learned on a small dataset, instead of $\mathbb{R}$, but the results must be valid for the whole $\mathbb{R}$. Thus, the accuracy of computation determines how precise are the local $(p^u_{L_i})$ and global (@) preferences, learned from a small subset of the domain w.r.t. the whole domain of objects.

Such an accuracy, which determines how precisely can be the user's preferences (valid on the whole domain) predicted by learning from a small dataset, is called *prediction accuracy.* It can be expressed in the following form:

$$\text{exact prediction accuracy} = \frac{\text{number of correctly learned objects}}{\text{number of all objects}} \tag{3}$$

where an object is learned correctly, if the learned value for the object is equal to user's rating of this object (i.e. the Condition 2 holds for the given object o).

An accuracy of learning on a mentioned small dataset (often called training set) is called *training accuracy*. It can be defined as usual in the concrete learning methods used in the learning process[13].

Note that in general the training accuracy is higher than the prediction accuracy.

Since it is often hard to find an aggregation function @ having exactly the same values on $\mathbb{R}$ as user's global preferences p_G^u, a task with a weaker condition is needed, whose main idea is that an aggregation function can't change the ordering of objects w.r.t. user's rating (i.e. his/her global preferences).

Definition 8 (Order preserving user preference learning task)**.** Suppose a monotone dataset $\mathbb{R} \subseteq \mathbb{D} \times \mathbb{D}_1 \times \ldots \times \mathbb{D}_n$ w.r.t. orderings $\preceq_{\mathbb{D}}^u, \preceq_{\mathbb{D}_1}^u, \ldots, \preceq_{\mathbb{D}_n}^u$, where $\preceq_{\mathbb{D}}^u$ is generated by $p_G^u : \mathbb{D} \longrightarrow [0,1]$.

The order preserving task of user preference learning has a sample set (training set) $\mathbb{S} \subseteq \mathbb{R}$ and $p_G^u \restriction \mathbb{S}$ on input. The task is to find mappings $p_{L_i}^u : \mathbb{D}_i \longrightarrow [0,1]$ which generate $\preceq_{\mathbb{D}_i}^u$, and to find a mapping $@ : [0,1]^n \longrightarrow [0,1]$, such that

$$\begin{aligned}(\forall o_r, o_s \in \mathbb{D})(\forall i \in \{1,\ldots,n\}) \quad & p_G^u(o_r) < p_G^u(o_s) \Longrightarrow \\ @(p_{L_1}^u((data_{\mathbb{R}}(o_r))_1), \ldots, p_{L_n}^u((data_{\mathbb{R}}(o_r))_n) \leq \\ \leq @(p_{L_1}^u((data_{\mathbb{R}}(o_s))_1), \ldots, p_{L_n}^u((data_{\mathbb{R}}(o_s))_n)\end{aligned} \tag{4}$$

where @ is called *aggregation function*[14].

The left side of the above implication determines that an object o_s is better in all attributes than an object o_r. The aim is to get a mapping (@) which does not evaluate o_s worse than o_r. Note that it is sufficient to define the mapping @ as equal to a constant value. This result will be correct, albeit useless.

In this case, the prediction accuracy is similar to exact prediction accuracy:

$$\text{order preserving pred. accuracy} = \frac{\text{number of correctly learned pairs}}{\text{number of all pairs}} \tag{5}$$

where a pair of objects is learned correctly, if the Condition 4 holds for the given pair of objects.

Figure 4 illustrates the situation presented in Definitions 7, 8. First step of our model assumes attribute domains $\mathbb{D}, \mathbb{D}_i$ and orderings $\preceq_{\mathbb{D}}^u, \preceq_{\mathbb{D}_i}^u$ (depending on user) which build a monotone dataset $\mathbb{R}$ (where $i \in \{1,\ldots,n\}$). Moreover, $\preceq_{\mathbb{D}}^u$ is

[13] The training accuracy will be discussed with the learning algorithm later.

[14] An aggregation function is often mentioned as utility function.

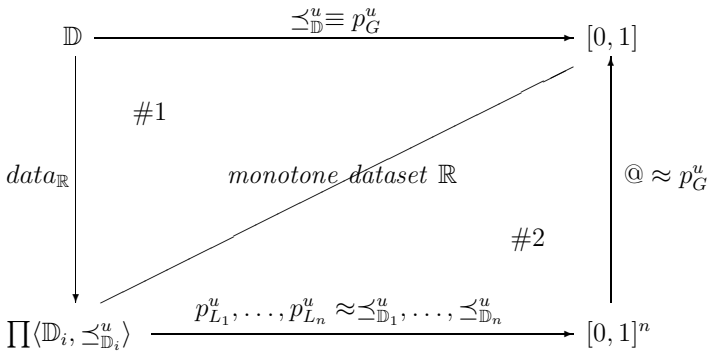

Fig. 4. Commutative diagram of our formal model of user preference learning

generated by a mapping p^u_G. This is illustrated in the #1 part of the commutative Diagram 4.

Further, we assume that orderings $\preceq_{\mathbb{D}_i}$ can be approximated (resp. generated) by mappings $p^u_{L_i}$. Let us note that these are very strong assumptions in general, in practice for finite domains and a finite subset of [0, 1], these mappings can be found (subject to some decrease of precision of the used learning method).

Finally, to complete the commutative diagram #2, we have to find an aggregation function @, which approximates p^u_G (and thus generates $\preceq^u_{\mathbb{D}}$), such that the diagram commutes.

The above-described of user preference learning tasks assumes that an ordering $\preceq^u_D$ generated by mapping p^u_G is known (on the whole $\mathbb{R}$). Thus, these can be mainly used for off-line computation of user preferences. For example, we have a huge amount of objects and their ratings by many users stored and we want to know which user was having stable preferences during the time (in this case the sample sets correspond to given time periods).

Now, imagine that we do not know the mapping p^u_G (or an ordering $\preceq^u_D$). The situation is as follows: The user gets some objects which she/he evaluates. Thus we have the sample set $\mathbb{S}$ and the mapping $p^u_G : \mathbb{S} \longrightarrow [0, 1]$, which form an input to the computation of the – local and global – preferences of a given user (i.e. the mapings $p^u_{L_i}$ and @). Note that @ approximates p^u_G and thus generates the ordering $\preceq^u_{\mathbb{D}}$ of objects from $\mathbb{R}$. Using $\preceq^u_{\mathbb{D}}$ we can choose the *top-k*[15] (i.e. the best) objects from the whole $\mathbb{R}$. These objects are again rated by the user and the whole process is repeated. This approach is called *PHASES* [14]. The respective task to this approach is introduced in Definition 9 and illustrated in Figure 5.

Definition 9 (Iterative user preference learning task with samples of size k)**.** Suppose $\mathbb{R} \subseteq \mathbb{D} \times \mathbb{D}_1 \times \ldots \times \mathbb{D}_n$ is a dataset and $\mathbb{S} \subseteq \mathbb{R}$ is an initial sample.

[15] An efficient top-k object search algorithm can be found in [21].

