Österreichisches Forschungsinstitut für /
Austrian Research Institute for /
Artificial Intelligence

TR–2003–26

Eyke Hüllermeier, Johannes Fürnkranz

Preference Learning: Models, Methods, Applications – Proceedings of the KI-2003 Workshop

• Freyung 6/6 • A-1010 Vienna • Austria •

• Phone: +43-1-5336112 •

• `mailto:sec@oefai.at` •

• `http://www.oefai.at/oefai/` •

Österreichisches Forschungsinstitut für / Austrian Research Institute for / Artificial Intelligence

TR–2003–26

Eyke Hüllermeier, Johannes Fürnkranz

Preference Learning: Models, Methods, Applications – Proceedings of the KI-2003 Workshop

The Austrian Research Institute for Artificial Intelligence is supported by the Federal Ministry of Education, Science and Culture.

Citation: Hüllermeier E., Fürnkranz J.: Pairwise Preference Learning and Ranking. Technical Report, Österreichisches Forschungsinstitut für Artificial Intelligence, Wien, TR-2003-26, 2003

Preference Learning: Models, Methods, Applications

The preferences of an individual, say, the participant of an electronic auction or the customer of an electronic store, can be expressed in various ways, either explicitly, e.g., in the form of preference statements or implicitly, e.g., through the way of acting in different situations. The problem of finding out about an individual's preferences, or about those of a group of individuals, is referred to as preference elicitation. This requires, among other things, models for the formal representation and methods for the (automatic) acquisition of preferences. Touching on various aspects of Artificial Intelligence, both theoretical and practical, preference elicitation is one of this field's most recent and interesting research topics.

Like other types of complex learning tasks that have recently entered the stage in the field of machine learning, preference learning deviates strongly from the standard machine learning problems of classification and regression. It is particularly challenging because it involves the prediction of complex structures, such as weak or partial order relations, rather than single values. Moreover, training input will not, as it is usually the case, be offered in the form of complete examples but may comprise more general types of information, such as relative preferences or different kinds of indirect feedback. For example, learning problems might be posed by providing – or, in the style of an active learner, by asking for – preference relations between the training examples rather than a target value (as in supervised learning) or a utility degree (as in reinforcement learning).

Apart from posing challenging theoretical problems, preference learning is highly relevant from a practical point of view. As an example, consider the important application of autonomous (web) agents performing various tasks on the net, such as acting on behalf of a user in electronic commerce or recommending decisions to users in collaborative filtering. Following the major paradigm of modern Artificial Intelligence, namely that of a rationally acting agent, ideas and concepts from decision theory often serve as a theoretical basis for implementing such agents. The behavior of a decision-theoretic agent has to be driven by an underlying preference model, and an agent recommending decisions or acting on behalf of a user should clearly reflect that user's preferences. Therefore, the formal modeling as well as the automatic learning, discovery and adaptation of preferences can be considered an essential aspect of autonomous agent design.

Workshop organizers

Eyke Hüllermeier
Department of Mathematics and Computer Science
Marburg University

Johannes Fürnkranz
Austrian Research Institute for
Artificial Intelligence, Vienna

Members of the organizing committee

Contents

Pairwise Preference Learning and Ranking

Johannes Fürnkranz
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Wien, Austria
juffi@oefai.at

Eyke Hüllermeier
Informatics Institute, Marburg University
Hans-Meerwein-Str., Lahnberge, D-35032 Marburg, Germany
eyke@mathematik.uni-marburg.de

Abstract

We consider supervised learning of a ranking function, which is a mapping from instances to total orders over a set of labels (options). The training information consists of examples with partial (and possibly inconsistent) information about their associated rankings. From these, we induce a ranking function by reducing the original problem to a number of binary classification problems, one for each pair of labels. The main objective of this work is to investigate the trade-off between the quality of the induced ranking function and the computational complexity of the algorithm, both depending on the amount of preference information given for each example. To this end, we present theoretical results on the complexity of pairwise preference learning. We also carry out some controlled experiments investigating the predictive performance of our method for different types of preference information, such as top-ranked labels and complete rankings. The domain of this study is the prediction of a rational agent's ranking of actions in an uncertain environment.

1 Introduction

The increasing trend to treat consumers, computer users and patients as *individuals* has produced, among other things, user-adapted software and operating systems (Horvitz et al., 1998), e-commerce personalization of products and services (Riecken, 2000), and systems for patient-centered medical care (Couch, 1998). A key prerequisite in all of these applications is the ability of discovering and capturing an individual's *preferences*, a problem often referred to as *preference elicitation*.

We consider the acquisition of preferences in the context of supervised learning. Roughly speaking, this means to generalize given examples to a "preference structure-valued" function, that is, a function which assigns preference structures to instances (computer users, customers, patients, ...). This problem, which can obviously be seen as an extension of learning a classification function, will be referred to as *preference learning*. It should be distinguished from preference elicitation in a more narrow sense, where the goal is to learn about the preferences of a single individual, and where specific questions can be asked to that individual.[1]

The problem of learning with or from preferences has recently received a lot of attention within the machine learning literature. The problem is particularly challenging because it involves the prediction of complex structures, such as weak or partial order relations, rather than single values. Moreover, training input will not, as it is usually the case, be offered in the form of complete examples but may comprise more general types of information, such as relative preferences or different kinds of indirect feedback.

More specifically, the learning scenario that we will consider in this paper consists of a collection of training examples which are associated with a finite set of decision alternatives. Following the common notation of supervised learning, we shall refer to the latter as *labels*. However, contrary to standard classification, a training example is not assigned a single label, but a set of *pairwise preferences* between labels, expressing that one label is preferred over another.

The goal is to use these pairwise preferences for predicting a total order, a *ranking*, of all possible labels for a new training example. More generally, we seek to induce a *ranking function* that maps instances (examples) to rankings over a fixed set of decision alternatives (labels), in analogy to a *classification function* that maps instances to single labels. To this end, we investigate the use of *round robin learning* or *pairwise classification*. As will be seen, round robin appears particularly appealing in this context since it can be extended from classification to preference learning in a quite natural manner.

The paper is organized as follows: In the next section, we introduce the learning problem in a formal way. The extension of pairwise classification to pairwise preference learning and its application to ranking are discussed in section 3. Section 4 provides some results on the computational complexity of pairwise preference learning. Results of several experimental studies investigating the predictive performance of our approach under various training conditions are presented in

[1]Here, the major problem is to ask such questions in a clever way, so as to find a good approximation of the individual's preference structure with an as small as possible number of questions.

section 5. We conclude the paper with an overview of related work in section 6 and some complementary final remarks in section 7.

2 Learning Problem

We consider the following learning problem:

Given:

- a set of *labels* $L = \{\lambda_i \mid i = 1 \dots c\}$
- a set of *examples* $E = \{e_k \mid k = 1 \dots n\}$
- for each training example e_k:
  - a set of *preferences* $P_k \subseteq L \times L$, where $(\lambda_i, \lambda_j) \in P_k$ indicates that label λ_i is preferred over label λ_j for example e_k.

Find: a function that orders the labels $\lambda_i, i = 1 \dots c$ for any given example.

We will abbreviate $(\lambda_i, \lambda_j) \in P_k$ with $\lambda_i \succ_k \lambda_j$, or even $\lambda_i \succ \lambda_j$ if the particular example e_k doesn't matter or is clear from the context.

This setting has been previously introduced as *constraint classification* by Har-Peled et al. (2002). As has been pointed out in their work, the above framework is a generalization of several common learning settings, in particular (see ibidem for a formal derivation of these and other results)

- *ranking:* Each training example is associated with a total order of the labels, i.e., for each pair of labels (λ_i, λ_j) either $\lambda_i \succ \lambda_j$ or $\lambda_j \succ \lambda_i$ holds.

- *classification:* A single class label λ_i is assigned to each example. This implicitly defines the set of preferences $\{\lambda_i \succ \lambda_j \mid 1 \leq j \neq i \leq c\}$.

- *multi-label classification:* Each training example e_k is associated with a subset $S_k \subseteq L$ of possible labels. This implicitly defines the set of preferences $\{\lambda_i \succ \lambda_j \mid \lambda_i \in S, \lambda_j \in L \setminus S\}$.

As pointed out before, we will be interested in predicting a ranking (total order) of the labels. Thus, we assume that for each instance, there exists a total order of the labels, i.e., they form a transitive and asymmetric relation. For many practical applications, this assumption appears to be acceptable at least for the *true* preferences. Still, more often than not the observed or *revealed* preferences will be incomplete or inconsistent. Therefore, we do not require the *data* to be consistent in the sense that transitivity and asymmetry applies to the P_k. In fact, this property is not compulsory for our learning algorithm. Yet, we do make the reasonable assumption that P_k is irreflexive ($\lambda_i \not\succ \lambda_i$) and anti-symmetric ($\lambda_i \succ \lambda_j \Rightarrow \lambda_j \not\succ \lambda_i$). (Note that $0 \leq |P_k| \leq c(c-1)/2$ as a consequence of the last two properties.)

3 Pairwise Preference Ranking

A key idea of our approach is to learn a separate theory for each of the $c(c-1)/2$ pairwise preferences between two labels. More formally, for each possible pair of labels (λ_i, λ_j), $1 \leq i < j \leq c$, we learn a model m_{ij} that decides for any given example whether $\lambda_i \succ \lambda_j$ or $\lambda_j \succ \lambda_i$ holds. The model is trained with all examples e_k for which either $\lambda_i \succ_k \lambda_j$ or $\lambda_j \succ_k \lambda_i$ is known. All examples for which nothing is known about the preference between λ_i and λ_j are ignored.

At classification time, an example is submitted to all $c(c-1)/2$ theories, and each prediction is interpreted as a vote for a label. If classifier m_{ij} predicts $\lambda_i \succ \lambda_j$, we count this as a vote for λ_i. Conversely, the prediction $\lambda_j \succ \lambda_i$ would be considered as a vote for λ_j. The labels are ranked according to the number of votes they receive from all models m_{ij}. Ties are first broken according to the frequency of the labels in the top rank (the class distribution in the classification setting) and then randomly.

We refer to the above technique as *pairwise preference ranking* or *round robin ranking*. It is a straight-forward generalization of pairwise or one-against-one classification, aka round robin learning, which solves multi-class problems by learning a separate theory for each pair of classes. In previous work, Fürnkranz (2002) showed that, for rule learning algorithms, this technique is preferable to the more commonly used one-against-all classification method, which learns one theory for each class, using the examples of this class as positive examples and all others as negative examples. Round robin has also been successfully used in other fields, in particular in the area of support vector machines (Hsu and Lin, 2002, and references therein). We refer to Section 8 of (Fürnkranz, 2002) for a brief survey of related work on pairwise classification.

More importantly, however, Fürnkranz (2002) showed that, despite its complexity being quadratic in the number of classes, the algorithm is no slower than the conventional one-against-all technique. We will generalize these results in the next section.

4 Complexity

Consider a learning problem with n training examples and c labels.

