Heterogeneous Attribute Utility Model: A new approach for modeling user profiles for recommendation systems

Vincent Schickel-Zuber, Boi Faltings
School of Computer and Communication Sciences – IC
Swiss Federal Institute of Technology (EPFL)
Lausanne, Switzerland
{vincent.schikel-zuber, boi.faltings}@epfl.ch

Abstract
With the growing size of product catalogs, the task of finding an object has become analogous to finding a needle in a haystack. Recommendation systems are tools that help people in this process. There are two types of techniques for implementing such systems. Collaborative filtering uses statistical properties of the buying behavior of other users to recommend items that the user is likely to buy. Preference-based techniques construct an explicit profile of an individual user's preferences, and find the best matches with that model. To make accurate recommendations, both methods currently require more data about a customer than is usually available: the preference-based method requires the user to specify preferences on the product attributes, while collaborative filtering requires that a significant number of items have been bought or ranked to determine the user's type with sufficient significance.
We conjecture that the weaknesses are due to a lack of inductive bias in the learning methods used to build the prediction models. We propose a new method, heterogeneous attribute utility model (HAUM), where the structure of user preferences are assumed to follow an ontology of product attributes. Using the data of the MovieLens system, we show that experimentally real user preferences indeed closely follow an ontology based on movie attributes, and that a recommender based just on a single individual's preferences and this ontology performs better than collaborative filtering, with the greatest differences when little data about the user is available. This points the way to how proper inductive bias can be used for significantly more powerful recommender systems in the future.

Key Words: Ontology, Utility Theory, User Model, Recommendation Systems.

1 Introduction
Consider a situation where you find yourself with an evening alone and would like to rent a DVD to watch. There are hundreds of movies to choose from. For several reasons, this is a difficult problem. First, most people have limited knowledge about the alternatives. Second, the set of alternatives changes frequently. Third, this is an example of a low user involvement decision process where the user is not prepared to spend hours expressing his preferences. Recommender systems have been devised as tools to help people in such situations. Two kinds of techniques are widely used in e-commerce sites today.

The first technique is item-to-item collaborative filtering (CF, [9]) which recommends products to users based on other users' experience. Amazon.com[1], with over 29 millions customers and several million catalog items [9], uses this technique which is more commonly known to end-user as "Customers who bought this item also bought these items:". Collaborative filtering generates recommendations based on the experience of like-minded groups of users, based on the assumption that similar users like similar objects. Therefore, CF's ability to recommend items depends on the ability to successfully identify the set of similar users, known as the target user's neighborhood. CF does not build an explicit model of user preferences for individual product attributes. Instead, preferences remain implicit in the ratings that the user gives to some subset of products, either explicitly or by buying them. In practice, CF is the most popular recommendation technique and this is due to three main reasons. First, studies have shown it to have satisfactory performance when sufficient data is available. Second, it can compare items without modeling them and thus can theoretically deal with any kind of item as long as they have been rated by other people. Finally, the cognitive requirements on the user are very low. However, as argued by many authors [10][11][15][17], CF suffers from profound problems such as :

- *Sparsity*. This is CF's major problem and occurs when the number of items far exceeds what an individual can rate.

[1] http://www.amazon.com

- *Cold start*: When a new user enters the system, he has not yet rated a sufficient number of items for CF to correctly locate the right neighborhood.
- *Latency* or *new item problem:* Product catalogs evolve over time; however, the collaborative approach cannot deal with new products as they have not been previously rated.
- *Scalability.* The computation of the neighborhood requires looking at all the items and users in the systems. Consequently, as the number of users grows, so does the complexity.
- *Privacy*. Users are becoming more and more reluctant to express their preferences. Currently, the similarity matrix is located on a server and is thus accessible to a third party.
- *Shilling attacks*: malicious users can alter user ratings in order to influence the recommendations.

The other widely used technique is preference-based recommendation. Here, a user is asked to express explicit preferences for certain attributes of the product. If preferences are accurately stated, multi-attribute utility theory (MAUT, [8]) provides methods to find the most preferred product even when the set of alternatives is extremely large and/or volatile, and thus has no problems of sparsity, cold starts, latency or scalability. Furthermore, since recommendations are based only on the individual user's data, there are no problems with privacy or shilling. However, the big drawback of preference-based methods is that the user needs to express a potentially quite complex preference model. This may require a large number of interactions, and places a higher cognitive load on the user since he has to reason about the attributes that model the product.

However, attribute-based preference models can also be learned from user choices or ratings, just as in collaborative filtering. In our experiments, this by itself can already result in recommendations that are almost as good as those of collaborative filtering. The main novelty of this paper, however, is to use an ontology of product attributes to provide an inductive bias that allows learning of this individual preference model to succeed even with very few ratings. This leads us to a technique for recommender systems that outperform the best known collaborative filtering techniques on the MovieLens data that we have been using for our experiments. Furthermore, very few ratings, just 5 movies, suffice to get recommendations that are almost as good as what can be reached with many, 30 or more, ratings. At the same time, the user effort required by this technique is not significantly different from that in a collaborative filtering system. Thus, we can effectively get the best of both techniques.

This paper is organized as follows: Section 2 provides fundamental background in Collaborative Filtering, Multi—Attribute Utility Theory, and Ontology Reasoning while our novel approach with its algorithm is explained in Section 3. Section 4 contains experimental results with comparison to existing techniques. Finally, Section 5 provides the conclusions.

2 Existing techniques

Our approach uses the cognitive simplicity of Collaborative Filtering whilst maintaining the advantages of the Multi-Attribute Utility Theory. This is achieved by employing the knowledge of ontology and reason over its concepts instead of the item's content itself. Before defining our model, we start by introducing fundamental background.