The iterative user preference learning task in *phase* 0 has on input $\mathbb{S} \subseteq \mathbb{R}$ and the mapping $p^u_{G,0} : \mathbb{S} \longrightarrow [0,1]$. The iterative user preference learning task in *phase* 0 has on output mappings $p^u_{L_{j,0}} : \mathbb{D}_i \longrightarrow [0,1]$,$@_0 : [0,1]^n \longrightarrow [0,1]$ and the set $\mathbb{S}_0 = topk(\mathbb{R})$ of objects computed w.r.t. ordering $\preceq_{\mathbb{D},0}$ of all objects $o \in \mathbb{D}$ generated by $@_0(p^u_{L_{1,0}}(data_{\mathbb{R}}(o))_1, \ldots, p^u_{L_{n,0}}(data_{\mathbb{R}}(o))_n)$.

The iterative user preference learning task in *phase* $j+1$ has on input $\mathbb{S}_j$ and the mapping $p^u_{G,j+1} : \mathbb{S}_j \longrightarrow [0,1]$. The iterative user preference learning task in *phase* $j+1$ has on output mappings $p^u_{L_{i,j+1}} : \mathbb{D}_i \longrightarrow [0,1]$,$@_{j+1} : [0,1]^n \longrightarrow [0,1]$ and the set $\mathbb{S}_{j+1} = topk(\mathbb{R})$ of objects computed w.r.t. ordering $\preceq_{\mathbb{D},j+1}$ of all objects $o \in \mathbb{D}$ generated by $@_{j+1}(p^u_{L_{1,j+1}}(data_{\mathbb{R}}(o))_1, \ldots, p^u_{L_{n,j+1}}(data_{\mathbb{R}}(o))_n)$, where $topk(\mathbb{R})$ can be an arbitrary procedure which gets top-k objects according to some orderings (thus $|\mathbb{S}_i| = k$).

The accuracy of iterative user preference learning task expresses how effectively the user's preferences are learned in several phases, i.e. if the offered $topk(\mathbb{R})$ objects computed w.r.t. $\preceq_{\mathbb{D},j}$ are better than the $topk(\mathbb{R})$ objects computed w.r.t. $\preceq_{\mathbb{D},j-1}$. This accuracy can be computed by several ways, such as:

$$\tau\text{-iterative correlation}_i = \tau(\langle \mathbb{S}_i, \preceq_{\mathbb{D},i} \rangle, \langle \mathbb{S}_i, \preceq_{G,i+1} \rangle) \tag{6}$$

where $\preceq_{G,i+1}$ is generated by the user's evaluation $p^u_{G,i+1}$ of a sample set $\mathbb{S}_i$ and $\tau(X,Y)$ refers to the *Kendall tau rank correlation coefficient*[16] [1] computed for ordered sets X and Y.

$$\text{average iterative rating}_i = \frac{\sum\limits_{x \in \mathbb{S}_i} p^u_{G,i+1}(x)}{k} \tag{7}$$

An important step is the selection of an initial sample of objects. One solution is the random selection. On the other hand, if we have the previously computed user preferences available (e.g. by an off-line computation from the user history), we can use these preferences to select the top-k objects to initial sample set.

As can be seen, the presented model meets the fundamental requirements (Definition 1): it is capable to represent and learn ordinal ratings ($@$) and attribute value orderings ($p^u_{L_i}$). If we consider an attribute with constant user's local preferences as nominal, we have included the remaining fundamental requirement (i.e. to deal with an arbitrary combination of nominal and ordinal attributes). The fulfilment of the special requirements (Definition 2) in this model is determined by the concrete learning method or framework[17] used in the learning process.

[16] Other correlation coefficients can be used, too.

[17] The proposed IGAP method will be introduced later in this work.

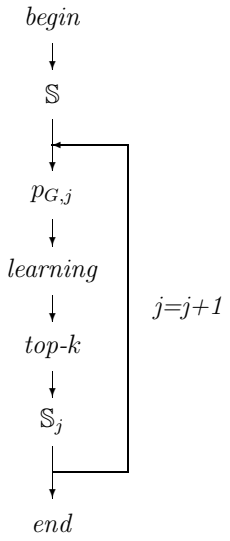

Fig. 5. The PHASES approach

4 LOCAL PREFERENCE LEARNING

As mentioned in the previous section, learning user's global preferences is preceded by learning user's local preferences. We concentrate on two methods for local preference learning in this section: one is used for non-numerical attributes (discussed in [15]) and one for numerical attributes (introduced in [14]).

In local preference learning, the main problem is whether the overall evaluation of objects depends on given attributes (e.g. price, distance) according to some ordering of the domain of these attributes.

We learn attribute orderings for every attribute, from the projections of attribute values to user's evaluation. As mentioned in the previous section, we assume finite domains and a finite number of evaluations (grades).

Thus, in case of objects $o_1, \ldots, o_m \in \mathbb{D}$, for every attribute $\mathcal{A}_i$, the set $L_{\mathcal{A}_i} = \{\langle (data_{\mathbb{R}}(o_j))_i, p^u_G(o_j)\rangle | j = 1, \ldots, m\}$ is considered. To allow numerical computations, we need to transform user's evaluation to numbers[18], if these are expressed in natural language (e.g. "poor"=1, "good"=2 and "excellent"=3).

4.1 Non-Numerical Attributes

If an attribute $\mathcal{A}_i$ with the domain $\mathbb{D}_i$ is non-numerical, we have two choices:

[18] Note that there is a linear ordering of user's evaluations, so the transformation to numbers can be made easily.

- We leave the computation of local preferences and consider the attribute $\mathcal{A}_i$ as nominal.
- The other choice is to compute for every $X \subseteq \mathbb{D}_i$ the *average* value of user's evaluation of objects having X in attribute $\mathcal{A}_i$. Thus, local preferences can be computed as

$$p_{L_i}(X) = \frac{\sum\limits_{\{o \in \mathbb{D} | (data_{\mathbb{R}}(o))_i = X\}} p^u_G(o)}{|\{o \in \mathbb{D} | (data_{\mathbb{R}}(o))_i = X\}|}. \tag{8}$$

  To make the computation more general, in case of non-numerical attributes we use sets of attribute values $X \subseteq \mathbb{D}_i$ instead of single attribute values $x \in \mathbb{D}_i$. It allows to deal with more values for one attribute (e.g. in the illustrative example in Table 1, an *equipment* attribute can be $\{tv, internet\}$).

Example 3. In case of the illustrative example (Table 1), local preferences for a non-numerical attribute "equipment" can be computed as follows:

- $p^u_{L_{equipment}}(tv) = \frac{user'sevaluation(Danube)+user'sevaluation(Pear)}{2} = \frac{2+2}{2} = 2$
- $p^u_{L_{equipment}}(internet) = \frac{2+2}{2} = 2$
- $p^u_{L_{equipment}}(tv, internet) = \frac{3+3+2+2+3+1+3}{7} = 2\frac{3}{7}$
- $p^u_{L_{equipment}}(nothing) = \frac{1+1+1}{3} = 1$

Since we defined local preferences as a mapping of the domain to the unit interval $[0, 1]$, we can normalize the values of $p^u_{L_{equipment}}(X)$ to this interval. In this case, we can divide the computed values by 3, thus we get $p^u_{L_{equipment}}(tv) = \frac{2}{3}$, $p^u_{L_{equipment}}(internet) = \frac{2}{3}$, $p^u_{L_{equipment}}(tv, internet) = \frac{17}{21}$, $p^u_{L_{equipment}}(nothing) = \frac{1}{3}$

Thus, the corresponding (partial) orderings on the domain of an attribute equipment from Table 1 are as follows: $\{nothing\} \preceq^u_{equipment} \{tv\}, \{internet\} \preceq^u_{equipment} \{tv, internet\}$.