Theorem 4.1 *The total number of training examples over all $c(c-1)/2$ binary preference learning problems is*

$$\sum_{k=1}^{n} |P_k| \leq n \max_k |P_k| \leq n \binom{c}{2} = n\frac{c(c-1)}{2}$$

Proof: Each of the n training examples will be added to all $|P_k|$ binary training sets that correspond to one of its preferences. Thus, the total number of training examples is $\sum_{k=1}^{n} |P_k|$. As the number of preferences for each example is bounded from above by $\max_k |P_k|$, this number is no larger than $\max_k |P_k| n$, which in turn is bounded from above by the size of a complete set of preferences $nc(c-1)/2$. 2

From this immediately follows a result of Fürnkranz (2002):

Corollary 4.2 *For a classification problem, the total number of training examples is only linear in the number of classes.*

Proof: A class label expands to $c-1$ preferences, therefore $\sum_{k=1}^{n} |P_k| = (c-1)n$. 2

Note that we only considered the number of training examples, but not the complexity of the learner that runs on these examples. For an algorithm with a linear run-time complexity $O(n)$ it follows immediately that the total run-time is $O(dn)$, where d is the maximum (or average) number of preferences given for each training example. For a learner with a super-linear complexity $O(n^a), a > 1$, the total run-time is much lower than $O((dn)^a)$ because the training effort is not spent on one large training set, but on many small training sets. In particular, for a complete preference set, the total complexity is $O(c^2 n^a)$, whereas the complexity for $d = c - 1$ (round robin classification) is only $O(cn^a)$ (Fürnkranz, 2002).

For comparison, the only other technique for learning in this setting that we know of (Har-Peled et al., 2002) constructs twice as many training examples (one positive and one negative for each preference of each example), and these examples are projected into a space that has c times as many attributes as the original space. Moreover, all examples are put into a single training set for which a separating hyper-plane has to be learned. Thus, under the (reasonable) assumption that an increase in the number of features has approximately the same effect as a corresponding increase in the number of examples, the total complexity becomes $O((cdn)^a)$ if the algorithm for finding the separating hyper-plane has complexity $O(n^a)$ for a two-class training set of size n.

In summary, the overall complexity of pairwise constraint classification depends on the (maximum or average) number of preferences that are given for each training example. While being quadratic in the number of labels if a complete ranking is given, it is only linear for the classification setting. In any case, it is more efficient than the technique proposed by Har-Peled et al. (2002). However, it should be noted that the price to pay is the large number of classifiers that have to be stored and tested at classification time.

5 Empirical Results

The previous sections have shown that an extended version of round robin learning can induce a ranking function from a set of preferences instead of a single label. Yet, it turned out that computational complexity might become an issue. Especially, since a ranking induces a quadratic number of pairwise preferences, the complexity for round robin ranking becomes quadratic in the number of labels. In this context, one might ask whether it could be possible to improve efficiency at the cost of a tolerable decrease in performance: Could the learning process perhaps ignore some of the preferences without decreasing predictive accuracy too much? Apart from that, incomplete training data is clearly a point of practical relevance, since complete rankings will rarely be observable.

The experimental evaluation presented in this section is meant to investigate issues related to incomplete training data in more detail, especially to increase our understanding about the trade-off between the number of pairwise preferences available in the training data and the quality of the learned ranking function. For a systematic investigation of questions of such kind, we need data for which, in principle, a complete ranking is known for each example. This information allows a systematic variation of the amount of preference information in the training data, and a precise evaluation of the predicted rankings on the test data. Since we were not aware of any suitable real-world datasets, we decided to conduct our experiments with synthetic data.

5.1 Synthetic Data

We consider the problem of learning the ranking function of an expected utility maximizing agent. More specifically, we proceed from a standard setting of expected utility theory: $A = \{a_1, \dots, a_c\}$ is a set of actions the agent can choose from and $\Omega = \{\omega_1, \dots, \omega_m\}$ is a set of world states. The agent faces a problem of *decision under risk* where decision consequences are lotteries: Choosing act a_i in state ω_j yields a utility of $u_{ij} \in \mathbb{R}$, where the probability of state ω_j is p_j. Thus, the *expected utility* of act a_i is given by

$$\mathbb{E}(a_i) = \sum_{j=1}^{m} p_j \cdot u_{ij}. \tag{1}$$

Expected utility theory justifies (1) as a criterion for ranking actions and, hence, gives rise to the following preference relation:

$$a_i \succ a_j \Leftrightarrow \mathbb{E}(a_i) > \mathbb{E}(a_j). \tag{2}$$

Now, suppose the probability vector $p = (p_1, \dots, p_m)$ to be a parameter of the decision problem (while A, Ω and the utility matrix matrix $U = (u_{ij})$ are fixed). We denote by $\succ_p$ the ranking of actions induced by the vector p according to (2).

The above decision-theoretic setting can be used for generating synthetic data for preference learning. The set of instances corresponds to the set of probability vectors p, which are generated at random according to a uniform distribution over $\{p \in \mathbb{R}^m \,|\, p \geq 0, p_1 + \ldots + p_m = 1\}$. The ranking function associated with an example e_k is given by the ranking $\succ_{e_k}$ as defined in (2). Thus, an experiment is characterized by the following parameters: The number of actions/labels (c), the number of world states (m), the number of examples (n), and the utility matrix which is generated at random through independent and uniformly distributed entries $u_{ij} \in [0, 1]$.

5.2 Experimental Setup

In the following, we will report on results of experiments with ten different states ($m = 10$) and various numbers of labels ($c = 5, 10, 20$). For each of the three configurations we generated ten different data sets, each one originating from a different randomly chosen utility matrix U. The data sets consisted of 1000 training and 1000 test examples. For each example, the data sets provided the probability vector $p \in \mathbb{R}^m$ and a complete ranking of the c possible actions.[2] The training examples were labeled with a subset of the complete set of pairwise preferences as imposed by the ranking in the data set. The subsets that were selected for the experiments are described one by one for the experiments.

[2]The occurrence of actions with equal expected utility has probability 0.

We used the decision tree learner C4.5 (Quinlan, 1993) in its default settings[3] to learn a model for each pairwise preference. For instances in the test set we obtained a final ranking using simple voting (and tie breaking) as described in section 3. The predicted ranks were then compared with the actual ranks on the test set, and evaluation measures were computed as follows: Denote by $(\rho_k^1, \ldots, \rho_k^c)$ the true ranking of a test example e_k, where ρ_k^1 is the top-ranked label (action). Likewise, denote by $(\tau_k^1, \ldots, \tau_k^c)$ the predicted ranking, again with τ_1^k being the label that has been assigned the top rank. Further, we use $r_k(\lambda_i)$ to denote the true rank of label λ_i for example e_k. The following four evaluation metrics were computed:

Error, the percentage of examples for which the *top rank* was incorrect:

$$\frac{1}{n}\sum_{k=1}^{n} \delta(\tau_k^1, \rho_k^1) \times 100\%,$$

where $\delta(i,j) = 1$ if $i \neq j$ and 0 if $i = j$.

Average Deviation, the average of the (average absolute) deviation of the predicted rank from the true rank:

$$\frac{1}{cn}\sum_{k=1}^{n}\sum_{r=1}^{c} |r - r_k(\tau_k^r)|$$

Maximum Deviation, the average of the maximum (absolute) deviations of the predicted rank from the true rank of each example:

$$\frac{1}{n}\sum_{k=1}^{n} \max_{r=1..c} |r - r_k(\tau_k^r)|$$

Correlation, the average Spearman rank correlation coefficient:

$$\frac{1}{n}\sum_{k=1}^{n} 1 - \frac{6\sum_{r=1}^{c}(r - r_k(\tau_k^r))^2}{c(c^2-1)} \qquad (3)$$

Note that this coefficient assumes values between -1 (for reversed rankings) and $+1$ (for identical rankings).

[3]Our choice of C4.5 as the learner was solely based on its versatility and wide availability. If we aimed at maximizing performance on this particular problem, we would resort to algorithms that can directly represent the separating hyperplanes for each binary preference.

Table 1: Comparison of ranking (a complete set of preferences is given) vs. classification (only the preferences for the top rank are given). Also shown are the results for the complementary setting (all preferences for the top rank are omitted).

c	prefs	error	avg dev.	max dev.	rank corr.
5	ranking	13.380 ± 8.016	0.295 ± 0.096	0.663 ± 0.201	0.907 ± 0.038
	classification	14.400 ± 8.262	0.567 ± 0.234	1.236 ± 0.537	0.783 ± 0.145
	complement	32.650 ± 14.615	0.401 ± 0.120	0.864 ± 0.248	0.872 ± 0.051
10	ranking	15.820 ± 8.506	0.594 ± 0.121	1.823 ± 0.293	0.940 ± 0.018
	classification	16.670 ± 9.549	1.559 ± 0.312	4.103 ± 0.757	0.711 ± 0.108
	complement	24.310 ± 9.995	0.617 ± 0.116	1.858 ± 0.287	0.937 ± 0.018
20	ranking	24.030 ± 4.251	1.012 ± 0.057	3.461 ± 0.204	0.966 ± 0.004
	classification	26.370 ± 5.147	3.320 ± 0.389	10.526 ± 1.125	0.697 ± 0.066
	complement	32.300 ± 3.264	1.026 ± 0.055	3.479 ± 0.191	0.966 ± 0.004

5.3 Ranking vs. Classification

Figure 1 shows experimental results for the cases where pairwise preferences are selected as follows: First, when using the full set of $c(c-1)/2$ pairwise preferences. Second, for the classification setting which uses only the $c-1$ preferences that involve the top label. Third, for the complementary setting that uses the $(c-1)(c-2)/2$ preferences that do *not* involve the top label.

There are several interesting things to note for these results. First, the difference between the error rates of the classification and the ranking setting is comparably small. Thus, if we are only interested in the top rank,[4] it may often suffice to use the pairwise preferences that involve the top label. The advantage in this case is of course the reduced complexity which becomes linear in the number of labels. On the other hand, the results also show that the complete ranking information can be used to improve classification accuracy, at least if this information is available for each training example and if one is willing to pay the price of a quadratic complexity.

The results for the complementary setting show that the information of the top rank preferences is crucial: When dropping this information and using only those pairwise preferences that do not involve the top label, the error rate on the top rank increases considerably, and is much higher than the error rate for the classification

[4]It should be noted that there is nothing special about the top rank. We expect that the same type of results can be observed if we focus on any arbitrary rank (e.g., the bottom rank or the median rank).

setting. This is a bit surprising if we consider that in the classification setting, the average number of training examples for learning a model m_{ij} is much smaller than in the complementary setting. Interestingly, the effective number of training examples for the top labels might nevertheless decrease. In fact, in our learning scenario we will often have a few *dominating* actions whose utility degrees are systematically larger than those of other actions. In the worst case, the same action is optimal for all probability vectors p, and the complementary set will not contain any information about it. While this situation is of course rather extreme, the class distribution is indeed very unbalanced in our scenario. For example, we determined experimentally for $c = m = 10$ and $n = 1000$ that the probability of having the same optimal action for more than half of the examples is $\approx 2/3$, and that the expected Gini-index of the class distribution is $\approx 1/2$.