2.1 Collaborative filtering

In pure collaborative filtering systems, users state their preferences by rating a set of items which are then stored in a user-item matrix called *S*. This matrix contains all the users' profile where the rows represents the users $\mathbf{U} = \{u_1,\ldots,u_m\}$, the columns the set of items $\mathbf{I} = \{i_1,\ldots,i_q\}$, and $S_{k,l}$ the normalized rating assigned to item l by user k. Given the matrix S and a target user u_i, a user-based CF predicts the rating of one or more target items by looking at the rated items of the *k* nearest neighbors to the target users. On the other hand, an item-based CF algorithm looks at the *k* most similar items that have been co-rated by different users. Due to the complexity of the former approach, we will not consider it and focus on the latter.

The first step of item-based collaborative filtering is to compute the similarity between two co-rated items. Many similarity measures exist and can be used, but the most common one is the *cosine metric* which measures the angle between two vectors of size m. When 2 items are similar, then the angle between be small and consequently give a cosine value close to 1; and inversely 0 when items are totally different. Over the years, the cosine metric has been updated in order to take into account the variance in the ratings of each user. The *adjusted cosine similarity* (1) does

exactly that by subtracting to each user rating $S_{j,a}$, the user's average grade $\hat{S}_j$. Once all the similarities have been computed, CF estimates the normalized rating of an item *a* by selecting the *k* most similar items to the target item *a* and predicts its weight (2).

$$sim(i_a, i_b) = \frac{\sum_{j=1}^{m}((S_{j,a} - \overline{S_j}) \times (S_{j,b} - \overline{S_j}))}{\sqrt{\sum_{j=1}^{m}(S_{j,a} - \overline{S_j})^2 \times \sum_{j=1}^{m}(S_{j,b} - \overline{S_j})^2}} \quad (1) \qquad S_{i,a} = \frac{\sum_{j=1}^{k}(S_{i,j} \times sim(i_j, i_a))}{\sum_{j=1}^{k} sim(i_j, i_a)} \quad (2)$$

Unfortunately, as the number of items and/or users grow so will the data sparsity in the matrix. This is CF's fundamental problem and explains why numerous authors [10][11][17] have focused their work to try to overcome it. Data-mining [17] or Two-way Aspect Model [15] are now used to extract the item similarity knowledge by using association between the user's profile [17] and the object's content [15] in order to augment standard similarity matrix. To overcome the latency problem, the content of the object has also been used to try to predict its rating. This is achieved by filling up the similarity matrix [4] or simply by using a weighted combination [10] of the content and collaborative prediction.

2.2 Multi-attribute utility theory

Multi-attribute utility theory dates from the early 40s when Von Neumann et al [19] have laid the foundations and proved, under condition of four axioms, that preferences and attitudes towards risk can be adequately modeled by a *utility function, u*. In this paper, we do not consider uncertainty, therefore the utility function becomes equivalent to a value function but later we consider expected values of similarity.

Formally, each outcome (item) is defined by a set of n attributes $\mathbf{X} = \{X_1,\ldots,X_n\}$. Each X_i can take any value d_i from a domain $D_i = \{d_1,\ldots,d_k\}$. The Cartesian product $\mathbf{D} = D_1 x\ D_2 x\ldots\ x\ D_n$ forms the space of all the possible outcomes **O**. The user expresses his preferences by defining the utility function and weight of each attribute. The simplified form of the Von Neumann and Morgenstern [19] theorem states that if outcome x is considered better or equivalent than outcome y, then the utility function *u* must satisfy equation (3).

$$\forall x,y \in \mathbf{O},\ x \succsim y \Leftrightarrow u(x) \geq u(y) \quad (3)$$

Furthermore, if we assume Mutual Preference Independence (MPI, [8]), then the theorem of Additive Value Function (WADD, [8]) can be used, and we can define the utility V of an outcome o_i as the sum of the sub-utility functions v_i of outcome o_k on each attribute multiplied by its weight w_i. The outcome with highest overall evaluation value is chosen as the optimal solution.

$$V(o_k) = \sum_{i=1}^{n} w_i v_i(o_k) \quad (4)$$

Theoretically, and under the MPI hypothesis, this strategy can achieve 100% accuracy if all the parameters can be precisely elicited. Unfortunately, the elicitation of the parameters is expensive and various authors have tried to simplify this elicitation process in order to make it usable in real systems. Stolze et al. [16] for example, exploit the idea of a scoring tree where a user expresses his preferences by modifying an existing tree via the use of rules. Once the preferences have been elicited, the system translates the scoring tree into a MAUT additive value function and then searches in the catalog for the most suitable products. Incremental Utility Elicitation [5], IUE, is another approach that eases the elicitation by an incremental process that interleaves utility elicitation and filtering of the outcomes based on the elicited information. A major contribution in that domain is the work done by Ha et al [5] [6] where polyhedral cones and pair wise comparison are used to estimate the user's true weights. Also, [6] makes the assumption that the utility function has a multi-linear form, and that all the sub-utility functions are known. Regrettably, computing the cone is a hard problem that makes it unsuitable for real life scenarios. More recently, Blythe in [2] has simplified the process by assuming MAUT additive value functions and used a linear programming formulation and pair wise comparison of alternatives to estimate the user's true utility. Nevertheless, all of the mentioned approaches work only for a small number of attributes which are difficult to apply to real life scenarios where alternatives are modeled by many features.

2.3 Ontology reasoning

With the emergence of the Semantic Web, it has been widely accepted that ontologies can be used to model the world in which we live in. In its general form, an ontology is a lattice where a node represents a concept (i.e.: an

object of the world we want to model) whilst the edges between concepts correspond to theirs semantic relations.

One of the simplest forms of an ontology is the concept tree (CT); a graph where the topology is a tree with only is-a relations. The tree structure makes the reasoning computationally efficient and the modeling of the domain easy. Despite its simplicity, a CT can greatly enhance modeling of the domain and the filtering process. Bradley et al in [3] have successfully used a concept tree to model the job domain and shown through experimentation that it is very useful for personalization.