Note that it can happen that the local preferences for an attribute can't be computed correctly. It is when there are not substantial differences between $p^u_{L_i}(X)$ for the values $X \subseteq \mathbb{D}_i$ of an attribute. We can view such local preferences as constant-local preferences and consider the given attribute as nominal.

4.2 Numerical Attributes

As mentioned, we learn local preferences from the projection of attribute values to user's evaluation. Such a projection is illustrated in Figure 6.

Usually, noncomplicated expressions of local preferences are desirable. Since an aggregation of local preferences forms global preferences, more complex local preferences indicate more complex global preferences.

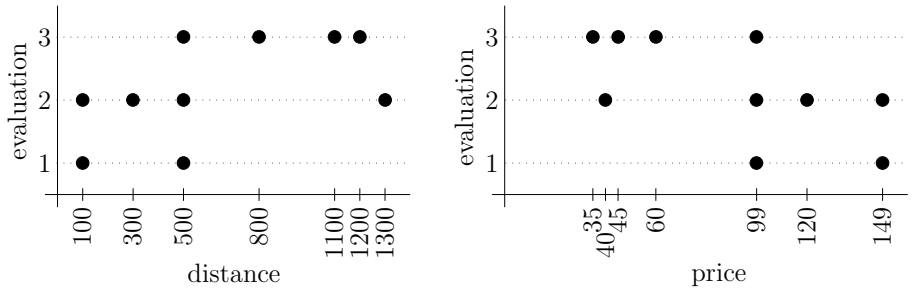

Fig. 6. Projection of attribute values (distance or price) to user's evaluation, corresponding to Table 1

So-called *simplified types of numerical attribute domain orderings* are considered, as illustrated in Figure 7. These are similar to ordering types, introduced in Figure 1, but more effectively usable in computation (even if we have to consider a slight decrease of correctness).

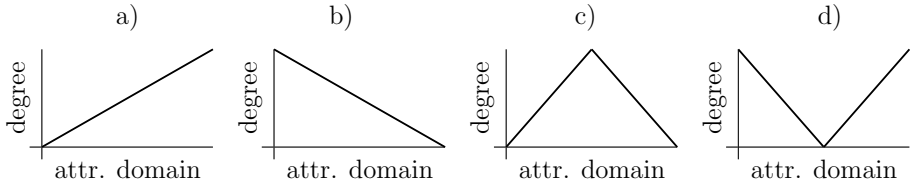

Fig. 7. Simplified types of orderings of numerical attribute domains: a) *simple higher-best* b) *simple lower-best* c) *simple middle-best* d) *simple marginal-best* related to ordering types, introduced in Figure 1

The main goal is to detect that area in the domain, in which the values are more preferable. This area is represented by one value, called *critical value* (see Figure 7), in contrast to basic ordering types (Figure 1), where this area was an interval.

First, we have to compute the quadratic polynomial on the projection of attribute values to user's evaluation, i.e. the quadratic polynomial regression task with an input $L_{\mathcal{A}_i}$ (introduced at the beginning of the chapter). Such computed polynomials are illustrated by solid lines in Figure 8 in Example 4.

Then, we have to find out the critical value c_i of an attribute $\mathcal{A}_i$, in which the polynomial has its global minimum or global maximum. If the critical value is out of the used domain of an attribute or is inside the domain, but very "near" to borders of the used domain, the ordering will be of the lower-best or higher-best type. If the critical value is somewhere in the middle of the used domain then the ordering will be of the middle-best or marginal-best type. There are six possible situations, as illustrated in Table 2, where $min(\mathbb{D}_i)$ and $\max(\mathbb{D}_i)$ are the minimal and maximal

values of attribute $\mathcal{A}_i$, present in the dataset, and ϵ represents some measure of *nearness* to borders of the values of a given attribute in the dataset.

c_i	$c_i > \max(\mathbb{D}_i) - \epsilon$	$c_i < min(\mathbb{D}_i) + \epsilon$	$c_i \in \langle min(\mathbb{D}_i) + \epsilon, \max(\mathbb{D}_i) - \epsilon \rangle$
min	*simple lower-best*	*simple higher-best*	*simple marginal-best*
max	*simple higher-best*	*simple lower-best*	*simple middle-best*

Table 2. Possible (simplified) types of orderings, detected by quadratic polynomial regression, according to the position of the *critical value* c_i (columns) and to the case, if c_i is the global maximum minimum of the polynomial (rows)

Finally, we compute the local preferences $p^u_{L_i}(x)$ for that $x \in \mathbb{D}_i$, which are present in the dataset. The computation depends on the detected simplified type of ordering of the attribute $\mathcal{A}_i$. Thus, we distinguish four cases:

- simple higher-best type of ordering of $\mathcal{A}_i$

$$p^u_{L_i}(x) = \frac{x - \min(\mathbb{D}_i)}{\max(\mathbb{D}_i) - \min(\mathbb{D}_i)} \tag{9}$$

- simple lower-best type of ordering of $\mathcal{A}_i$

$$p^u_{L_i}(x) = \frac{\max(\mathbb{D}_i) - x}{\max(\mathbb{D}_i) - \min(\mathbb{D}_i)} \tag{10}$$

- simple middle-best type of ordering of $\mathcal{A}_i$

$$p^u_{L_i}(x) = \begin{cases} \frac{x - \min(\mathbb{D}_i)}{c_i - \min(\mathbb{D}_i)}, & \text{if } x \leq c_i, \\ \frac{\max(\mathbb{D}_i) - x}{\max(\mathbb{D}_i) - c_i}, & \text{else} \end{cases} \tag{11}$$

- simple marginal-best type of ordering of $\mathcal{A}_i$

$$p^u_{L_i}(x) = \begin{cases} \frac{c_i - x}{c_i - \min(\mathbb{D}_i)}, & \text{if } x \leq c_i, \\ \frac{x - c_i}{\max(\mathbb{D}_i) - c_i}, & \text{else.} \end{cases} \tag{12}$$

Example 4. We compute the local preferences for the illustrative example (Table 1) as follows: The quadratic polynomials for the attributes distance and price are computed by a standard least squares regression:

$$polynom_{distance}(x) = -0.000000963524x^2 + 0,00250191x + 1.02457$$

$$polynom_{price}(x) = +0.0000172466x^2 - 0,0161825x + 3.36741$$

for which the critical values are $c_{distance} = 1298.31$ (global maximum) respectively $c_{price} = 469.15$ (global minimum).

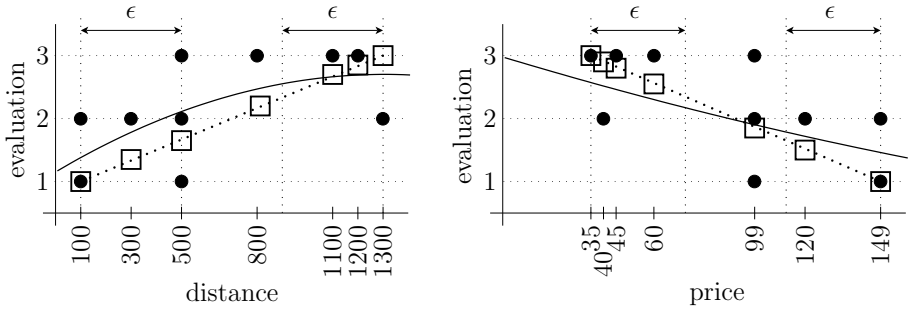

Fig. 8. Learning local preferences by quadratic polynomial regression. The solid lines represent the polynomials, the dotted lines represent the simple higher-best and lower-best types of orderings. The exact values of $p^u_{L_{distance}}$ resp. $p^u_{L_{price}}$ are represented by squares.