With respect to the prediction of complete rankings, the performance for learning from the complementary set of preferences is almost as good as the performance for learning from the complete set of preferences, whereas the performance of the ranking induced from the classification setting is considerably worse. This time, however, the result is hardly surprising and can easily be explained by the amount of information provided in the two cases. In fact, the complementary set determines the ranking of $c-1$ among the c label, whereas the top label alone does hardly provide any information about the complete ranking.

As another interesting finding note that the classification accuracy decreases with an increasing number of labels, whereas the rank correlation increases (this is also revealed by the curves in Figure 2 below). In other words, the quality of the predicted rankings increases, even though the quality of the predictions for the individual ranks decreases. This effect can first of all be explained by the fact that the (classification) error is much more affected by an increase of the number of labels. As an illustration, consider random guessing: The chances of guessing the top label correctly are $1/m$, whereas the expected value of the rank correlation (3) is 0 regardless of m. Moreover, one might speculate that the importance of a correct vote of each individual learner m_{ij} decreases with an increasing number of labels. Roughly speaking, incorrect classifications of individual learners are better compensated on average. [5] This conjecture is also supported by an independent experiment in which we simulated a set of homogeneous learners m_{ij} through biased coin flipping with a prespecified error rate. It turned out that the quality measures for predicted rankings tend to increase if the number of labels becomes large (see Fig. 1), though the dependence of the measures on the number of labels is not necessarily monotone.

[5]This gives some intuitive support to the interpretation of round robin learning as an ensemble learning technique (Fürnkranz, 2003).

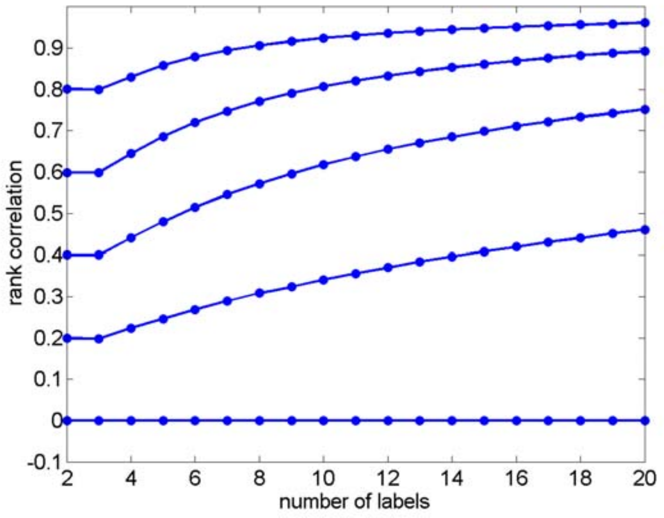


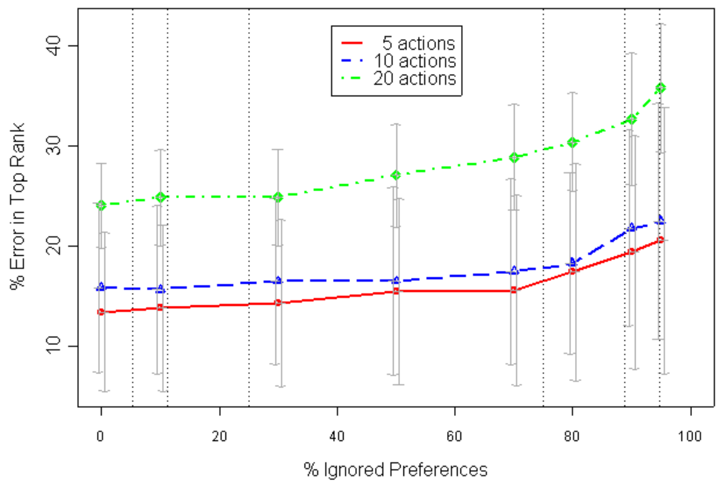
Figure 1: Expected Spearman rank correlation as a function of the number of labels if all learners m_{ij} have an error rate of ϵ (curves are shown for $\epsilon = 0.1, 0.2, 0.3, 0.4, 0.5$).

5.4 Missing Preferences

While the previous results shed some light on the trade-off between utility and costs for two special types of preference information, namely top-ranked labels and complete rankings, they do not give a satisfactory answer for the general case. The selected set of preferences in the classification setting is strongly focused on a particular label for each example, thus resulting in a very biased distribution. In the following, we will look at the quality of predicted rankings when selecting subsets of pairwise preferences from the full sets with equal right.

Figure 2 shows the curves for the classification error in the top rank and the average Spearman rank correlation of the predicted and the true ranking over the number of preferences. To generate these curves, we started with the full set of preferences, and ignored increasingly larger numbers of them. This was implemented with a parameter p_i that caused any given preference in the training data to be ignored with probability p_i ($100 \times p_i$ is plotted on the x-axis).

The similar shape of the three curves (for 5, 10, and 20 labels) suggests that the decrease in the ranking quality can be attributed solely to the missing preferences while it seems to be independent of the number of labels. In particular, one is p



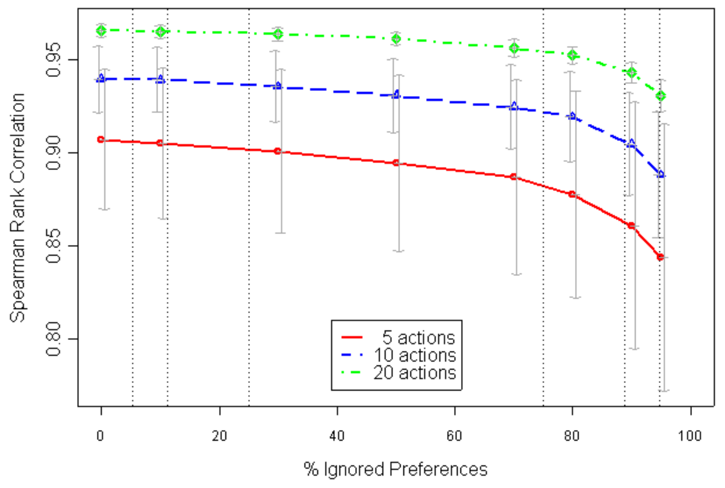


Figure 2: Average error rate (left) and Spearman rank correlation (right) for various percentages of ignored preferences. The error bars indicate the standard deviations. The vertical dotted lines on the right indicate the number of preferences for classification problems (for 5,10, and 20 classes), those on the left are the complementary sizes.

inclined to conclude that—contrary to the case where we focused on the top rank—it is in general *not* possible to reduce the number of training preferences by an order of magnitude (i.e., from quadratic to linear in the number of labels) without severely decreasing the ranking quality. This can also be seen from the three dotted vertical lines on the right. These lines indicate the percentage of preferences that were present in the classification setting for 5, 10, and 20 labels (from inner-most to outer-most). A comparison of the error rates, given by the intersection of a line with the corresponding curve, to the respective error rates in Figure 1 shows an extreme difference between the coincidental selection of pairwise preferences and the systematic selection which is focused on the top rank.

Nevertheless, one can also see that about half of the preferences can be ignored while still maintaining a reasonable performance level. Even though it is quite common that learning curves are concave functions of the size of the training set, the descent in accuracy appears to be remarkably flat in our case. One might be tempted to attribute this to the redundancy of the pairwise preferences induced by a ranking: In principle, a ranking ρ could already be reconstructed from the $c-1$ preferences $\rho_1 \succ \rho_2, \ldots, \rho_{c-1} \succ \rho_c$, which means that only a small fraction of the pairwise preferences are actually needed. Still, one should be careful with this explanation. First, we are not trying to reconstruct a single ranking but rather to solve a slightly different problem, namely to learn a ranking function. Second, our learning algorithm does actually not "reconstruct" a ranking as suggested above. In fact, our simple voting procedure does not take the dependencies between individual learners m_{ij} into account, which means that these learners do not really cooperate. On the contrary, what the voting procedure exploits is just the redundancy of preference information: The top rank is the winner only because it is preferred in $c-1$ out of the $c(c-1)/2$ pairwise comparisons.

Finally, note that the shape of the curves probably also depends on the number of training examples. We have not yet investigated this issue because we were mainly interested in the possibility of reducing the complexity by more than a constant factor without losing too much of predictive accuracy. It would be interesting, for example, to compare (a) using $p\%$ of the training examples with full preferences and (b) using all training examples with $p\%$ of the pairwise preferences.

5.5 Mislabeled Preferences

Recall that our learning scenario assumes preference structures to be complete rankings of labels, that is transitive and asymmetric relations. As already pointed out, we do not make this assumption for *observed* preferences: First, we may not have access to complete sets of preferences (the case studied in the previous section). Second, the process generating the preferences might reproduce the underly-

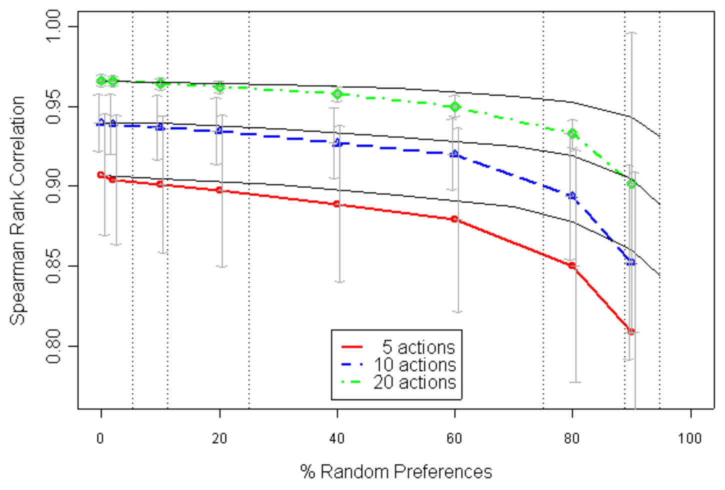


Figure 3: Average Spearman rank correlation over various percentages of random preferences. The error bars indicate the standard deviations. The solid thin lines are the curves for ignored preferences (Figure 2).

ing total order incorrectly and, hence, produce inconsistent preferences. The latter problem is quite common, for example, in the case of human judgments.

To simulate this behavior, we adopted the following model: Proceeding from the pairwise preferences induced by a given ranking, a preference $\lambda_i \succ \lambda_j$ was kept with probability $1 - p_s$, whereas with probability p_s, one of the preferences $\lambda_i \succ \lambda_j$ and $\lambda_j \succ \lambda_i$ was selected by a coin flip. Thus, in approximately $p_s/2$ cases, the preference will point into the wrong direction.[6] For $p_s = 0$, the data remain unchanged, whereas the preferences in the training data are completely random for $p_s = 1$.