Concept similarity is the predominant form of ontology reasoning. Most technique use the distance between the concepts in a concept tree or similar graphical ontology to estimate their similarities; the smaller the distance between two concepts the more similar they are. In [3] for example, the distance was simply the number of edges between the concepts while Yang and al in [20] used the depth of the common ancestor. Resnik in [12] defined the similarity based on the information content shared by the concepts rather than its distance. Resnik's metric postulates higher similarity among rare concepts. While this makes sense for concepts themselves, there is no reason why this would also hold for preferences.

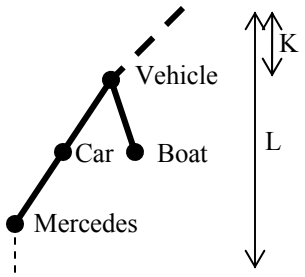

Figure 1: A CT on transports

The metrics in [3][12][20] assume that the similarity between two concepts is symmetric, i.e.: the similarity between concepts A and B is identical to the similarity between concepts B and A. However, in real life situations, the distance should be considered asymmetric. Consider for example the concept tree in Figure 1. If a user liked any kind of vehicle, then he will probably like Mercedes cars to a similar degree. On the other hand, liking Mercedes cars does not necessarily mean liking any vehicle as some people become sick on boats. Formally, and assuming that P(X) corresponds to the probability of X occurring [12] and $P(V \cap M)$ is equal to α, then the probability $P(V \mid M)$ is equal to $(L*\alpha)/(K+2)$ while $P(M \mid V)$ is equal to $(L* \alpha) / K$. This implies that $P(V \mid M) < P(M \mid V)$, which means that the similarity function is asymmetric.

Andreason and al in [1] defined an asymmetric metric based on the principle of upward reachable nodes. Their metric can be applied to any graph structure and differentiates between the cost of traveling upward or downward the ontology. To the best of our knowledge, this metric has not been used in practice due to the computational complexity. Nevertheless, three important properties that we use in our model are defined in [1]:

1. *Generalization cost property*. The cost of generalization should be significantly higher that the cost of specialization.
2. *Specificity cost property*. The cost of traversing edges should be lower when nodes are more specific.
3. *Specialization cost property*. Further specialization reduces similarity.

The generalization cost property models the asymmetry of the similarity function, which implies that the similarity function is not a metric .The specificity cost property represents the fact that sub-concepts are more meaningful to the user than super-concepts whilst the specialization property reflects the fact that the further away two concepts are, then the more dissimilar they become. As a consequence, the specificity property reduces the cost of traversing edges as we go deeper in the ontology and the specialization property increases the cost between two concepts if other concepts are found on the path between those concepts.

3 Heterogeneous attribute utility model

The Utility model defined in section 2.2 is a powerful strategy as long as we have a complete user model and all the outcomes are defined by the same set of attributes. As previously argued, the former condition is unrealistic in most situations and consequently makes preference elicitation the central problem. Furthermore, in volatile environments, products continuously change, yet the utility model can only compare outcomes if they have the same attribute. In this section, we introduce the *heterogeneous attribute utility model*, HAUM, which is capable to build an accurate user model from implicit preferences. HAUM relaxes the previous two conditions and uses two key components to evaluate the user's parameters:

- *Ontologies* to model and estimate utility values.
- *Multiple regression* to estimate the weights.

3.1 Basic idea

In recent years, the concept that ontologies could be used to represent knowledge has been developed [3] and accepted in the Semantic Web community. This paper goes further than existing work by using the ontology as a knowledge source in order to estimate missing user preferences. Our basic assumption is that an ontology can be used to model and estimate user preferences as long as a minimum of information about the user is known.

To illustrate this principle, take the following problem, where outcomes are defined by a set of 3 attributes X_1, X_2, and X_3. For our DVD example, X_1 could be the theme, X_2 the duration of the movie, and X_3 its MPPA rating. Figure 2 illustrates a possible representation of the domain as a concept tree, where each depth represents an attribute and the concept at depth i the possible domain values of the attribute i. Let outcome o_i be defined by the domain values $\{d_{1,1}; d_{2,1}; d_{3,1}\}$ and outcome o_j by $\{d_{1,1}; d_{2,2}; d_{3,2}\}$. Again, Figure 2 shows clearly that the 2 outcomes have some similarity as they both share the node $d_{1,1}$, which means that they both have this domain value. According to ontology theory, if two concepts are closely related in terms of distance in the graph, then we can assume that they are very similar in terms of meaning.

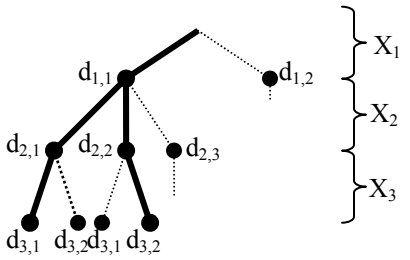

Figure 2: Representation of the Attributes

Following this, and ignoring the three properties in section 2.3, we can define a simple similarity function (5) between 2 outcomes o_i and o_j as the number of attributes they share over the total number of possible attributes; this normalization guarantees that the similarity value lies in the interval [0..1].

$$sim(o_i, o_j) = \frac{|\{x \mid x(o_i) = x(o_j)\}|}{|\{X\}|} \tag{5}$$

where $x(o_i)$ is the set of attributes defining outcome o_i, and $\{X\}$ is the set of all attributes. Furthermore, this equation assumes that each attribute is equally likely to contribute to whether one likes an outcome, and that each outcome is defined by the same number of attributes. In our example and using equation (5), we can deduce that outcomes o_i and o_j have a similarity equal to 1/3. Informally, this means that if the user liked outcome o_i, then there is one chance out of three that he will like outcome o_j. Notice that the additive utility model makes the assumption that if a user has liked features A and B, then he/she will like items containing A+B. This assumption is not made in probabilistic models, where each feature and its combination are considered independent. We argue that this inductive bias is realistic in most real life situations; and this is supported by our experimental results.