The local preferences are computed[19] by using the Equations (9), (10):

$$100 \preceq^u_{distance} 300 \preceq^u_{distance} \cdots \preceq^u_{distance} 1200 \preceq^u_{distance} 1300$$

or

$$149 \preceq^u_{price} 120 \preceq^u_{price} \cdots \preceq^u_{price} 40 \preceq^u_{price} 35.$$

5 GLOBAL PREFERENCE LEARNING

A multi-relational learning method with the ability to deal with imperfect information should fulfil all requirements to flexible learning at once (Definitions 1 and 2).

Inductive logic programming (ILP) [13] combines first-order logic and machine learning algorithms. Briefly, the task of ILP is to find a correct hypothesis from the sets of positive and negative examples under the presence of background knowledge.

To fulfil all requirements to flexibe user preference learning (Definitions 1 and 2) we have to join ILP with generalized annotated programs (GAP) [30]. Thus an inductive GAP model was developed, called IGAP [26][20].

The language of GAP consists of qualitative and quantitative part. The qualitative part of GAP language is the usual language of predicate logic (with variables, constants, predicates and function symbols). The quantitative part of the language in our approach is typed (sorted) and for each logical predicate p there is a (possibly different) truth values set T_p with ordering $\leq_p$.

[19] Note that $\min(\mathbb{D}_{distance}) = 100$, $\max(\mathbb{D}_{distance}) = 1300$ and $\min(\mathbb{D}_{price}) = 35$, $\max(\mathbb{D}_{price}) = 149$. Ususally, we choose the value of ϵ as the 30 % of the used domain (i.e. 400 for the attribute distance and 50 for the attribute price).

[20] Note that we do not know about any inductive GAP system.

A *rule* of GAP is an implication $A : \rho(\mu_1, \ldots, \mu_k) \leftarrow B_1 : \mu_1 \wedge \ldots \wedge B_k : \mu_k$, where $B : \mu_1, \ldots, B : \mu_k$, and $\mu \in [0, 1]$ are so-called variable-annotated atoms and $A : \rho$ is a possibly complex annotated atom.

A mapping $f : \mathcal{B}_L \rightarrow [0, 1]$ is a *Herbrand interpretation for annotated logic*, where $\mathcal{B}_L$ is a Herbrand base. The satisfaction is defined along the complexity of formulas as in the classical logic. An annotated atom $A : \mu$ is true in an interpretation f ($f \models_{GAP} A : \mu$), i.e. f is a model of $A : \mu$ iff $f(A) \geq \mu$.

$$\begin{gathered} f \models_{GAP} A : \rho(\mu_1, \ldots, \mu_k) \leftarrow B_1 : \mu_1 \wedge \ldots \wedge B_k : \mu_k \\ \text{if for all assignments } e \text{ of annotation variables } \mu_1, \ldots, \mu_k \text{ we have} \\ f(A) \geq_A \rho(e(\mu_1), \ldots, e(\mu_k)) \leftarrow f(B_1) \geq_{B_1} e(\mu_1) \wedge \ldots \wedge f(B_k) \geq_{B_k} e(\mu_k) \end{gathered} \tag{13}$$

Definition 10 (the learning from entailment setting of the IGAP task)**.** When learning from GAP entailment, a set of annotated examples E is given. The annotated background knowledge B is given. The task is to find an annotated hypothesis H, such that the following conditions hold:

$$\begin{aligned} (\forall e : \alpha \in E) H \wedge B &\models_{GAP} e : \alpha \quad \text{(gap-completeness of } H\text{)} \\ (\forall e : \alpha \in E)(\forall \beta > \alpha) H \wedge B &\not\models_{GAP} e : \beta \quad \text{(gap-consistency of } H\text{).} \end{aligned} \tag{14}$$

The φ-GAP algorithm we present here is based on the multiple use of the ILP system ALEPH [37]. Practical illustration of the algorithm can be found in Example 5.

The φ-GAP algorithm

- *Input:* Annotated E and B, local preferences $p^u_{L_i}$ for $p_i \in B$.
- *Output:* Annotated H

---

1. Initialize the two-valued (crisp) hypothesis $H^c = \emptyset$.
2. Find out
   - every annotations α, present in E ($\alpha_1 < \ldots < \alpha_n$)
   - every $m_1, \ldots, m_k$ classes of annotations for predicates $p_1, \ldots, p_k \in B$ $(\beta_{p_{11}}, \ldots, \beta_{p_{1m_1}}, \ldots, \beta_{p_{k1}}, \ldots, \beta_{p_{km_k}})$.
3. Transform the annotated background knowledge B to a crisp background knowledge B^c by an additional attribute for the annotation, i.e. $p_i(x_1, \ldots, x_{i_s}) : \beta_{p_{ij}} \dashrightarrow p^c_i(x_1, \ldots, x_{i_s}, \beta_{p_{ij}})$, where $j \in \{1, \ldots, m_i\}$.
4. For every predicate $p^c_i \in B^c$ corresponding to $p_i \in B, i \in \{1, \ldots, k\}$, add ordering-completion axioms to B^c which consist of
   - ordering-completion axioms: $p^c_i(x_1, \ldots, x_{i_s}, X) \leftarrow le_{p_i}(X, Y), p^c_i(x_1, \ldots, x_{i_s}, Y)$
   - attribute-value ordering predicates: $le_{p_i}(\beta_{p_{iv}}, \beta_{p_{iw}})$ if $p^u_{L_i}(\beta_{p_{iv}}) \leq p^u_{L_i}(\beta_{p_{iw}})$ for $v, w \in \{1, \ldots, m_i\}, v \neq w$.

5. For all α_i, where $1 < i \leq n$ do the following:
   - split the example set E to negative $E_i^{c-} = \{e^c(x_1, \ldots, x_t, \alpha_i))|e(x_1, \ldots, x_t) : \gamma \in E \text{ and } \gamma < \alpha_i\}$ and positive $E_i^{c+} = \{e^c(x_1, \ldots, x_t, \alpha_i))|e(x_1, \ldots, x_t) : \gamma \in E \text{ and } \gamma \geq \alpha_i\}$ parts
   - compute with ALEPH the hypothesis H_i^c for the two-valued background knowledge B^c, positive E_i^{c+} and negative E_i^{c-} example sets
   - add the hypothesis H_i^c to H^c.
6. Transform the crisp hypothesis H^c to annotated hypothesis H by transforming the additional attributes in literals back, i.e.
   $p_i^c(x_1, \ldots, x_{i_s}, \beta_{p_{i_j}}) \dashrightarrow p_i(x_1, \ldots, x_{i_s}) : \beta_{p_{i_j}}$, where $j \in \{1, \ldots, m_i\}$
   $e^c(x_1, \ldots, x_t, \alpha_i) \dashrightarrow e(x_1, \ldots, x_t) : \alpha_i$, where $1 < i \leq n$

Example 5. We demonstrate the work of the φ-GAP algorithm on the illustrative example in Table 1. First, we have to define inputs[21]:

- E = { `eval(apple):1,eval(danube):2,...,eval(tulip):3` }
- B = { `di(apple):100, pr(apple):99, eq(apple):[],...`
  `...,di(tulip):800, pr(tulip):45, eq(tulip):[tv,int]` }
- we use local preferences for numerical attributes *di* and *pr* computed in Example 4. For the non-numerical attribute *eq* local preferences from the Example 3 are used.