Figure 3 shows the average Spearman rank correlations that were observed in this experiment. Note that the shape of the curve is almost the same as the shape of the curves for ignored preferences. It is possible to directly compare these two curves because in both graphs a level of $n\%$ means that $100 - n\%$ of the preferences are still intact. The main difference is that in Figure 2, the remaining $n\%$ of the preferences have been ignored, while in Figure 3 they have been re-

[6]In fact, we implemented the procedure by selecting $p_s/2$ preferences and reversing their sign.

assigned at random. To facilitate this comparison, we plotted the curves for ignored preferences (the same ones as in Figure 2) into the graph (with solid, thin lines).

It is interesting to see that in both cases the performance degrades very slowly at the beginning, albeit somewhat steeper than if the examples are completely ignored. Roughly speaking, completely omitting a pairwise preference appears to be better than including a random preference. This could reasonably be explained by the learning behavior of a classifier m_{ij}: If m_{ij} does already perform well, an additional correct example will probably be classified correctly and thus improve m_{ij} only slightly (in decision tree induction, for example, m_{ij} will even remain completely unchanged if the new example is classified correctly). As opposed to this, an incorrect example will probably be classified incorrectly and thus produce a more far-reaching modification of m_{ij} (in decision tree induction, an erroneous example might produce a completely different tree). All in all, the "expected benefit" of m_{ij} caused by a random preference is negative, whereas it is 0 if the preference is simply ignored.

From this consideration one may conclude that a pairwise preference should better be ignored if it is no more confident than a coin flip. This can also be grasped intuitively, since the preference does not provide any information in this case. If it is more confident, however, it clearly carries some information and it might then be better to include it, even though the best way of action will still depend on the number and reliability of the preferences already available. Note that our experiments do not suggest any strategy for deciding whether or not to include an *individual* preference, given information about the uncertainty of that preference. In our case, each preference is equally uncertain. Thus, the only reasonable strategies are to include all of them or to ignore the complete sample. Of course, the first strategy will be better as soon as the probability of correctness exceeds $1/2$, and this is also confirmed by the experimental results. For example, the correlation coefficient remains visibly above 0.8 even if 80% of the preferences are assigned by chance and, hence, the probability of a particular preference to be correct is only 0.6. One may conjecture that pairwise preference ranking is particularly robust toward noise, since an erroneous example affects only a single classifier m_{ij} which in turn has a limited influence on the eventually predicted ranking.

6 Related Work

As pointed out before, especially relevant for our work is the framework of *constraint classification,* introduced as an extension of standard classification by Har-Peled et al. (2002). The learning method proposed in this work constructs two training examples for each preference $\lambda_i \succ \lambda_j$, where the original d-dimensional training examples are mapped into a cd-dimensional space. The positive example copies the original training vector into the components $d(i-1)+1 \ldots di$ and its negation into the components $d(j-1)+1 \ldots dj$ of a vector in the new space. The remaining entries are filled with 0, and the negative example has the same elements with reversed signs. In this cd-dimensional space, the learner tries to find a separating hyperplane. For classifying a new example e, the labels are ordered according to the response resulting from multiplying e with the i-th d-element section of the hyperplane. This technique also compares favorably to a one-against-all approach.

There has also been some recent work on ranking algorithms. For example, Crammer and Singer (2003) consider a variety of on-line learning algorithms for the problem of ranking possible labels in a multi-label text categorization task. However, we are only aware of one work that actually uses a *complete* ranking of the available labels for each example for training or evaluation: Brazdil et al. (2003) investigate the meta-learning task of ranking learning algorithms according to their suitability for a new dataset, based on the characteristics of this dataset.

Some authors have investigated the problem of preference elicitation in a more narrow sense, that is, the learning of one single preference function. For example, Cohen et al. (1999) propose a two-step approach for ranking a set of objects (and not a set of labels associated with the objects as in our approach) given feedback in the form of preference judgments. Similarly, Haddawy et al. (2003) assume training data to be available in the form of pairwise comparisons of objects. Given such data, they train an artificial neural network that takes as input two objects and outputs either 0 or 1, depending on whether or not the first object is preferred to the second one. (A somewhat similar approach has already been proposed by Wang (1994)). Joachims (2002) analyzes "click-through data" in order to rank documents retrieved by a search engine according to their relevance. This is a nice example of a kind of *indirect* preference information. Using this information, learning of a retrieval function is accomplished by training a support vector machine.

The problem of learning a preference relation over a set of labels L can also be approached in a somewhat indirect way, namely through learning a value or utility function that assigns a utility degree to each label. Note that the preference relation induced by a utility function is necessarily *complete* (linear) in the sense that all tuples of labels are assumed to be comparable. Moreover, note that learning a utility function can be considered a more difficult problem than learning a (linear) preference relation, since the latter subsumes the former but not vice versa.

Depending on the underlying utility scale one can distinguish between learning a numeric function and learning a function that maps into an ordinal (ordered categorical) scale. These two cases involve, respectively, a problem of standard regression and ordinal regression (also called ordinal classification). Ordinal regression has been investigated thoroughly in statistics and econometrics (McCullagh and

Nelder, 1983) and has recently also received attention in machine learning. For example, a method for ordinal regression based on a modification of regression tree learning has been proposed by Kramer et al. (2001). Frank and Hall (2001) suggest a method for translating an ordinal regression problem into a set of ordinary (binary) classification problems. In (Herbich et al., 2000), ordinal regression is approached in the context of support vector machines, using a special type of loss function suitable for comparing predictions on an ordered categorical scale.

The problem of learning (eliciting) *real-valued* utility functions has been investigated in fields such as decision theory and economics for a long time, and has more recently become a topic of research in AI and machine learning as well. A particularly elegant approach is due to Tesauro (1989), who proposes a symmetric network architecture that can be trained with representations of two states and a training signal that indicates which of the two states is preferable. The elegance of this *comparison training* approach comes from the property that one can replace the two symmetric parts of the network with a single network, which can subsequently provide a real-valued evaluation of single states. More recently, Chajewska et al. (1998) simplify the elicitation of utility functions by clustering exemplary utility functions, deriving prototypes from the clusters, and inducing a decision tree whose inner nodes are associated with properties of utility functions (questions that can be asked to a person) and whose leaf nodes are identified with the prototypes. The idea of Chajewska et al. (1999) is to simplify elicitation by exploiting the *additive independence* of variables. Given a database of exemplary utility functions, statistical learning (model selection) methods are used in order to induce a factorization of utility functions into additive subutility functions. Chajewska et al. (2000) accomplish learning of a utility function by treating utility as a random variable. Starting with some prior distribution (derived from analyzing a database of available utility functions), the model is incrementally updated based on information elicited from the user. In order to decide on which questions should be asked next to the user, the authors fall back on the principle underlying the *value of information*. Chajewska et al. (2001) study the problem of learning the utility function that determines the behavior of an agent which is rational in the sense of expected utility theory. The approach proposed by the authors proceeds from a prior probability distribution over a class of utility functions having a certain (linear) structure. The agent's decisions are then used for defining constraints on its true utility function (see (Ng and Russell, 2000) for a quite similar approach). Finally, these constraints are employed in order to turn the prior distribution over the class of utility functions into a posterior distribution.

Learning preferences is also a key topic in recommender systems and collaborative filtering (Goldberg et al., 1992; Resnick and Varian, 1997; Kautz, 1998). Methods proposed in this field are closer to learning utility functions, but are often specifically adjusted to commercial applications where the set of alternatives (labels) to be recommended is usually very large. The method of choice is quite often a case-based or memory-based approach, where the basic idea is to estimate a user's preferences from the preferences of other users that appear to be *similar* (see e.g. Ha and Haddawy (2003); Nakamura and Abe (1998); Billsus and Pazzani (1998)).

7 Concluding Remarks

We have introduced pairwise preference learning as an extension of pairwise classification to constraint classification, a learning scenario where training examples are labeled with a preference relation over all possible labels instead of a single class label as in the conventional classification setting. From this information, we also learn one model for each pair of classes, but focus on learning a complete ranking of all labels instead of only predicting the most likely label. Our main interest was to investigate the trade-off between ranking quality and the amount of training information (in terms of the number of preferences that are available for each example). We experimentally investigated this trade-off by varying parameters of a synthetic domain that simulates a decision-theoretic agent which ranks its possible actions according to an unknown utility function. Roughly speaking, the results show that large parts of the information about pairwise preferences can be ignored in round robin ranking without losing too much predictive performance. In the classification setting, where one is only interested in predicting the top label, it also turned out that using the full ranking information rather than restricting to the pairwise preferences involving the top label does even improve the classification accuracy, suggesting that the lower ranks do contain valuable information. For reasons of efficiency, however, it might still be advisable to concentrate on the smaller set of preferences, thereby reducing the size of the training set raises by an order of magnitude.

The main limitation of our technique is probably the assumption of having enough training examples for learning each pairwise preference. For data with a very large number of labels and a rather small set of preferences per example, our technique will hardly be applicable. In particular, it is unlikely to be successful in collaborative filtering problems (Goldberg et al., 1992; Resnick and Varian, 1997; Breese et al., 1998), although these can be mapped onto the constraint classification framework in a straightforward way. A further limitation is the quadratic number of theories that has to be stored in memory and evaluated at classification time. However, the increase in memory requirements is balanced by an increase in computational efficiency in comparison to the technique of Har-Peled et al. (2002). In addition, pairwise preference learning inherits many advantages of pairwise classification, in particular its implementation can easily be parallelized because of its reduction to independent subproblems.

There are several directions for future work. First of all, it is likely that the prediction of rankings can be improved by combining the individual learners' votes in a more sophisticated way. Several authors have looked at more sophisticated ways for combining the predictions of pairwise theories into a final ranking of the available options. Proposals include weighting the predicted preferences with the classifiers' confidences (Fürnkranz, 2003) or using an iterative algorithm for combining pairwise probability estimates (Hastie and Tibshirani, 1998). However, none of the previous works have evaluated their techniques in a ranking context, and some more elaborate proposals, like error-correcting output decoding (Allwein et al., 2000), organizing the pairwise classifiers in a tree-like structure (Platt et al., 2000), or using a stacked classifier (Savicky and Fürnkranz, 2003) are specifically tailored to a classification setting. Taking into account the fact that we are explicitly seeking a ranking could lead to promising alternatives. For example, we are thinking about selecting the ranking which minimizes the number of predicted preferences that need to be reversed in order to make the predicted relation transitive. Departing from the counting of votes might also offer possibilities for extending our method to the prediction of preference structures more general than rankings (total orders), such as weak preference relations where some of the labels might not be comparable.

Apart from theoretical considerations, an important aspect of future work concerns the practical application of our method and its evaluation using real-world problems. Unfortunately, real-world data sets that fit our framework seem to be quite rare. In fact, currently we are not aware of any data set of significant size that provides instances in attribute-value representation plus an associated complete ranking over a limited number of labels.

Acknowledgments

Johannes Fürnkranz is supported by an APART stipend (no. 10814) of the *Austrian Academy of Sciences*. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture.

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

D. Billsus and M. Pazzani. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pages 46–54. Morgan Kaufmann, 1998.

P. B. Brazdil, C. Soares, and J. P. da Costa. Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, 50(3): 251–277, March 2003.

J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, Madison, WI, 1998. Morgan Kaufmann.