In volatile environments, outcomes do not always have the same number of attributes. For example, suppose that o_i is defined by the values $\{d_{1,1}; d_{2,1}\}$, o_k by $\{d_{1,1}; d_{2,2}\}$ and o_j by $\{d_{1,1}; d_{2,2}; d_{3,2}\}$. Consequently to equation (5), the similarity between o_i and o_k is equal to 1/3, which is also equal to the similarity between o_i and o_j. Furthermore, the similarity between o_i and o_j is equal to the similarity o_j and o_i. However, both situations are incorrect as the former violates the specialization property, while the latter violates the generalization cost property.

According to structural analysis, the similarity between two concepts is equal to one minus the distance between those concepts, where the distance is defined as a metric. Consider for example the simple generic graph in Figure 3; the distance between nodes n_i and n_j can be given as the distance between node n_i and its ancestor n_p plus the distance between node n_p and n_j. However, due to the generalization property, the distance from n_i to its ancestor should be higher than the distance from the ancestor to n_j. This property implies that the distance is in fact asymmetric and taking into account that the transitivity property of the distance, we can define the asymmetric distance between two nodes as:

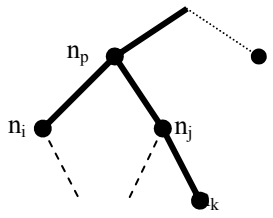

Figure 3: A simple tree

$$adist(n_i, n_j) = \rho \times dist(n_i, anc(n_i, n_j)) + (1-\rho) \times dist(anc(n_i, n_j), n_j) \tag{6}$$

Where ρ is a generalization coefficient that must be included in the interval (0.5, 1] to satisfy the generalization property, and $anc(n_i,n_j)$ is the closest common ancestor to node n_i and n_j. This coefficient implements the generalization property stated in section 2.3.

After substituting the distance by one minus the similarity and simplified equation (6), we are able to decompose the asymmetric similarity between two nodes n_i and n_j as:

$$asim(n_i, n_j) = \rho \times sim(n_i, anc(n_i, n_j)) + (1-\rho) \times sim(anc(n_i, n_j), n_j) \quad (7)$$

Furthermore, and by considering equation (5), we can define the similarity between a node n_i and the closest common ancestor of node n_i and n_j as the number of nodes on the path from the root to the common ancestor over the number of nodes on the path from the root to node n_i.

$$sim(n_i, anc(n_i, n_j)) = \frac{|\{N \mid N(anc(n_i, n_j))\}|}{|\{N(n_i)\}|} \quad (8)$$

In this equation, $\{N(n_i)\}$ is the set of nodes from the root to node n, which corresponds to the path from the root to the node, $path_{root}(n_i)$. Subsequently, $\{N \mid N(anc(n_i, n_j))\}\}$ is the set of nodes from the root to the closest common ancestor of both nodes n_i and n_j, which corresponds to the path from the root to the closest common ancestor of both nodes, $path_{root}(anc(n_i, n_j))$. Note that in our model, the root node is not contained in the path and the notation $path_{root}(n_i)$ is used instead of $\{N(n_i)\}$ to simplify the writing of the equations. By considering equation (7) and (8), we can improve equation (5) to measure the similarity between heterogeneous outcomes o_i and o_j as:

$$asim(o_i, o_j) = \rho \times \frac{|path_{root}(anc(o_i, o_j))|}{|path_{root}(o_i)|} + (1-\rho) \times \frac{|path_{root}(anc(o_i, o_j))|}{|path_{root}(o_j)|} \quad (9)$$

Following this, if we know that a user has liked outcome o_i with a value equal to x, then we can estimate that he or she will also like outcome o_j by a value equal to $x*asim(o_i, o_j)$. Consequently, if we know that a user has liked the values $d_{1,1}$ and liked $d_{2,1}$ by a value x, then we can estimate that they will like $d_{2,2}$ by a value equal $x*asim(d_{2,1}, d_{2,2})$. Hence, we can make use of concept tree as user model to store the user's utility values and also estimate missing ones by looking at the similarity between the concept representing the missing value and the nearest concept on which the user has expressed a preference.

Once all the utility values are known, we propose a novel way of estimating the weights. Rather than eliciting the weights from the user, we suggest to estimate them by multiple regression. This is made possible by the fact that each user has rated a given number of items and thus we can obtain a system of n equations with n unknowns.

3.2 Definitions

Section 3.1 introduced the idea that attributes could be modeled by a concept tree, and that a concept tree could be used to estimate missing values. We can extend this idea on domain values as they usually have also some kind of relationships between them. For example, if you like *action* movies then you will probably also like *adventure* ones as action and adventure are closely related. We represent this information by an *attribute concept tree,* where the domain values are modeled by concepts, the value of the concept represents the user's utility value, and the relationships between them are *is-a* relations.

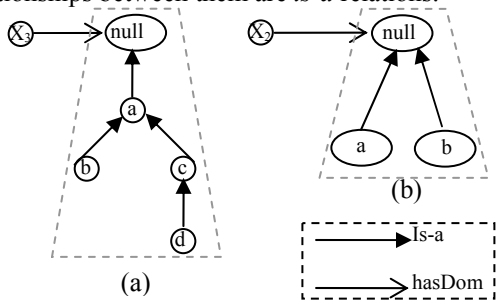

Figure 4: (a) ACT with is-a relationship between domain values, and (b) without relationships

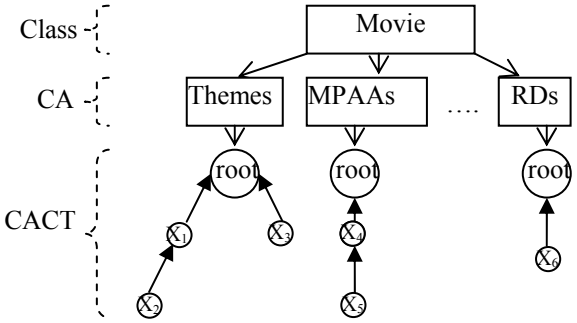

Figure 5: Overview of Movie Ontology

Definition 1 *An attribute concept tree ACT_i is defined by the 8-tuple: $<X_i, D_i, T_i, \leq_i, DCT_i, f_{DCT,i}, sim_{DCT,i}, comb_{DCT,i}>$, where*