At the first step of the algorithm, we initialize H^c = {}. In step 2, we find the following annotations:

- $\alpha_1 = 1 < \alpha_2 = 2 < \alpha_3 = 3$
- there are three classes $m_{di} = 7, m_{pr} = 7, m_{eq} = 4$ of annotations with the following values:
  $\beta_{p_{di_1}} = 100, \beta_{p_{di_2}} = 300, \ldots, \beta_{p_{di_{m_{di}-1}}} = 1200, \beta_{p_{di_{m_{di}}}} = 1300$
  $\beta_{p_{pr_1}} = 35, \beta_{p_{pr_2}} = 40, \ldots, \beta_{p_{di_{m_{pr}-1}}} = 120, \beta_{p_{di_{m_{pr}}}} = 149$
  $\beta_{p_{eq_1}} = [], \beta_{p_{eq_2}} = [tv], \beta_{p_{eq_{m_{eq}-1}}} = [int], \beta_{p_{eq_{m_{eq}}}} = [tv, int]$.

In the following step (3), the transformation of background knowledge is provided to the form of Prolog atoms with the following result:
B^c = {`di(apple, 100), pr(apple, 99), eq(apple, []), ...,`
`di(tulip, 800), pr(tulip, 45), eq(tulip, [tv, int])`}.

[21] We use shortest notations for attributes, i.e. *eval* for user's evaluation, *na* for hotel name, *di* for the distance, *pr* for price and *eq* for equipment. Similarly, the grades of user's evaluation are expressed by numbers 1 (poor), 2 (good) and 3 (excellent). Since Prolog perceives capitals as variables, we use lower-case notation, e.g. "apple" instead of "Apple".

As the next step, we add the following axioms and predicates to B^c:

- ordering-completion axioms:
  `di(A, X) :- ledi(X, Y), di(A, Y)`
  `pr(A, X) :- lepr(X, Y), pr(A, Y)`
  `eq(A, X) :- leeq(X, Y), eq(A, Y)`
- attribute-value ordering predicates:
  `le_di(100, 300), le_di(300, 500), le_di(500, 800),`
  `le_di(800, 1100), le_di(1100, 1200), le_di(1200, 1300)`
  `le_pr(149, 120), le_pr(120, 99), le_pr(99, 60), le_pr(60, 45),`
  `le_pr(45, 40), le_pr(40, 35).`
  `le_eq([], [tv]), le_eq([], [int]), le_eq([tv], [tv, int]),`
  `le_eq([int], [tv, int])`

In step 5, we have two iterations, in this case, namely for $\alpha_2 = 2, \alpha_3 = 3$, with the following example sets and resulting hypotheses:
$E_2^{c+} = \{$`eval(danube, 2), eval(cherry, 2), eval(iris, 2),`
`eval(linden, 2), eval(oak, 2), eval(pear, 2), eval(poplar, 2),`
`eval(rose, 2), eval(spruce, 2), eval(tulip, 2)`$\}$
$E_2^{c-} = \{$`eval(apple, 2), eval(lemon, 2), eval(rhine, 2),`
`eval(themse, 2)`$\}$
$H_2^c = \{$`eval(A, 2) :- di(A, 300), eq(A, [tv]);`
`eval(A, 2) :- pr(A, 120), eq(A, [int])`$\}$
or
$E_3^{c+} = \{$`eval(iris, 3), eval(linden, 3), eval(rose, 3), eval(tulip, 3)`$\}$
$E_3^{c-} = \{$`eval(apple, 3), eval(danube, 3), eval(cherry, 3),`
`eval(lemon, 3), eval(oak, 3), eval(pear, 3), eval(poplar, 3),`
`eval(rhine, 3), eval(spruce, 3), eval(themse, 3)`$\}$
$H_3^c = \{$`eval(A, 3) :- di(A, 500), pr(A, 99), eq(A, [tv, int])`$\}$

As the last step, we transform the crisp hypothesis $H^c = H_3^c \cup H_2^c$. Thus, the final result is:
$H = \{$ `eval(A):2 :- di(A):300, eq(A):[tv];`
`eval(A):2 :- pr(A):120, eq(A):[int];`
`eval(A):3 :- di(A:500, pr(A):99, eq(A):[tv, int]` $\}$

According to the computed local preferences, we can express these results as:

- IF distance$\geq$300 AND equipment=$\{$tv$\}$ THEN user's evaluation ≥ 2
- IF price$\leq$120 AND equipment=$\{$int$\}$ THEN user's evaluation ≥ 2
- IF distance$\geq$500 AND price$\leq$99 AND equipment=$\{$tv, int$\}$
  THEN user's evaluation ≥ 3

Note that we do not normalize the annotations to a unit interval $[0, 1]$, since we want to make the computation simpler.

Now, note about the *training accuracy*, mentioned in Section 3. It concerns the quality of learned hypothesis. As can be seen in the φ-GAP algorithm, in case of n

grades of user's evaluation we have $H_2^c, \ldots, H_n^c$ hypotheses, whose qualities can be expressed by several measures. One measure can be the standard accuracy measure for hypotheses, used in ALEPH [37]:

$$\varphi - \text{accuracy}(H_i^c) = \frac{\text{cov}(H_i^c, E_i^{c+}) + (|E_i^{c-}| - \text{cov}(H_i^c, E_i^{c-}))}{|E_i^{c+} + E_i^{c-}|} \tag{15}$$

where $\text{cov}(H_i^c, E_i^{c+})$ and $\text{cov}(H_i^c, E_i^{c+})$ are the number of covered examples from E_i^{c+} and E_i^{c-}, respectively by the hypothesis H_i^c.

Since we do not allow the hypothesis to cover negative examples, the more convenient measure for training accuracy is the ratio of covered positive examples to all positive examples:

$$\varphi - \text{coverage}(H_i^c) = \frac{\text{cov}(H_i^c, E_i^{c+})}{|E_i^{c+}|}. \tag{16}$$

Note that the complexity of an ILP system is quite great[22] since it deals with joins of relations (and several substitutions).

In our case, the data in object-attribute representation (i.e. Table 1) are transformed to RDF-triples *Object-Attribute-Value*, represented by a predicate $attribute(object, value)$. Thus, all predicates are joined via an object identificator and substitutions are considered just for attribute values[23].

Moreover, user's preferences are usually learned from a small dataset[24] like our illustrative example (Table 1), so the learning process is relatively fast. Thus, using the φ-GAP algorithm is convenient for on-line user preference learning (see the following chapter about experiments).