U. Chajewska, L. Getoor, J. Norman, and Y. Shahar. Utility elicitation as a classification problem. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in AI (UAI-98)*, pages 79–88, 1998.

U. Chajewska, D. Koller, and D. Ormoneit. Learning an agent's utility function by observing behavior. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*, pages 35–42, 2001.

U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 363–369, 2000.

U. Chajewska, M. Kuppermann, and D. Koller. Discovering the structure of utility functions based on additive and conditionally additive independence properties between utility attributes. In *Proceedings of the 21st Annual Meeting of the Society for Medical Decision Making (MDM-99)*, 1999.

W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

J. Couch. Disease management: An overview. In J. Couch, editor, *The Health Care Professional's Guide to Disease Management: Patient-Centered Care for the 21st Century*. Aspen Publishers, 1998.

K. Crammer and Y. Singer. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3:1025–1058, 2003.

E. Frank and M. Hall. A simple approach to ordinal classification. In L. D. Raedt and P. Flach, editors, *Proceedings of the 12th European Conference on Machine Learning (ECML-01)*, pages 145–156, Freiburg, Germany, 2001. Springer-Verlag.

J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

J. Fürnkranz. Round robin ensembles. *Intelligent Data Analysis*, 7(5), 2003. to appear.

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave and information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

V. Ha and P. Haddawy. Similarity of personal preferences: theoretical foundations and empirical analysis. *Artificial Intelligence*, 2003. To appear.

P. Haddawy, V. Ha, A. Restificar, B. Geisler, and J. Miyamoto. Preference elicitation via theory refinement. *Journal of Machine Learning Research*, 2003. To appear.

S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: A new approach to multiclass classification. In N. Cesa-Bianchi, M. Numao, and R. Reischuk, editors, *Proceedings of the 13th International Conference on Algorithmic Learning Theory (ALT-02)*, pages 365–379, Lübeck, Germany, 2002. Springer.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10 (NIPS-97)*, pages 507–513. MIT Press, 1998.

R. Herbich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In A. J. Smola, P. J. Bartless, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The Lumiere project: Bayesian user modeling for inferring goals and needs of software users. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in AI (UAI-98)*, pages 256–265, 1998.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, March 2002.

T. Joachims. Optimizing search engines using clickthrough data. In *KDD–2002, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, 2002.

H. Kautz, editor. *Recommender Systems: Papers from the AAAI Workshop*, Menlo Park, CA, 1998. AAAI Press. Technical Report WS-98-08.

S. Kramer, G. Widmer, B. Pfahringer, and M. DeGroeve. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, XXI:1001–1013, 2001.

P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapman & Hall, London, 1983.

A. Nakamura and N. Abe. Collaborative filtering using weighted majority prediction algorithms. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pages 395–403. Morgan Kaufmann, 1998.

A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML-00)*, 2000.

J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12 (NIPS-99)*, pages 547–553. MIT Press, 2000.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

P. Resnick and H. R. Varian. Special issue on recommender systems. *Communications of the ACM*, 40(3), 1997.

D. Riecken. Perzonalized views of personalization. *Communications of the ACM*, 43(8):26–29, 2000.

P. Savicky and J. Fürnkranz. Combining pairwise classifiers with stacking. Submitted for publication, 2003.

G. Tesauro. Connectionist learning of expert preferences by comparison training. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 1 (NIPS-88)*, pages 99–106. Morgan Kaufmann, 1989.

J. Wang. Artificial neural networks versus natural neural networks: a connectionist paradigm for preference assessment. *Decision Support Systems*, 11:415–429, 1994.

Active Learning of Ranking Functions

Klaus Brinker

International Graduate School of Dynamic Intelligent Systems
University of Paderborn
33098 Paderborn, Germany
kbrinker@uni-paderborn.de

Abstract. Active learning aims at controlling and reducing the need of labeled training data in supervised learning. Starting with only a small amount of labeled examples, the learner selects new training examples from a finite set of initially unlabeled examples, then requests their correct labels and incrementally learns the target function. While active learning with kernel machines has been successfully applied in classification to accelerate learning processes, we consider an extension to active learning of ranking functions. We employ a transformation process to state a ranking problem in terms of an equivalent binary classification problem. Analyzing active selection for binary problems yields a natural framework for active selection criteria in the case of more complex ranking problems. We study different strategies within this framework, and conduct an experimental study on synthetic data to investigate their predictive performance. As binary component learners, both support vector machines and (large scale) Bayes point machines are employed.

1 Introduction

The standard supervised machine learning model assumes that a completely labeled training set of examples is available to learn a target function. This implies that each example within the training set has to be assigned to the correct output value. However, in many practical applications, labelling examples cannot be performed automatically but involves human judgement or costly experimental measurements and is therefore time-consuming and expensive.

Pool-based active learning (selective sampling) [1] aims at controlling the labelling effort and accelerating the learning process: Starting with only a small amount of training examples, the algorithm selects new training examples from a finite set of initially unlabeled examples, then requests their correct labels and incrementally learns the desired function. The objective for the algorithm is to learn a target function at a certain level of accuracy using only a small amount of labeled training examples.

We consider the active learning framework in the field of kernel machines [2, 3], which have received ample treatment being both theoretically well founded and showing excellent generalization performance in practice. It has been shown empirically that active learning with kernel machines outperforms learning by randomly adding training examples in the field of character recognition [4], document classification [5, 6] and computational chemistry [7]. Whilst active learning in the field of kernel machines has been exclusively applied to classification problems so far, we present an extension to ranking problems. Prediction of rankings (preferences), i.e. total orders over a finite set of alternatives, is of particular importance in various fields of application such as personalized computer systems.

As in the case of multiclass classification there exist different approaches to reduce ranking problems to binary classification problems. As a straightforward generalization of one-against-one (multiclass) classification, ranking problems can be decomposed considering all pairwise preferences between two alternatives [8]. Each pairwise preference problem is treated independently as a binary classification problem and predictions are made by applying a voting scheme.

We adopt a different approach introduced in [9] to the expression of ranking problems in terms of single binary classification problems. While this approach is limited to linear classifiers as binary component learners, it is particularly suitable for generalizing binary active learning: All preference dependencies are encoded into a single binary problem which is amenable to well-studied binary active selection. Studying the transformation process, we derive a natural framework for active selection criteria in the case of ranking.

The remainder of this paper is organized as follows: In section 2, we recapitulate the transformation process to express ranking problems as single binary classification problems with special emphasis on the efficient kernelization of this approach. Based on this reduction, section 3 considers active selection of binary examples and naturally derives a framework of selection criteria for ranking problems. Computational complexity issues of this approach are discussed in section 4. The subsequent section presents experimental results on synthetic data of different selection criteria within our framework. As linear component learners, both support vector machines [2] and (large scale) Bayes point machines [10] are investigated. Finally, in section 6 we discuss further research issues.

2 General Setting

The subsequent section introduces the general class of ranking functions modelling the unknown preference concept. We closely follow [9] to transform a ranking problem into an equivalent binary classification problem. In general, any algorithm learning a consistent linear classifier can be employed to solve this classification problem. Expanding the binary classifier yields a solution to the initial ranking problem. We explicitly apply this transformation process to kernel machines both to increase the expressivity of the resulting ranking function and to make use of (typically) highly accurate classifiers.

Suppose we are given a training set

$$T = \{(x_1, y_1), \ldots, (x_m, y_m)\} \subset (\mathcal{X} \times \mathcal{S}^{(d)})^m$$

with $\mathcal{X}$ denoting a nonempty set and $\mathcal{S}^{(d)}$ being the symmetric group of degree d, i.e.

$$y_i = ([y_i]_1, \ldots, [y_i]_d) \quad \text{with} \quad \{[y_i]_1, \ldots, [y_i]_d\} = \{1, \ldots, d\}.$$

In other words, the objects to be learnt are full orders over a finite and a-priori fixed set of alternatives, represented as permutations.

To increase flexibility and expressivity, we embed patterns from input space $\mathcal{X}$ using a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The corresponding kernel feature space is denoted by $\mathcal{F}$ and the feature map by $\phi : \mathcal{X} \to \mathcal{F}$ [2]. We focus on the class of linear sorting functions within the kernel induced feature space $\mathcal{F}$ as our hypothesis space. In other words, we consider ranking functions having the following structure:

$$\begin{aligned} f : \mathcal{X} &\to \mathcal{S}^{(d)} \\ x &\mapsto \operatorname*{argsort}_{n=1,\ldots,d} \langle w_n, \phi(x) \rangle \end{aligned}$$

with $w_1, \ldots, w_d \in \mathcal{F}$ being normal vectors of hyperplanes and argsort returning a permutation of $\{1, \ldots, d\}$ where i precedes j if $\langle w_i, \phi(x) \rangle > \langle w_j, \phi(x) \rangle$.

Transforming the initial ranking problem both involves embedding the training data in a higher dimensional space and expanding single ranking examples into multiple binary classification examples. Consider a training example (x_i, y_i) which is mapped into feature space $\mathcal{F}$ by $x_i \mapsto \phi(x_i)$. We expand this example into $2(d-1)$ binary classification examples in $\mathcal{F}^d \times \{-1, +1\}$ in the following way:

Let $\overrightarrow{x}^{+}_{i,j}$ denote a vector in $\mathcal{F}^d$ with components

$$[\overrightarrow{x}^{+}_{i,j}]_n = \begin{cases} \phi(x_i) & \text{if } n = [y_i]_j, \\ -\phi(x_i) & \text{if } n = [y_i]_{j+1}, \\ 0 & \text{else.} \end{cases}$$

for $j = 1, \ldots, d-1$. $\overrightarrow{x}^{-}_{i,j}$ is defined analogously with a componentwise change of sign. Note that the components of $\overrightarrow{x}^{+}_{i,j}$ and $\overrightarrow{x}^{-}_{i,j}$ respectively are potentially infinite dimensional feature vectors.

Using this notion, we define a set T' of binary training examples corresponding to initial ranking examples

$$T' = \bigcup_{\substack{i=1,\ldots,m \\ j=1,\ldots,d-1}} \{(\overrightarrow{x}^{+}_{i,j}, +1), (\overrightarrow{x}^{-}_{i,j}, -1)\}$$

consisting of both $(d-1)m$ positive and negative examples.

Since mapping examples into the kernel feature space is performed implicitly, we cannot directly calculate dot products in hypothesis space $\mathcal{F}^d$ but have to express it in terms of the kernel. Suppose we are given two examples $(\overrightarrow{x}, \overrightarrow{y})$, $(\overrightarrow{x}', \overrightarrow{y}') \in T'$. Then, the components of $\overrightarrow{x}$ and $\overrightarrow{x}'$ can be expressed as

$$[\overrightarrow{x}]_n = \beta_n \, \phi(x_{i_n}) \quad \text{and} \quad [\overrightarrow{x}']_n = \beta'_n \, \phi(x_{j_n})$$

with $\beta_n, \beta'_n \in \{-1, 0, +1\}$. Hence,

$$\langle \overrightarrow{x}, \overrightarrow{x}' \rangle = \sum_{n=1}^{d} \beta_n \beta'_n \, k(x_{i_n}, x_{j_n}).$$

Note that there are at most two nonzero elements in this sum - independently of the number d of alternatives! Thus, by exploiting the sparsity of the expanded examples we can calculate dot products with at most twice the original computational cost as in input space $\mathcal{F}$. Similarly, using an indirect sparse encoding yields only a constant need of storage space for each expanded example.