- X_i is a concept representing the attribute X_i and linked to DCT_i with the binary relations *hasDom*;
- D_i is the domain of the attribute X_i;
- T_i defines the type of all the elements in D_i;
- $\leq_i$ is an ordering relation on all the element of D_i;
- DCT_i is a concept tree ontology modeling all the elements in D_i and with the *null* concept as the root;
- $f_{DCT,i}$ is a function that given a domain value from D_i returns the corresponding concept in DCT_i;

- $sim_{DCT,i}$ is a function that computes the similarity between any two concept in DCT_i; and
- $comb_{DCT,i}$ is a function that estimates the value of a concept in DCT_i.

When the domain D_i is discrete, HAUM exploits the *is-a* relationship between the values by building a hierarchy (Figure 4.a). On the other hand, if a domain value is without any relationship, then it will be directly attached to the null concept (Figure 4.b). The null concept represents the null domain value and is used if an object does not contain that attribute. However, if D_i is continuous, we tend to use the classic ordering (i.e.:<) and each element of D_i is directly connected to the root concept as before. Hence, if we know that two concepts are linked to the root concept, then we can deduce that they have nothing in common and the similarity function will return zero.

In many situations, different attributes represent the same concept or have similar meaning. In the computer domain, for example, the attributes processor speed and RAM are related because large memory usually implies a fast processor. Our model exploits this pattern by grouping similar attributes together in a set called *compound attribute*.

Definition 2 *A compound attribute concept tree $CACT_j$ is defined by the 7-tuple: $<CA_j, Z_j, \leq_j, AsCT_j, f_{AsCT,j}, sim_{AsCT,j}, comb_{AsCT,j}>$, where*

- CA_j is a compound attribute concept representing the compound attribute CA_j and linked to $AsCT_j$ with the binary relations *hasDom*;
- Z_j is the set of attributes in CA_j;
- $\leq_j$ is an ordering relation on all the element of Z_j;
- $AsCT_i$ is a concept tree ontology modeling all the elements in Z_j and with the *root* concept as the root;
- $f_{AsCT,i}$ is a function that given an attribute from CA_i returns the corresponding concept in $AsCT_j$;
- $sim_{AsCT,j}$ is a function that computes the similarity between any two concept in $AsCT_j$; and
- $comb_{DCT,j}$ is a function that estimates the value of a concept in $AsCT_j$.

Note that the concepts of the AsCT are in fact attribute concept trees. Informally, CACT can be seen as a bi-dimensional ontology where one dimension represents the attributes, while the other represents their corresponding domain values. Similarly to the ACT, if 2 attributes have no relationship between them, then they will be attached to the root concept.

Definition 3 *Given an ontology λ made of CACTs, we define the heterogeneous attribute utility model (HAUM) as an extension from the multi-attribute utility theory (MAUT) where all the outcomes are modeled by λ, and where the mutual preferential independence hypothesis holds on all the attributes.*

Definition 4 *Given a problem where outcomes are defined using the HAUM model, the optimal solution, if exists, is the outcome maximizing the utility in* (10).

$$V(o_k) = \sum_{i=1}^{M} w_i' v_i'(o_k) \quad , where \quad v_i'(o_k) = \sum_{j=0}^{n_i} w_j v_j(o_k) \tag{10}$$

In this equation, M is the number of compound attributes defining the class modeling the outcomes, and n_i is the number of attributes in the compound attribute i. The sub-utility functions v_j and v'_i, and the weights w'_i of the compound attributes are defined on the interval [-1,1] and $\sum|w_i'|=1$. However, the weights w_j are defined on the interval [0,1] and $\sum w_j=1$. In our model, we apply the Equal Weight Policy [21] to compute the value of each w_j. This strategy consists of assigning the same value ($w_j=1/n_i$) to each weight while making sure that $\sum w_j=1$. We have preferred this strategy as it has a very high relative accuracy compared to more complex one [21] while still being very simple to implement. The computation of the w_i' is explained in details in section 3.4.2.

3.3 Reasoning with the ontology

We can now extend the reasoning of section 3.1 to take into account the specificity of our model. Firstly, equation (9) is generalized to measure the similarity between any two concepts in a given concept tree as:

$$asim(c_i, c_j) = \rho \times \frac{|\, path_{root}(anc(c_i, c_j))\,|}{|\, path_{root}(c_i)\,|} + (1-\rho) \times \frac{|\, path_{root}(anc(c_i, c_j))\,|}{|\, path_{root}(c_j)\,|} \tag{11}$$

Where ρ is the same generalization coefficient defined in section 3.1. In our experiments, ρ was arbitrary set to 0.75 but further research is under progress to determine exactly how to evaluate this coefficient.

HAUM estimates the missing utility value of concept c_i by looking at the set of the closest concepts on which we have preferences, CCP, and compute an average value based on the similarity it has with the concept c_i using:

$$utility\_value(c_i) = \frac{\sum_{i=0}^{|\{CCP\}|} (value(\{CCP\}_i) * asim_x(\{CCP\}_i, c_i))}{\sum_{i=0}^{|\{CCP\}|} asim_x(\{CCP\}_i, c_i)} \quad (12)$$

In this equation, $\{CCP\}_i$ is the i^{th} closest concept on which we have a utility value, and sim_x is the similarity metric of the ACT_x or a DCT_x.

3.4 HAUM process

The HAUM algorithm is a four steps iterative process as show in Figure 6.