Le us have a look, why the IGAP model (and the respective φ-GAP algorithm) is convenient for user preference learning. We do this by considering the defined fundamental and special requirements (Definitions 2 and 1):

- Since IGAP is a generalization of ILP (proved in [26]), we are able to represent an arbitrary combination of *ordinal* and *nominal* attributes. Moreover, in ILP it is easy to deal with *several data models* (Object-Attribute, Multi-Relational, RDF).
- The truth values in the quantitative part of GAP can be used to represent *imperfection* (uncertainty, vagueness, imprecision, ...) and the notions of a *natural language* (e.g. vague concepts, such as "cheap", "near"). Moreover, by truth values the *orderings of attribute domains* can be represented.
- Considering ρ in GAP rules as aggregation function @, *ordinal ratings* (classification) of objects can be represented. Since IGAP is an inductive model, we are capable to learn these ratings.

[22] It is well-known fact in data mining.

[23] All these settings can be determined in the ALEPH background knowledge.

[24] Note that rating a big number of objects is unusual for users.

Thus, we can conclude that IGAP (or the φ-GAP learning method) cover all the fundamental and special requirements for a flexible user preference learning model (Definitions 1 and 2).

6 EXPERIMENTS

The φ-GAP method was tested in two ways: as a standalone data-mining algorithm and as an integrated tool to a so-called recommender system, developed during the NAZOU project [34]. Since the aim was to check the precision of our models, we concentrated mainly to accuracy measures.

First, φ-GAP was tested on the real dataset of 206 Slovak companies, which had to submit their preferences (ratings from 1 to 7) of business competitiveness and information systems usage, where the aim was to learn the impact of information systems on nine processes identified in the companies. The results are discussed in [23], where the φ-GAP was compared with a linear regression model. To sum up, φ-GAP on the given data yields at least three times better results in every dimension of competitiveness than regression. It could be expected that large homogenous sets of data will be in most cases better explained by φ-GAP than by regression. Note that time complexity was not measured (it is evident that regression is faster than φ-GAP).

The second experiment concerned the well-known *auto-mpg* dataset from the UCI machine learning repository [41]. The above mentioned UCI data describes properties of 398 cars, concerning city-cycle fuel consumption. There are 9 attributes, 2 nominal, and 7 ordinal, from which the fuel consumption in miles per gallon (mpg) was considered as the target attribute, on which the user preferences were based. The dataset contains missing attribute values. Note that the more miles per gallon the car runs the better for the user. Thus, this attribute was discretized to 5 classes of preferences (5 – the best fuel consumption, 1 – the worst). The discretisation was made equidistantly, i.e. the domain of the attribute mpg was divided into 5 parts of the same size. We tested the order preserving accuracy (see Equation (5)) of φ-GAP by the mentioned method of proportion of correctly classified pairs of objects to all comparable pairs of objects. The accuracy of φ-GAP was over 97 %, so, at most 3 % of all comparable pairs was classified badly. This experiment was discussed in [24]. Other characteristics of this experiment – as the $\varphi-$coverage, number of rules and time of the computing are presented in Table 3.

i	5	4	3	2
$\varphi-$coverage(H_i^c)	0.22	0.58	0.78	0.85
time of computing H_i^c (sec)	32	106	209	364
number of rules in H_i^c	2	8	12	5

Table 3. Characteristics of the experiment of φ-GAP on the auto-mpg dataset according to the hypotheses H_i^c

During the NAZOU project [34], φ-GAP was integrated to a chain of tools forming a type of a recommender system [20]. φ-GAP was used for an Iterative User preference learning task (see Definition 9). An experiment with 109 sessions with real users was performed, from which in each session users made at least two iterations of the recommendation process. The accuracy of φ-GAP was measured in two ways:

- τ-iterative correlation$_i$ was computed for each iteration i of the recommendation process (see Equation (6)). Thus the sequence τ-iterative correlation$_0$, τ-iterative $correlation_1, \ldots,$ τ-iterative correlation$_k$ of accuracies was obtained for each session. About 60 % of these sequences were non-decreasing, 20 % were non-increasing, and 20 % were non-monotone (thus, in one iteration the accuracy grows, in the following iteration it falls, or conversely). We deduced that in the dominant part of the sessions, the user's preferences computed by φ-GAP approximate to the real preferences of users, i.e. in most cases, φ-GAP was helpful in recommendation;
- average iterative rating$_i$ was computed for each iteration i of the recommendation process (see Equation (7)). Thus the sequence of average evaluations average iterative rating$_0$, average iterative rating$_1, \ldots,$ average iterative rating$_k$ of objects was obtained for each session. The results were similar to the previous ones: about 60 % of these sequences was non-decreasing, while 20 % was non-increasing and 20 % it was non-monotone. These results prove the previous consequences, i.e. that in most cases φ-GAP was helpful in the recommendation.

Since users rated just a few objects (as usual in real life), the computations on these small datasets were fast (within a second). Such computing times make φ-GAP approppriate for on-line user preference learning.

7 COMPARISON TO OTHERS

There are several approaches and models of user preference learning in recommender systems [2, 10, 31, 36]:

Term extraction and text categorization methods are used in [42, 39, 35] for classification of documents to user profiles according to words they contain. It is often hard to precisely get the attributes of objects (e.g. the semantics of data) in case of unstructured data repersentation. Thus, textual representation of objects is not considered further in this work.

Statistical approaches [28, 29, 43, 3] usually assume attribute independence or require assumptions on probability (usually normal)distribution. These assumptions are very strong and not general in real situations.

Clustering algorithms are used in [16, 11, 38]. In [8, 6], preferences are represented by graphs. Note that these approaches deal just with correlations between objects, not considering the attributes of objects. Thus, these models are not good to express more complex user preferences (for example in form of rules).

Support Vector Machines and k-Nearest Neighbour classification techniques are used to compute user preferences in [12, 18]. These techniques are more approppriate in case of homogenenous (mainly numerical) data but this is not the general case in real applications.

In [17], Knowledge-Based Artificial Neural Networks are used for preference elicitation, with the ability to encode assumptions concerning preferential independence and monotonicity. The use of pairwise preferences is assumed in this work. E.g., consider a rule $x_i < x_j \wedge y_i \geq y_j \longrightarrow o_i \succ o_j$, where $o_i \succ o_j$ indicates that the object o_i is preferred to the object o_j. This rule expresses that if an object o_i has smaller value in the attribute x and greater or equal value in the attribute y than object o_j, then o_i is preferable to o_j for the user.

In [19], an evolutionary approach is used. In this model, a user's utility function U applied to a product (object) p with n attributes is defined as $U(p) = \sum_{i=1}^{n} w_i f_i(x_i)$, where $w_i \in \mathbb{R}$ denotes the weight (importance) of an attribute a_i and $f_i : Domain(a_i) \to \mathbb{R}$ denotes the "attribute utility" function[25] of an attribute a_i, and x_i is the value of an attribute a_i of the product p. Except the complete utility function, the proposed algorithm learns both the attribute weight and attribute utility function.

A language called DD-PREF is introduced in [27]. Preferences are learned over sets of objects, where the learning method takes as input one or more sets of objects that have been identified by a user as desirable[26]. In DD-PREF preferences are represented as tuples $P = \langle \overrightarrow{q}, \overrightarrow{d}, \overrightarrow{w}, \alpha \rangle$, where $q_f : V_f \to [0, 1]$ is the desired "depth"[27] (preferred feature values), $d_f \in [0, 1]$ is the desired "diversity" (preferred distribution of values across the desired range), $w_f \in [0, 1]$ is the feature preference "weight" for the feature f with values in the set V_f and $\alpha \in [0, 1]$ specifies the relative importance of diversity versus depth across all features. q_f, d_f and w_f are estimated using probability methods.