Assuming that the kernel is chosen such that the expanded binary classification problem is linearly separable, we can train a kernel machine to calculate a discriminative linear classifier

$$\begin{aligned} g : \mathcal{F}^d &\to \{-1, +1\} \\ \overrightarrow{x} &\mapsto \operatorname{sign}(\langle \overrightarrow{w}, \overrightarrow{x} \rangle) \end{aligned}$$

which is consistent with the training set. The normal vector $\overrightarrow{w}$ can be expanded in terms of the $2(d-1)m$ training examples $\overrightarrow{x}_i$ with $[\overrightarrow{x}_i]_n = \beta_{i,n}\phi(x_{i_n})$ in T':

$$\overrightarrow{w} = \sum_{i=1}^{2(d-1)m} \alpha_i \overrightarrow{x}_i \quad \text{and} \quad [\overrightarrow{w}]_n = \sum_{i=1}^{2(d-1)m} \alpha_i \beta_{i,n} \, \phi(x_{i_n}).$$

Now, we decompose the above stated binary classifier into its d components to construct a ranking function:

$$\begin{aligned} f &: \mathcal{X} \to \mathcal{S}^{(d)} \\ x &\mapsto \operatorname*{argsort}_{n=1,\ldots,d} \langle [\overrightarrow{w}]_n, \phi(x) \rangle = \operatorname*{argsort}_{n=1,\ldots,d} \sum_{i=1}^{2(d-1)m} \alpha_i \beta_{i,n} \, k(x_{i_n}, x). \end{aligned} \tag{1}$$

This ranking function does not involve direct computation of the kernel feature map anymore since it is expressed exclusively in terms of the kernel function.

Suppose we are given an arbitrary example $(x_i, y_i) \in T$ and consider two succeeding components of y_i at position j and $j+1$. To show consistency with respect to this training example, we have to prove that

$$\begin{aligned} & \left\langle [\overrightarrow{w}]_{[y_i]_j}, \phi(x_i) \right\rangle > \left\langle [\overrightarrow{w}]_{[y_i]_{j+1}}, \phi(x_i) \right\rangle \\ \Leftrightarrow \quad & \left\langle [\overrightarrow{w}]_{[y_i]_j} - [\overrightarrow{w}]_{[y_i]_{j+1}}, \phi(x_i) \right\rangle > 0 \\ \Leftrightarrow \quad & \left\langle \overrightarrow{w}, \overrightarrow{x}^{+}_{i,j} \right\rangle > 0. \end{aligned}$$

The last equation holds since we assumed $g(\cdot) = \operatorname{sign}(\langle \overrightarrow{w}, \cdot \rangle)$ to be a consistent binary classifier for the transformed ranking problem.

In this section, we discussed a transformation process of a ranking problem into an equivalent binary classification problem with emphasis on the kernelization of this method. Thus, to train a ranking function any kernel machine can be utilized based on the expanded binary training set. Furthermore, we can deal with noisy, linearly nonseparable data in an elegant way by adding some constant $\nu > 0$ to the diagonal elements of the kernel matrix, $k(x_i, x_j) + \delta_{ij}\nu$, such that the training set becomes linearly separable [11].

3 Active Learning

Before deriving a framework for active selection criteria for ranking problems, we briefly recapitulate (pool-based) active learning in the case of a binary classification problem: Let us consider a linearly separable binary classification problem in feature space. The nonempty set

$$\mathcal{V} \overset{\text{def}}{=} \{w \in \mathcal{F} \,|\, \operatorname{sign}(\langle w, \phi(x_i)\rangle) = y_i \ \text{ for } \ i = 1, \ldots, n \quad \text{and} \quad \|w\| = 1\}$$

is called *version space* [12]. $\mathcal{V}$ consists of all (normalized) weight vectors corresponding to linear classifiers in feature space which separate the training set without errors. We can view learning as a search problem within version space. Each training example (x_i, y_i) limits the volume of the version space because to correspond to a consistent classifier a weight vector has to satisfy

$$\operatorname{sign}(\langle w, \phi(x_i)\rangle) = y_i \quad \Leftrightarrow \quad y_i \langle w, \phi(x_i)\rangle > 0.$$

In other words, consistent solutions can only lie on one side of the hyperplane with normal vector $\phi(x_i)$, depending on the class label y_i.

Suppose we are given an approximation of the center of mass of version space w_{center}, then it is a reasonable strategy to select that unlabeled example which corresponds to the restricting hyperplane in version space closest to w_{center}. Independent of the actual class label, the version space is reduced to approximately half the initial volume. In the empirical section, we consider two kernel machines, support vector machines [2] and (large scale) Bayes point machines [10], which approximate the center of mass of version space. It is easy to verify that this selection strategy is equivalent to choosing unlabeled examples closest to the classification hyperplane corresponding to w_{center}, i.e. we select examples minimizing $|\langle w_{\text{center}}, \phi(x_i)\rangle|$. Apart from the version space model which has been considered in [6], there are other theoretical justifications for this approach [4, 5].

Coming back to ranking problems, we introduce a generalization of the distance from the classification hyperplane by employing the transformation process discussed in section 2. Each ranking example is expanded into $2(d-1)$ binary classification examples. Therefore, a straightforward extension of the distance measure to ranking problems is to consider the minimal distance within the set of expanded binary examples:

Definition 1 (Generalized Distance). *Consider a linear sorting function*

$$\begin{aligned} f : \mathcal{X} &\to \mathcal{S}^{(d)} \\ x &\mapsto \underset{n=1,\ldots,d}{\operatorname{argsort}} \, \langle [\overrightarrow{w}]_n, \phi(x)\rangle \,. \end{aligned}$$

The distance δ of a ranking example (x, y) with respect to f is defined as

$$\begin{aligned}
&\delta : \mathcal{X} \times \mathcal{S}^{(d)} \to \mathbb{R}_{\geqslant 0} \\
&\qquad (x, y) \mapsto \min_{j=1,\dots,d-1} \left| \left\langle [\overrightarrow{w}]_{[y]_j}, \phi(x) \right\rangle - \left\langle [\overrightarrow{w}]_{[y]_{j+1}}, \phi(x) \right\rangle \right|.
\end{aligned}$$

In a similar fashion, one can define a generalized margin [9] which is maximized if a support vector machine is used as the component learner on the expanded binary training set. Both definitions reduce to the standard margin and to the standard distance from the classification boundary respectively for ranking problems with $d = 2$ (which can be considered as binary classification problems).

While in the case of binary classification ($d = 2$) this distance measure can be evaluated even without knowledge of the class label y,

$$\delta(x, y) = \delta(x, 1) = \delta(x, 2) = \left| \langle [\overrightarrow{w}]_1, \phi(x) \rangle - \langle [\overrightarrow{w}]_2, \phi(x) \rangle \right|,$$

this is not true in general ($d > 2$). However, it is possible to state a lower bound and an upper bound on the distance:

$$\begin{aligned}
\delta(x, y) &\geq \min_{\substack{n_1, n_2 = 1, \dots, d \\ n_1 \neq n_2}} \left| \langle [\overrightarrow{w}]_{n_1}, \phi(x) \rangle - \langle [\overrightarrow{w}]_{n_2}, \phi(x) \rangle \right| \qquad (2) \\
&\overset{\text{def}}{=} \delta^-(x).
\end{aligned}$$

and

$$\begin{aligned}
\delta(x, y) &\leq \max_{n=1,\dots,d} \langle [\overrightarrow{w}]_n, \phi(x) \rangle - \min_{n=1,\dots,d} \langle [\overrightarrow{w}]_n, \phi(x) \rangle \qquad (3) \\
&\overset{\text{def}}{=} \delta^+(x).
\end{aligned}$$

It is easy to see that these bounds are tight in the sense that for every x there exists a y such that equality holds in (2) or (3) respectively.

As stated above, distance based active selection of new training examples has been successfully applied in binary classification. Class labels of examples with minimal distance from the classification boundary are requested and then these examples are added to the current training set. However, since distances cannot be evaluated for ranking examples in general, there is no straightforward generalization of this selection strategy.

Nevertheless, instead of exactly evaluating distances we are able to employ criteria that approximate this quantity. The bounds δ^- and δ^+ provide a framework for reasonable approximate distance based selection strategies. Furthermore, both δ^- and δ^+ can be considered as approximate distance measures at the extreme points of the spectrum. In the following, we investigate active selection of ranking examples based on δ^- and δ^+, denoted by *distance*$^-$ and *distance*$^+$. In other words, unlabeled examples minimizing δ^- (δ^+) are selected to be labeled and subsequently added to the current training set.

In the beginning with the number of labeled training examples being small, our intermediate models are expected to perform poorly in terms of generalization accuracy. As a consequence, δ^- is likely to underestimate the true distance of an unlabeled example. Similarly, when our models get more accurate δ^+ is likely to overestimate the true distance.

To overcome this problem, we consider a hybrid strategy starting with the δ^+ strategy which switches to δ^- in a data dependent fashion. When employing support vector machines as the binary component learner, the minimum distance of a labeled example to the classification boundary is normalized to 1. Furthermore, only those unlabeled example with a distance less than 1 to the boundary are guaranteed to intersect with the version space. Therefore, when selecting an unlabeled ranking example x for which $\delta^+(x) \geq d$ holds, labeling this example might not put further restrictions on the version space. Based on this observation, we suggest switching to the δ^- strategy whenever there is no unlabeled example with $\delta^+(x) \leq d$. Since the reasoning above does solely hold for support vector machines, we consider this hybrid strategy only in the case of support vector machines.

4 Computational Complexity

Consider a ranking problem consisting of m training examples with d preferences in a standard supervised batch learning setting. In section 2, we discussed a technique which has been proposed in [9] to reduce a ranking problem to a single binary classification problem. This transformation process relies on an expansion of the ranking training set into $2(d-1)m$ binary classification examples and a mapping into the higher dimensional space $\mathcal{F}^d$.

However, as stated above, by using an indirect sparse encoding it is not necessary to explicitly increase the dimension of the problem. Thus, independently of d, calculating dot products is at most twice as expensive as in input space $\mathcal{F}$ in terms of computational time, and only a constant amount of storage space for each expanded example is necessary (if we preserve a copy of the original training set). Therefore, the computational complexity when using a kernel machine with running time $\mathcal{O}(n^\alpha)$ in the number of training examples amounts to $\mathcal{O}((2dm)^\alpha)$. Since the running time of support vector machines is empirically estimated at $\mathcal{O}(n^2)$, learning a ranking function is of order $\mathcal{O}(d^2m^2)$ in this special case. Note that the pairwise decomposition technique proposed in [8] requires computational time of equal order. Finally, predicting a ranking is of order $\mathcal{O}(dm)$ independently of the kernel machine as can be easily derived from (1).