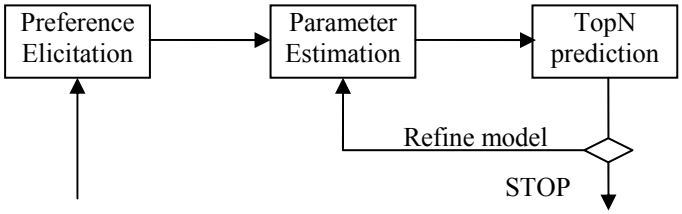

Figure 6: Illustration of the HAUM Process

3.4.1 Preference elicitation

The preference elicitation process is designed to be very simple and similar to collaborative filtering. Each user is asked to give at least M outcomes with their associated ratings. The number M corresponds to the number of compound attributes whilst the rating is usually an integer ranging from one to five. It is necessary to ensure that the given objects are uniformly distributed in two distinct sets: liked or disliked objects that we call respectively LS and DS. The union of both sets, the user's preference set (UPS), is going to be our learning set for the parameters estimation process.

3.4.2 Parameter estimation

Once the elicitation is completed, we apply a simple data mining algorithm, *frequency count* (FC), on the UPS in order to estimate some of the utility values. The FC works as follows: first we convert the grade into a utility value ranging from [-1, .., 1]. Then, for each present domain value of each attribute, we sum its utility value and divide it by the number of times it was present in the UPS. This is a simple algorithm that assumes the independence of the attributes and do not look at any combination what so ever. Note that our model is built on the additive utility model that makes such hypothesis.

Unfortunately, the Frequency Count algorithm is very unlikely to have computed the sub-utility of each possible domain value. Consequently, we use a *utility estimation* (UE) algorithm that estimates the missing utilities based on the surrounding concepts in a CACT. The utility estimation will be done using equation (12). However, this equation needs to identify the closest concepts CCP on which we have utility values. To do this, we start by instantiating the various DCTs in the ontology with the utilities computed by the FC algorithm. Next, we select each concept without values, look for its closest neighbors that have values, and apply equation (12). Two cases need to be considered when looking for neighbors: when neighbors in the same Attribute Concept Tree can be found, or when they cannot. The former is straight forward and requires navigating through the DCT while the latter is more complex and requires jumping from one ACT to another via the CACT until we find a similar concept with value.

Once all the sub-utility functions have been computed, we have to estimate the weights. After the UE algorithm has been applied, we are left with a system with at least M equations, M unknowns, and M grades. We translate the grades into adequate utility values and then use classical multiple regression (MR) to estimate the weights of the compound attributes. Finally, the weights are normalized to satisfy to the properties defined in section 3.2.

3.4.3 Top-N selection

Once all the user parameters have been estimated, we can compute the utility of each outcome by applying equation (10). Finally, we rank the outcomes in decreasing order of the computed utility and select the first N one; this is called the *top-N items recommendation* strategy where N is a parameter set by the system and usually $\in$ [3, .., 10].

3.4.4 Refinement process

Finally, the user has the opportunity to add or remove outcomes from the UPS in order to refine his model. Furthermore, he can directly modify the utility value or weights assigned to each attribute. This allows building a dynamic user model using both implicit and explicit preferences.

4 EXPERIMENTAL RESULTS

In this section, we explain the experimental methodology and the metric used to validate the hypothesis of our model. We ran our model on the famous MovieLens[2] data; we used this data set as it is widely being used through out the research community [11][14][15] and it contains the data requires to solve our example given in the introduction. MovieLens is a data set containing the rating of 943 users on at least 20 movies. There are in total 1682 movies in the database described by 19 themes (drama, action, …, war). To increase the description of the movies, we wrote a wrapper that extracted the year, MPPA rating, duration, actors and directors (due to the high sparsity of the actors and directors, it was decided to ignore those attributes in our experiments) from the IMDB[3] website given a movie's URL.

4.1 HAUM model analysis

The most important aspect we tested was whether or not the HAUM using 5 ratings in the learning set was close to the optimal model. To test this facet, we implemented a cross validation test close to the one proposed by Salzberg [13] with k=5 and using the McNemar test. For the purpose of this experiment, we simplified the model and only considered one compound attribute: CA_{Themes} and its 19 attributes. We implemented two different models: a data-rich one – RICH, and our new model – HAUM. Obviously, obtaining the optimal model as such is impossible as it is unknown. Therefore, we estimated it by using the frequency count algorithm and learned the utility values of all the domain values using 50 ratings from the user's preference set. On the other hand, the HAUM model used 5 ratings from the user's preference set to learn the utility value of 5 representative attributes, the RCSet, while the remaining were estimated using equation (12) and with a ρ set to 0.75.

The experiment was as follows: we filtered the users with insufficient data and only kept those with at least 75 ratings in the MovieLens data set; after this selection process, the number of users was reduced from 943 to 286. For all remaining users, we randomly selected exactly 75 ratings from the MovieLens data set, and 50 unrated movies that we stored respectively in two sets; the learning set and the test set. The learning set was then divided into k subsets, where k=5, for cross validation, while the test set was used to test both models. For each subset of the learning set, 5 ratings were randomly selected to estimate the values of the RCSet, whilst another 50 ratings were randomly picked from the remaining four subsets to estimate all the utility values of the RICH model. Once all the utility values of each model were computed, we used both models to estimate the rating of every movie in the test set. As our experiment used two models to rate an item, the rating of a movie was represented as a tuple: $\langle rating_{HAUM}, rating_{RICH} \rangle$. Once all the movies have been rated, each rating was rescaled as follows: if the rating was from 1 to 4 exclusive, then we assigned the value *rejected*; otherwise we used the value *accepted*. For each user, we then counted how many times each combination (i.e.: ⟨accepted, accepted⟩… ⟨rejected, rejected⟩) was obtained and added the result into the table *result*. This table contained the possible rating combination for each user where the rows represent the four rating combinations and the columns represent the users. Finally, and after the k runs, the data in the table result was averaged, summed up, and the McNemar test was performed to test the difference in prediction between the two models.