The language of first-order logic is used to represent user preferences in [32]. In [9] preferences are learned using an Inductive Logic Programming system TILDE [5]. ILP systems are able to learn from multiple-relations. Due to their expressiveness, input and output of ILP systems are readable, in contrast to sub-symbolic systems like neural networks.

The presented rules in [17] allow to define complex preferences but deal with pairwise preferences (rankings[28], instead of ratings). Moreover, these consider only numerical data. The attribute utility function in [19] allows to define some "preferences" for both numerical and non-numerical data. The drawback of this model is that the preferences can be expressed strictly as a weighted sum. The model presented in [27] seems to be more flexible, however, the α parameter is a-priori

[25] For example, $f_{price}(x) = 1 - x$

[26] The idea is that it is often easier for users to express their preferences by a set of preferable objects (for example, a set of favourite songs).

[27] The depth and the attribute utility function in [19] have the same meaning.

[28] As stated before, this paper deals with rating.

given. Moreover, the preferences are learned not from ratings but from a set of preferable objects. Except [32, 9], none of the approaches presented considers multi-relational data. Even if ILP systems are able to learn from multiple relations, the approaches presented in [32, 9] do not consider imperfection nor ordinal ratings, or orderings of attribute domains.

As can be seen, unlike IGAP none of the other approaches to preference learning in (content-based) recommender systems cover all fundamental and special requirements introduced in Definitions 2 and 1.

Comparing IGAP to other models in [26, 25] we find out that IGAP can be viewed as a "generalization" of other fuzzy inductive logic programming and ordinal classification models.

8 CONCLUSIONS

This paper is focused to a formal model of user preference learning for content-based recommender systems.

Fundamental and special requirements to user preference learning were identified.

Three learning tasks were introduced as the exact, the order preserving and the iterative user preference learning tasks. The first two tasks concerned the situation where we have the user's rating available for a large part of objects. The third task does not require any prior knowledge about the user's ratings history. Local and global preferences were distinguished in the presented model.

Two methods for local preference learning were proposed. The model of induction of generalized annotated programs (IGAP) was described. The φ-GAP algorithm was presented as the learning method for IGAP. All these methods were illustrated in the example. Note that we do not know about any other inductive GAP model.

φ-GAP was tested as a standalone data mining application and as an integrated tool to a recommender system. IGAP was compared to other approaches to user preference learning. The comparison of IGAP to other models of inductive fuzzy logic programming and ordinal classification was made in [26]. The experiments and the comparison to other approaches shows promising results.

The research showed that, however, a flexible model has to consider multi-relational data (in case of some background knowledge about objects, users and the used domain), in most cases data are represented in one table. Thus, a propositional (i.e. one table on input) version of φ-GAP, called π-GAP was developed. This algorithm is now in the testing phase, the results seem promising – it is considerably faster than the φ-GAP.

In the future, an integration of π-GAP to a recommender system and its experimental verification is planned, especially for the domain of used cars.

Acknowledgement

This work was supported by the NAZOU project [34].

REFERENCES

[1] Abdi, H.: The Kendall Rank Correlation Coefficient. In N. J. Salkind (Ed.): Encyclopedia of Measurement and Statistics, Sage, Thousand Oaks (CA), 2007.

[2] Adomavicius, G.—Tuzhilin, A.: Using Data Mining Methods to Build Customer Profiles. Computer, Vol. 34, 2001, No. 2, pp. 74–82.

[3] Ardissono, L.—Gena, C.—Torasso, P.—Bellifemine, F.—Chiarotto, A.—Difino, A.—Negro, B.: Personalized Recommendation of TV Programs. In: Proceedings of the 8th AI*IA Conference, Pisa, 2003, Springer, LNAI, Vol. 2829, ISBN 978-3-540-20119-9, pp. 474–486.

[4] Balabanovic, M.—Shoham, Y.: Fab: Content-Based, Collaborative Recommendation. Communications of the ACM archive, Vol. 40, Issue 3, 1997, ISSN 0001-0782, pp. 66–72.

[5] Blockeel, H.—de Raedt, L.—Jacobs, N.—Demoen, B.: Scaling up Inductive Logic Programming by Learning from Interpretations. Data Mining and Knowledge Discovery, Vol. 3, 1999, No. 1, ISSN 1384-5810, pp. 59–93.

[6] Branting, L. K.—Broos, P. S.: Automated Acquisition of User Preferences. International Journal of Human-Computer Studies, Vol. 46, 1997, pp. 55–77.

[7] Burke, R.: Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, Vol. 12, 2002, No. 4, ISSN 0924-1868, pp. 331–370.

[8] Crane, M.: Efficiently Learning Trends in User Preferences. Poster at Computer Science 294, Practical Machine Learning, Fall 2006.

[9] Dastani, M.—Jacobs, N.—Jonker, C. M.—Treur, J.: Modelling User Preferences and Mediating Agents in Electronic Commerce. Lecture Notes in Computer Science, Vol. 1991, ISBN 3-540-41671-4, pp. 163–193, Springer 2001.

[10] Delgado, J.—Ishii, N.: Formal Models for Learning of User Preferences, a Preliminary Report. In: Proceedings of the IJCA 99 Workshop on Learning about Users, Stockhpm 1999, pp. 13–20.

[11] Delgado, J.—Ishii, N.: On-line Learning of User Preferences in Recommender Systems. In: Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering, 1999.

[12] Deng, L.—Chai, X.—Tan, Q.—Ng, W.—Lee, D. L.: Spying Out Real User Preferences for Metasearch Engine Personalization. In: Proceedings of the 6th WEBKDD Workshop, Seattle, 2004.

[13] Džeroski, S.—Lavrač, N.: An Introduction to Inductive Logic Programming. Relational data mining, Springer, 2001, ISBN 3-540-42289-7, pp. 48–73.

[14] Eckhardt, A.—Horváth, T.—Vojtáš, P.: PHASES: A User Profile Learning Approach for Web Search. In: Proceedings of the 2007 IEEE/WIC/ACM Interna-

tional Conference on Web Intelligence (WI 2007), Silicon Valley, USA, IEEE Computer Society, 2007, ISBN 0-7695-3026-5, pp. 780–783.

[15] Eckhardt, A.—Horváth, T.—Vojtáš, P.: Learning Different User Profile Annotated Rules for Fuzzy Preference Top-k Querying. In: Proceedings of the 1st International Conference on Scalable Uncertainty Management, (SUM '07), Washington DC, USA, Lecture Notes in Artificial Intelligence, Vol. 4772, 2007, ISSN 0302-9743, ISBN 978-3-540-75407-7, pp. 116–130, Springer, 2007.

[16] Goerge, T.—Merugu, S.: A Scalable Collaborative Filtering Framework Based on Co-clustering. In: Proceedings of the Fifth IEEE International Conference on Data Mining, 2005, ISBN 1550-4786, pp. 625–628.

[17] Geisler, B.—Ha, V.—Haddawy, V.: Modeling User Preferences via Theory Refinement. In: Proceedings of 6th International Conference on Intelligent User Interfaces (IUI), Santa Fe, New Mexico, ACM, 2001, ISBN 1-58113-325-1, pp. 87–90.