Evaluating the active selection criteria *distance*$^-$ and *distance*$^+$ for a single example is of comparable computational complexity as prediction. Therefore, actively learning m examples from a set of n examples requires computational time of order $\mathcal{O}((2d)^\alpha\, m^{\alpha+1} + ndm^2)$.

5 Empirical Evaluation

5.1 Experimental Setting

To evaluate the efficiency of the *distance*$^-$ and *distance*$^+$selection strategy we have conducted a number of experiments using both support vector machines and Bayes point machines as linear component learners. Due to the lack of suitable real-world datasets, we decided to generate synthetic data using a setting described in [8] from expected utility theory. In a nutshell, the data generation process works as follows: Fixing the number of input features n and the number of preferences d, we generate a $d \times n$ (utility-) matrix U with independently and uniformly distributed entries $U_{ij} \in [0, 1]$. To produce a set of ranking examples, we independently draw features of examples from a uniform distribution over $\{x \in \mathbb{R}^n \mid x \geq 0,\, x_1 + \cdots + x_n = 1\}$. The corresponding rankings $y \in \mathcal{S}^{(d)}$ are generated according to the preference relation $\succ$ induced by $E(i) = \sum_{j=1}^{n} x_j \cdot U_{ij}$, i.e.

$$i \succ j \quad \Leftrightarrow \quad E(i) > E(j).$$

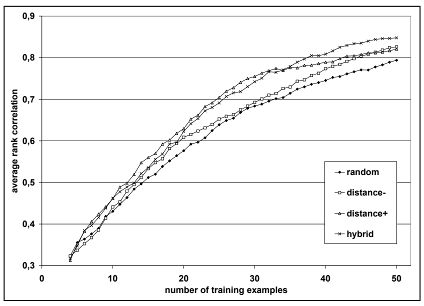
Obviously, this corresponds to a noise-free scenario in our learning model since given a feature vector x an alternative way to express the corresponding ranking is $y = \mathrm{argsort}_{j=1,\ldots,d} \langle x, u_j \rangle$ with u_j denoting the j-th row vector of U.

In all experiments, we fixed the number of input features $n = 10$. For $d \in \{5, 10\}$ we generated 5 different datasets consisting of 2000 examples, each dataset originating from a different matrix U. Each dataset was randomly split 10 times into a training set and a test set of equal size. Active selection of new training examples was restricted to the training set, while the Spearman rank correlation coefficient was evaluated on the test set after each iteration (selection and training). The results were averaged over all splits and 5 datasets. We started with a randomly selected training set of size 4 in all experiments.

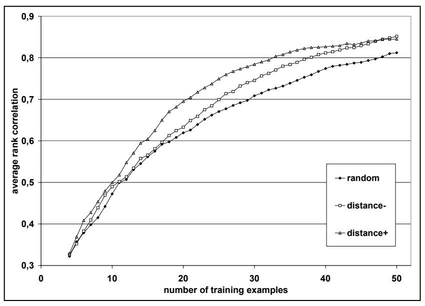
Active selection of new ranking examples and training was performed using the *distance*$^-$, *distance*$^+$ and a hybrid strategy based on two different underlying kernel machines, support vector machines [2] and (large scale) Bayes point machines [10]. In all experiments involving support vector machines we fixed the penalty parameter $c = 1000$, and when using Bayes point machines we always averaged over 100 kernel perceptrons. For both kernel machines we used linear kernels. Furthermore, we considered the random selection of new examples, which served as a baseline strategy.

5.2 Experimental Results

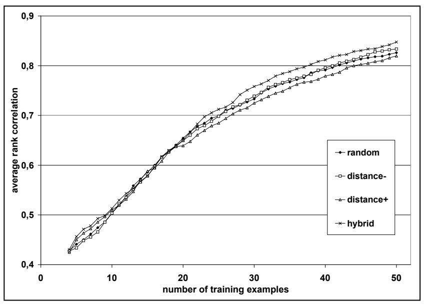
In all experiments with $d = 5$ (see figures 1(a) and 1(b)), *distance*$^-$ and *distance*$^+$ based on both support vector machines and Bayes point machines outperform random selection after only a few iterations. For $d = 10$ (see figures 1(c) and 1(d)), active selection seems to be more challenging. The *distance*$^-$ strategy still shows slight advantages over random selection after 50 steps, while *distance*$^+$ is clearly outperformed by random selection except for slight advantages at the



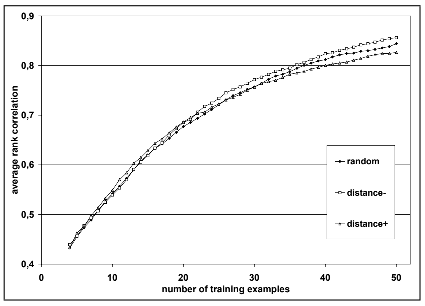
(a) Support vector machine ($d = 5$)



(b) Bayes point machine ($d = 5$)



(c) Support vector machine ($d = 10$)



(d) Bayes point machine ($d = 10$)

beginning of the learning process. This observation suggests that with the number of alternatives growing *distance*$^+$ is too coarse a measurement for selection. Furthermore, as predicted on theoretical grounds, *distance*$^-$ is too optimistic at the beginning of the learning process where our intermediate models are not very accurate. In contrast to this, the advantage of *distance*$^+$ is restricted to the beginning, with *distance*$^-$ reaching the same level of accuracy after circa 45 ($d = 5$) and 20 ($d = 10$) iterations respectively.

Furthermore, in the case of support vector machines, we investigated a hybrid strategy which starts with the *distance*$^+$ strategy and then switches to *distance*$^-$ in a data-dependent fashion (see section 3). While both *distance*$^-$ and *distance*$^+$ seem to have some advantages and disadvantages at certain stages of the learning process, the hybrid strategy combines the strengths of both methods in our experiments (see figures 1(a) and 1(c)).

In theory, Bayes point machines are able to approximate the center of version space more accurately than support vector machines. This property is reflected in our experiments with Bayes point machines achieving a higher level of accuracy in terms of rank correlation.

6 Conclusions and Future Research

We applied a transformation process to express a ranking problem in terms of an equivalent binary classification problem. Considering active selection in binary classification provides a natural framework for active selection criteria in the case of the more complex class of ranking problems. While we merely investigated two simple strategies on the basis of synthetic data within this framework, our analysis might serve as a starting point to develop more sophisticated ones. Preliminary results of a hybrid strategy are promising and indicate that it is possible to combine the strengths of both strategies. Furthermore, we discussed how to efficiently employ kernel machines as binary component learners both increasing the expressivity of this model and providing an elegant technique to deal with noisy data. In practice, data might contain incomplete preference information. However, a slight modification of the expansion step can incorporate this setting. Therefore, we plan to explore the full potential of using kernels and incorporating incomplete preference information on real-world data.

Acknowledgments We would like to thank Eyke Hüllermeier and Hans Kleine Büning for fruitful discussions and suggestions. This work was supported by the International Graduate School of Dynamic Intelligent Systems, University of Paderborn.

References

1. David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In W. Bruce Croft and Cornelis J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.
2. B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, 2002.
3. Ralf Herbrich. *Learning Kernel Classifiers.* MIT Press, 2002.
4. C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 111–118, 2000.
5. G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.
6. S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 999–1006. Morgan Kaufmann, San Francisco, CA, 2000.
7. M. K. Warmuth, G. Rätsch, M. Mathieson, J. Liao, and C. Lemmen. Active learning in the drug discovery process. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural information processings systems*, volume 14, 2002.
8. J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. Technical report, Austrian Research Institute for Artificial Intelligence, 2003.

9. Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification: A new approach to multiclass classification and ranking. In *Advances in Neural Information Processing Systems 15 (NIPS)*, 2002.
10. Ralf Herbrich and Thore Graepel. Large scale bayes point machines. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13 (NIPS)*, pages 528–534. MIT Press, 2001.
11. John Shawe-Taylor and Nello Cristianini. Further results on the margin distribution. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 278–285. ACM Press, 1999.
12. Tom M. Mitchell. Generalization as search. *Journal of Artificial Intelligence*, 18:203–226, 1982.

Preference Learning in Internet Collaboration Environments

Thomas Friese and Bernd Freisleben

Department of Mathematics and Computer Science, University of Marburg
Hans-Meerwein-Str., D-35032 Marburg, Germany
{friese,freisleb}@informatik.uni-marburg.de

Abstract. In this paper, we present an overview of preference learning applications to support collaborative work on the internet. Based on deriving some basic requirements a more generalized and extensible internet collaboration environment should satisfy, a system architecture for the flexible integration of different web applications to support a workgroup in different everyday aspects of their internet based work is proposed. By using WebService technology, the system implements such web applications by *orchestrating* local or remote components. Adoption of the WebService paradigm also increases the flexibility in the deployment and extensibility of the system. By defining an extensible component for preference learning, the system supports the integration of recommendation techniques into web applications.

1 Introduction

The World Wide Web has become an essential source of information and an environment for e-commerce. Using email as a primary means of communication is commonplace, and we spend a substantial share of our time processing email messages. More and more business applications that have traditionally been implemented as specialized client applications (perhaps in a client/server-environment) are implemented as intranet- or even internet- applications that are accessible to the user via a standard web-browser. Leveraging the experience of a workgroup and collaborating in these tasks is an obvious way of increasing productivity.

With the growing success of online shopping websites, a lot of attention from both the industrial world as well as the academic world went into recommender systems. These systems usually focus on enhancing the usage experience of the visitor of a particular website by predicting his/her future browsing direction or product preference. This is usually done by exploiting his/her past behaviour or clustering users to derive recommendations from selections of other members of that group. (The “customers who bought this book also bought:” recommendation of amazon.com is a common example for such a recommender system.)

The acceptance of HTTP as common transport protocol for information on the internet and XML based markup languages has led to the development of the

WebService standards. By transferring SOAP[1] messages via HTTP connections one can carry out remote procedure calls on a functional entity that exposes its interface described using WSDL[2] to the Web. The adoption of the WebService paradigm is leading to a different way of application development. WebServices can be seen as functional components that may be used to compose or orchestrate more complex applications.

In this paper, we review related work in the field of preference learning applied to internet collaboration (e.g. collaborative filtering, annotating resources) and derive requirements a generalized internet collaboration environment should satisfy. Based on these requirements, we present a proposal for the design of a generalized framework that supports the creation of flexible and adaptive applications to support group collaboration; flexibility and adaptivity is achieved by relying on the WebService paradigm to assemble an application from distributed components. Our main goal is to develop a platform that enables non-intrusive support to user communities, helping with the usual workflow rather than radically changing it. Therefore, our system does not require changes to the client software and integrates into heterogeneous computing environments without forcing clients to a particular operating system. We chose a proxy design that enables us to catch user interaction with the internet. This choice allows us to weave additional information and user interfaces into the replies of user requests while not breaking the medium the user interacts with. A second building block of our system is a generic and extensible component that encapsulats methods for preference learning and experience management. The focus of our work is on integration of learning methods into a collaboration environment, not on the development or evaluation of learning methods.