After 5 runs of the experiments, we obtained an average chi-square of 2.03 and p-value of 0.157. This p-value clearly indicates that we cannot reject the null hypothesis that both models perform equally. Concretely, this means

[2] http://www.cs.umn.edu/Research/GroupLens/data/ml-data.zip
[3] http://www.imdb.com

that the difference in prediction between the optimal model and HAUM is not statistically significant and show that our model with just 5 ratings is close to the optimal one, so that inferring preferences through the ontology indeed seems to make sense.

4.2 Overall performance analysis

Finally, we tested the accuracy of the HAUM against existing recommendation techniques and studied how many ratings were required by the parameters estimation algorithm in order to obtain an accurate model.

The experiment was as follows. First, users with less than 115 ratings were removed from the data set and for each remaining user, 15 ratings were inserted into a test set while the rest was inserted into a temporary set. From the temporary set, 11 learning sets of varying size were created in order to test the accuracy of the model. The size of the sets varied incrementally from 4 to 100 ratings and with $learning\_set_{i+1}=learning\_set_i \cap ratings\_from\_temporary\_set$. Various techniques were used to estimate the weights and the utility functions from the learning set: the model defined in the previous section with a ρ set to 0.75 – *HAUM*, random utility values for all the attributes but compound attributes' weights estimated by multiple regression– *randFunc*, and random compound attributes' weights but with the utility value estimated by the ontology – randWeights. HAUM was benchmarked against the random policy – *RAND*, and the adjusted cosine collaborative filtering with 90 neighbors – *CF*. Collaborative Filtering algorithm was chosen as benchmark over classical content based filtering as it is known that it is today's best performing filtering and most widely used recommendation system. Moreover, experimental results by Melvielle et al [10] have shown that CF performs better than pure content based filtering in the movie domain. We did not implement the CF algorithm; instead, we used the freely available MultiLens[4] package with the filter ZScore which will allow us to perform adjusted cosine similarity. We set the neighbors to 90 as authors [11][14] have shown that the optimal for the MovieLens data set is very close to this value.

The top-5 policy was used to select the 5 best outcomes based on their estimated grade. The accuracy of the prediction was measured using the Mean Absolute Error (MAE) in order to compare it with existing techniques. However, Herlocker et al in [7] have argued that MAE is a less appropriate measure when considering the top-N policy over rated items due to the small granularity of the user's preferences.

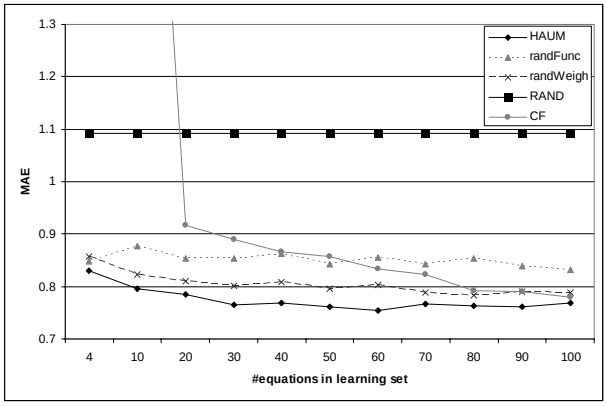

Figure 7: MAE vs. size of learning set with policy

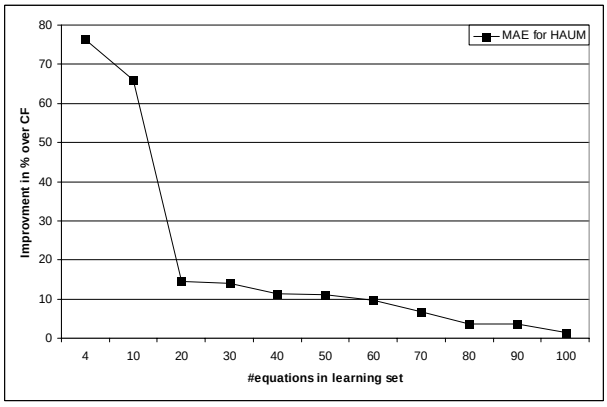

Figure 8: Improvement of HAUM over CF

The results illustrated in Figure 7 were much better than expected and shows that HAUM can be used to build robust recommendation systems. As expected, collaborative filtering performs poorly when the number of ratings in the learning set is very low. It rapidly improves until it reaches the 20 ratings threshold and then the improvement slows down. This is a well known effect that is due to the high data sparsity (sparsity>0.99), it also reflects the behavior of the systems when new items are added and need to be evaluated. These results clearly show that CF is unsuitable for the volatile environment when dealing with low involvement user decision process. On the other hand, the HAUM approach performs extremely well even with high data sparsity. Furthermore, it performs much better than CF (Figure 8) with improvement up to 76.4%. The student test reinforces this statement with a t-obs=-39.99 and t-

[4] http://knuth.luther.edu/~bmiller/multilens.html

value<<0.01. The random policy was also plotted to see if our HAUM approach does actually improve something; and with an MAE of ~1.09, it is clear that it does.

What is interesting is to consider whether or not our utility estimation algorithm and frequency count is better than just using random values. The graph clearly illustrates that HAUM performs better than randFunc and statistical analysis showed that the improvement is significant when we had at least 10 ratings in the learning set (t-test: t-obs= 2.99, p-value=0.0016). This behavior makes sense and implies that a minimum of knowledge on some of the values must be known in order to estimate the rest of the utility values. Concerning the utility of the multiple regression to estimate the weights, we cannot assert anything as we obtained a p-value~0.1. However, the graph shows that HAUM performs slightly better than randWeights whatever the size of the learning set. Finally, it is worth pointing out that with only 4 ratings in the learning set, the HAUM model is able to perform reasonable well. The optimal accuracy is obtained with 60 ratings in the learning set, which is 10% better that CF. Moreover, this experiment shows that HAUM is quite robust as the accuracy hardly changes with varying size of the learning set.