[18] Grčar, M.—Fortuna, B.—Mladenič, D.—Grobelnik, M.: kNN Versus SVM in the Collaborative Filtering Framework. Data Science and Classification, ISBN 978-3-540-34415-5, pp. 251–260, Springer, 2006.

[19] Guo, Y.—Műller, J. P.—Weinhardt, C.: Learning User Preferences for Multi-attribute Negotiation: An Evolutionary Approach. In: Proceedings of CEEMAS 2003, LNAI 2691, pp. 303–313, Springer, 2003.

[20] Gurský, M.—Horváth, T.—Novotný, R.—Vaneková, V.—Vojtáš, P.: UPRE: User Preference Based Search System. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI '06), Hong Kong, 2006, IEEE Computer Society, 2006, ISBN 0-7695-2747-7, pp. 841–844.

[21] Gurský, P.—Horváth, T.: Dynamic Search of Relevant Information. In: 4th Conference Znalosti '05, Stará Lesná, Slovakia, 2005: FEI VB-TU Ostrava, Czech Republic, 2005, ISBN 80-248-0755-6, pp. 194–201.

[22] Hawkins, R.: Ranking and Scoring – Guidelines. Manual, ICRA Learning Resources.

[23] Horváth, T.—Sudzina, F.—Vojtáš, P.: Mining Rules from Monotone Classification Measuring Impact of Information Systems on Business Competitiveness. In: 6th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services, Vienna, Austria, 2004, Springer, IFIP International Federation For Information Processing (Vol. 159), 2004, ISSN 1571-5736, ISBN 0-387-22828-4, pp. 451–458.

[24] Horváth, T.—Vojtáš, P.: GAP – Rule Discovery for Graded Classification. In: Workshop of Advances in Inductive Rule Learning (W8) of the 15th European Conference on Machine Learning and the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD '04), Pisa, Italy, 2004, TU Darmstadt (J. Fuernkranz, Ed.), Darmstadt, Germany, 2004, pp. 46–63.

[25] Horváth, T.—Vojtáš, P.: Ordinal Classification with Monotonicity Constraints. In: Proceedings of the 6th Industrial Conference on Data Mining (ICDM '06), Leipzig, Germany, 2006, Springer-Verlag, Lecture Notes in Artificial Intelligence (Vol. 4065), 2006, ISSN 0302-9743, ISBN 3-540-36036-0, pp. 217–225.

[26] Horváth, P.—Vojtáš, P.: Induction of Fuzzy and Annotated Logic Programs. In: Proceedings of the the 16th International Conference on Inductive Logic Pro-

gramming (ILP '06), Santiago de Compostela, Spain, 2006, Springer-Verlag, Lecture Notes in Artificial Intelligence, Vol. 4455, 2007, ISSN 0302-9743, ISBN 978-3-540-73846-6, pp. 260–274.

[27] des Jardins, M.—Eaton, E.—Wagstaff, K. L.: Learning User Preferences for Sets of Objects. In: Proceedings of the 23rd International Conference on Machine Learning, ISBN 1-59593-383, Pittsburgh, PA 2006, pp. 273–280.

[28] Jung, S. Y.—Hong, J. H.—Kim, T. S.: A Statistical Model for User Preference. IEEE Transactions on Knowledge and Data Engineering, Vol. 17, 2005, No. 6, pp. 834–843.

[29] Kang, S.—Lim, J.—Kim, M.: Statistical Inference Method of User Preference on Broadcasting Content. In: Proceedings of ICCS 2005, Springer, LNCS 3514, 2005, pp. 971–978.

[30] Kifer, M.—Subrahmanian, V.S.: Theory of Generalized Annotated Logic Programming and Its Applications. J. Logic Programming, Vol. 12, 1992, pp. 335–367.

[31] Kobsa, A.: Generic User Modeling Systems. The adaptive web: Methods and strategies of web personalization, Heidelberg, Germany, Springer, LNCS 4321, ISBN 978-3-540-72078-2, pp. 136–154.

[32] Leite, J.—Babini, M.: Dynamic Knowledge Based User Modeling for Recommender Systems. In: Proceedings of the ECAI 2006 Workshop on Recommender Systems, Riva del Garda, Italy, 2006, pp. 134–138.

[33] Meteren, R. V.—Someren, M. V.: Using Content-Based Filtering for Recommendation. MLnet/ECML 2000 Workshop, May 2000, Barcelona, Spain.

[34] The NAZOU project. `http://nazou.fiit.stuba.sk`.

[35] Pazzani, M.—Billsus, D.: Learning and Revising User Profiles: The Identification of InterestingWeb Sites. Machine Learning, Vol. 27, 1997, pp. 313–331.

[36] Al Rashid, M.—Albert, I.—Cosley, D.—Lam, S. K.—McNee, S. M.—Konstan, J. A.—Riedl, J.: Getting to Know You: Learning New User Preferences in Recommender Systems. In: Proceedings of the 7th International Conference on Intelligent User Interfaces IUI '02, ISBN 1-58113-459-2, pp. 127–134.

[37] Srinavasan, A.: The Aleph Manual. Technical Report, Comp. Lab., Oxford University.

[38] Suryavanshi, B. S.—Shiri, N.—Mudur, S. P.: A Fuzzy Hybrid Collaborative Filtering Technique for Web Personalization. In: Proceedings of 3rd Workshop on Intelligent Techniques for Web Personalization, in conjunction with IJCAI '05, Edinburgh, Scotland, 2005.

[39] Taschuk, M.: A Hybrid Knowledge-based/Content-based Recommender System in the Bluejay Genome Browser. Undergraduate Honours Thesis, Faculty of Medicine at the University of Calgary, 2007.

[40] Towle, B.—Quinn, C.: Knowledge Based Recommender Systems Using Explicit User Models. Knowledge-based Electronics Markets, papers from the AAAI Workshop, AAAI Technical Report WS-00-04, AAAI Press, pp. 74–77.

[41] Asuncion, A.—Newman, D. J.: UCI Machine Learning Repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Irvine, CA, 2007, University of California, School of Information and Computer Science.

[42] Xu, J. A.—Araki, K.: A Personalized Recommendation System for Electronic Program Guide. In: Proceedings of AI 2005, Springer, LNAI 3809, 2005, pp. 1146–1149.

[43] Yu, Z.–Zhou, X.—Yang, Z: A Hybrid Learning Approach for TV Program Personalization. In: Proceedings of KES 2004, Springer, LNAI 3213, 2004, pp. 630–636.

[44] Vojtáš, P.: Fuzzy Logic Programming. Fuzzy Sets and Systems, Vol. 124, 2004, No. 3, pp. 361–370.

[45] Vojtáš, P.—Vomlelová, M.: Transformation of Deductive and Inductive Tasks Between Models of Logic Programming With Imperfect Information. In: Proceedings of IPMU, 2004, Roma, Italy, pp. 839–846.

[46] Vojtáš, P.—Vomlelová, M.: On Models of Comparison of Multiple Monotone Classifications. In: Proc. IPMU 2006, Paris, France, Editions EDK, Paris, pp. 1236–1243.

Tomáš Horváth has finished his Ph. D. study at Pavol Jozef Šafárik University in Košice, Slovakia in 2008. He works in the area of fuzzy relational data mining, user preference learning and recommender systems. Now, he is a research assistant at Institute of Computer Science of Pavol Jozef Šafárik University.