4.3 Discussion

Many researchers have tried to boost the accuracy of Collaborative Filtering. Melvielle et al [10] is probably the most famous work in this domain. They use a content-based approach to try to fill up the CF's matrix in order to avoid the sparsity problem and new-item problem. They performed experiments on the EachMovie data set and obtained an improvement of 4% over classical CF when they have 25% of the ratings in the test set. With this configuration, HAUM generates an improvement of over 10%. However, MovieLens is a subset of the EachMovie data so we cannot draw any clear conclusion. Mobasher et al in [11] improved CF by adding semantic context to the problem. They extracted the attribute values of each movie and from it created a semantic matrix. This matrix is exploited in order to compute the similarity between two movies. Finally, they predict the rating using a weighted combination of the semantic matrix result and CF. Their method shows very good performance with up to 22% improvement on the MAE when data sparsity is high. The best improvement is achieved when the ratio of the learning / test set is around 0.3. At very low ratio, the improvement is around 15% which is 60% less than our HAUM approach.

Those results tend to show that we could improve our HAUM by combining it in some form with CF. By doing that, we could improve the accuracy but we would then violate the user's privacy and make it vulnerable to schilling attacks. This would have to be a trade-off that has to be studied.

5 CONCLUSIONS

Existing recommendation techniques include insufficient inductive bias to obtain an accurate user model from the amount of data that is typically available. We have shown how ontologies can provide the right inductive bias so that accurate recommendations are possible even with very little data about the user's preferences. On experiments using the MovieLens data, they consistently outperform collaborative filtering even when every little data is available.

Note that our HAUM technique does not use any information about other user's preferences as it the case in collaborative filtering. Instead, knowledge of what is common behavior is brought in through the ontology. The next step is now to apply ontology learning techniques to automatically construct reasonable ontologies for a given attribute model of the items. This should allow to better fine-tune the ontology to actual user preferences, and achieves further significant performance gains. We also consider using collaborative approach, as well as learning technique, to try to better estimate the generalization coefficient rather than arbitrary value.

6 References

[1] T. Andreasen, H. Bulskov, and R. Knappe, *From Ontology over Similarity to Query Evaluation*, In 2nd International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems (ODBASE), Catania, Sicily, Italy, November 2003

[2] J. Blythe, *Visual Exploration and Incremental Utility Elicitation*, In Eighteenth national conference on Artificial intelligence, Edmonton, Alberta, Canada , pp. 526 - 532, 2002.

[3] K. Bradley, R. Rafter, and B. Smyth, *Cased-Based User Profiling for Content Personalization*, In Proceedings

of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, 2000.

[4] M. Claypool, A. Gokhale, and T. Miranda, *Combining Content-Based and Collaborative Filters in an Online Newspaper,* In ACM SIGIR Workshop on Recommender Systems, 1999.

[5] V. Ha, and P. Haddawys, *Problem-focused incremental elicitation of multi-attribute utility model.* In Besnard, P., and Hanks, S., eds., Proc. 13th Conference on Uncertainty in Artificial Intelligence, pp. 215-222, 1997.

[6] V. Ha, and P. Haddawys, *A Hybrid Approach to Reasoning with Partially Elicited Preference Models*, In 15th Conference in Uncertainty in Artificial Intelligence,UAI-99, pp. 263-270, Stockholm, Sweden, July 1999

[7] J.L. Herlocker, J.A. Konstan, L. G. Terven, and J.T. Riedl, *Evaluating Collaborative Filtering Recommender Systems*, In ACM Transactions on Information Systems, 22(1):5-53, January 2004.

[8] R. Keeney, and H. Raiffa. *Decisions with Multiple Objectives: Preference and Value Tradeoffs,* Cambridge University Press, 1993.

[9] G. Linden, B. Smith, and J. York, *Amazon.com Item-to-Item Collaborative Filtering*, IEEE Internet Computing, January-February 2003.

[10] P. Melville, R. J. Mooney, and R. Nagarajan, *Content-Boosted Collaborative Filtering*, In Proceeding of the SIGIR-2001, Workshop on Recommender Systems, New Orleans, LA, 2001.

[11] B. Mobasher, X. Jin, and Y. Zhou, *Semantically Enhanced Collaborative Filtering on the Web*, in Web Mining: From Web to Semantic Web, EWMF04, LNAI Volume 3209, Springer, 2004.

[12] P. Resnik, *Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language*, in Journal of Artificial Intelligence Research, pp 95-130, 1999.

[13] S. L. Salzberg, *On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach,* In Data Mining and Knowledge Discovery, Volume 1, Issue 3, pp 317-327, 1997.

[14] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, *Analysis of Recommendation Algorithms for E-Commerce*, In the ACM Conference on Electronic Commerce, EC2000, October 2000.

[15] A.L. Schein, A. Popesucl, L.H. Ungar and D. M. Pennock, *Methods and Metrics for Cold-Start Recommendations*, In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 2002.

[16] M. Stolze, *Soft navigation in electronic product catalogs*, In International Journal on Digital Libraries, 3(1):60-66, July 200.

[17] D.O. Sullivan, B. Smyth, D.C. Wilson, K. McDonald, and A. Smeaton, *Improving the Quality of the Personalized Electronic Program Guide*, In User Modeling and User-Adapted Interaction, 14(1), 2004.

[18] G. Villard, *Computation of the Inverse and Determinant of a Matrix*, In Algorithms Seminar INRIA, pp. 29-32, 2003.

[19] J. Von Neumann, and O. Morgenstern, The Theory of Games and Economic Behavior. *Princeton University Press*, Princeton, 1944.

[20] J. Yang, L. Wenyin, H. Zhang, and Y. Zhuang, *Thesaurus-Aided Approach For Image Browsing and Retrieval*, In IEEE International Conference on *Multimedia and Expo,* Tokyo, August 2001.

[21] J. Zhang, and P. Pu, *Effort and Accuracy Analysis of Choice Strategies for Electronic Product Catalogs*, In Proceedings of the 2005 ACM Symposium on Applied computing, SAC-2005, pp. 808- 814, 2005.