This paper is organised as follows. Section 2 gives an overview of related work in the field of preference learning in an internet environment. In Section 3, we present our design of an internet collaboration environment that leverages group experience not only in simple web browsing but also in more advanced web applications. In Section 4 we discuss some use cases and sample applications to be built upon our framework. Conclusions and directions for future work are presented in section 5.

2 Related Work

In this chapter, we give an overview of related work from the perspective of preference learning in an internet context and derive a number of requirements for an advanced internet collaboration environment. Most applications facilitating preference learning in an internet context can be characterized as recommender systems or collaborative filtering systems. They are intended to support users in gathering information or classifying items, singling out the ones most interesting to the user.

[1] Simple Object Access Protocol
[2] Web Service Description Language

2.1 Recommender Systems

A number of applications that try to capture a users preferences for the purpose of suggesting other items that might be of interest to the user have been implemented in the past and in general are referred to as *recommender systems.* Different techniques for recommendation have been proposed, based e.g. on features of the items/content, historical data about user decisions, demographic data or a combination of different techniques as in *hybrid* recommender systems [1, 2]. A classification of recommendation techniques for a set of items and users is presented by [1], where the most important techniques can be described as:[3]

- *Collaborative* recommendation takes a number of ratings of items and groups similar users to extrapolate from their ratings of other items (it is often referred to as *collaborative filtering*).
- *Content-based* recommenders use features of the items to generate a classifier that fits a users ratings of items and apply this classifier to other items.
- *Knowledge-based* recommenders take a description of a users needs as well as the features of items into account, employing knowledge of how these items meet a users need to infer a match.

Even though every recommendation process can be seen as some kind of inference, adding knowledge about how an item meets a users needs introduces additional value by enabling a more advanced inference. Recent work [4, 3] argued to integrate ontologies and languages with a well defined semantic data model to improve the preference learning process and make the results more expressive. According to the authors of these papers, the expectation of acquiring better results by learning from RDF[4] marked-up instances instead of plain-text data was not supported by the results, but using RDF reduced the computational complexity and lacked ontological support for the data and relations between instances [3].

Therefore, a generalized collaboration system to support preference learning should at least be able to capture user ratings and features associated to the items of interest. Additionally, it should allow for the seamless integration of ontological knowledge and processes, taking advantage of this formal and knowledge rich information.

2.2 Collecting Preference Data

A question arising in the design of recommender systems is the way in which user preferences will be captured. In general, there is the choice between making use of *implicit* vs. *explicit* expressions of preference by a user. This choice extends to the collection of relevance feedback about recommended items to the system. The problem of collecting preference data is closely related to the *new-user* problem

[3] Other recommendation techniques include using demographic data about users and using a utility-function representing users preferences.

[4] Resource Description Framework

faced by collaborative filtering (CF) systems. As preferences are predicted based on the similarity of users, the system has to elicit initial preference values for a new user to give reasonable recommendations. Strategies to overcome this problem have recently been proposed [11, 2].

The implicit capturing of user preference is often addressed by mining web usage data (i.e. requests of a search engine, log files of a web server) [5, 6]. Using these techniques, one can discover common sequences of web usage or improve search results by grouping URLs matching certain keywords. Furthermore, they take the users choice of presented search results into account for improving future results [7]. The work most closely related to our approach presented in the next section uses a proxy-based zero-input interface with non-intrusive presentation of page rankings, based on access statistics and user profiles for registered users [12]. The ranking of pages is solely based on statistical data about the number of visitors (and the number of experts thereof) to a page and the total number of visits to the page.

The explicit capturing of user preferences involves the user to actively state his/her preference of an item A over item B or to assign a value indicating the usefulness of an item or a recommendation. An early system to rely on the explicit feedback on items that a user experienced is GroupLens [8]. In a system taking content into account, this might require the user to actively annotate items with metadata that can be used by the system to infer semantic relationships and relevance of items [9].

While gathering explicit ratings of a users likes and dislikes provides a CF system with better data to derive recommendations from, users tend to be reluctant to take the burden of explicitly rating items.

This leads to the conclusion that: *A generalized collaboration environment should provide means to implicitly as well as explicitly collect preferences and relevance feedback. Ideally, the mechanisms themselves should be adaptive to the operational context, to elicit as much information from the user as possible in a non-intrusive way.*

2.3 System Integration

As stated before, preference learning has mostly been employed as an integral part of other web applications (e.g. the well known product recommenders in web based shopping systems). Obviously, the learning and deduction mechanisms can easily access a local database of items that comprise their domain of recommendation. It is also easy for the application developer to integrate facilities for rating collection as well as result presentation into the single user interface. Preference data is usually captured and evaluated in a single server that is also the point of interaction with the user.

A different approach of system design in collaboratively filtering internet information (i.e. web pages) is taken by gathering preference information in a proxy environment. In this case, preference data is collected and processed apart from the original source of information in a proxy server [12]. Preferences learned from the collected usage information is later on used to recommend the most valuable search result and is nonintrusively woven into the google[5] search result pages. Another system based on a proxy architecture is aimed at supporting cooperative browsing [13] by showing group members the pages that other users logged into the proxy currently visited. It did not employ techniques to learn user preferences.

A third place for application integration of preference learning or recommendation systems is the internet client software. Some personal information scouts have been implemented as personal recommendation agents extending a web-browser. Supporting collaboration imposes the need to enable user agents to communicate and probably exchange preference information or otherwise implement some distributed machine learning algorithms. An algorithm and protocol to support collaborative filtering in a distributed environment has recently been proposed for privacy reasons [14]. Since collected preference information allows a centralized system to gain intimate knowledge about a users habits, the author favors a scheme where every user keeps his/her preference data protecting his/her privacy.

In the previous subsection we presented different grades of explicitness of preference collection. This is affected by the choice of system integration. While a solution integrated into a web application does not, or only to a very limited extent, capture preference in using other web applications, a system integrated into the client software is capabale of accurately tracking a user's actions.

In conclusion, we derive that: *A general internet collaboration environment should be as flexible as possible, with regard to the internet information sources and applications it can be used with. Furthermore, it should be as minimaly invasive to a user's habits or software as possible.*

3 System Architecture

To summarize the findings of the previuous section, the requirements for a flexible internet collaboration environment that adapts to the users' needs by learning their preferences are:

- Extensible storage of preference assertions and features related to the items of interest. The seamless extension towards integration of more knowledge rich data should be supported.
- The system should be flexible in the way and explicitness of the processes to collect preference and relevance feedback.
- The collaboration environment should be usable with as many internet resources as possible.
- To support users rather than distract them, the collaboration framework should integrate into their software environment with the least possible required changes and should also aim at integration into the usual workflow of accessing resources.

[5] http://www.google.com

The system architecture we propose to satisfy these requirements can roughly be divided into two main building blocks as shown in figure 1: A *proxy* component, that handles the interaction with the user and an *analysis* component, that stores instance data and encapsulates learning and analysis functionality. This separation of components allows the analysis component to reside on a different host than the proxy. Instances of the proxy could, for example, run on every client machine and use a central analysis component.

The WebService paradigm is central to the design of our system, as it allows for a very flexible implementation of applications composed of loosely coupled entities. Using remote procedure calls with SOAP messages allows us to access system components as well as remotely deployed components of third parties (e.g. a specialized translation service, or an advanced information extractor). The application of WebService communication in the system is shown by dashed lines in figure 1.

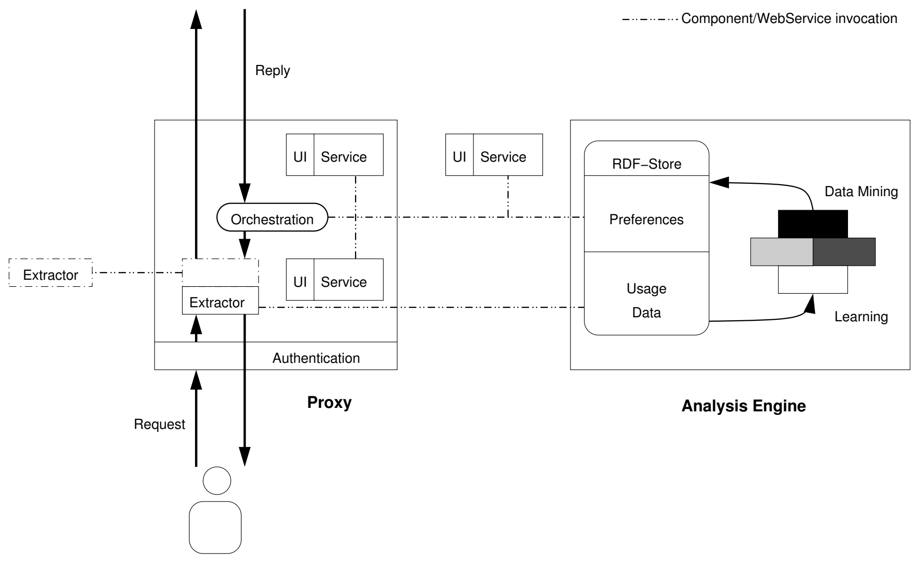


Fig. 1. Schematic view of the internet collaboration environment

To use the system, a user configures his/her web browser to use the proxy. As a means of identifying a user, we require a client to authenticate itself via the standard proxy authorization mechanism [17]. Disregarding all additional functionality of the system, it then works like an ordinary web proxy. A user's request is forwarded to the remote server, then the proxy delivers the reply to the client. In the following subsections, we focus on the extractor and application integration components of the proxy as well as some aspects of the analysis engine.

3.1 Extraction of Instance Data

In the process of answering requests, web usage data is collected. This collection is performed by the request extractor. The target of a request, its originator and the time of its occurence are collected as the basic information. Other extractor components may optionally be used to extract further information (e.g. the values of cgi attributes from requests occuring as submission of HTML form data) or to regard more complex constraints in the extraction process (e.g. the context of a user).

To allow content-based learning, the system needs a facility to extract content features. The extraction of features is even more closely coupled to the wide variety of possible content types than in the case of a users requests. The extraction of features may also require more sophisticated methods a provider might not want to disclose. Integrating such components as WebService instances allows the system to use remote methods to extract features. The external extractor on the left in figure 1 could be located at some other provider's computer.

3.2 Application Integration by WebService Orchestration

Applications to support a group of users in internet collaboration are implemented by orchestrating basic WebService components into more complex process flows. This is done by an orchestration or flow engine that takes the description of a compound service as input and executes the resulting, more complex application. For its semantic expressivenes and to examine reasoning about the processes, flows are specified using the DAML-S [16] language and are also stored in the analyser's RDF-store.

The user interacts with the WebServices using dedicated web applications bound to the services for presentation[6]. Natural services of the system have their user interface (UI) bound to the deployed service, while UI components for remote services have to be deployed in the system, in order to access third party services that bring no own UI. The reply of a server is processed by the proxy, allowing the orchestration engine and individual services to alter its contents or - in other words - *weave* its own user interface into the original content. This mechanism enables an application to integrate references or results of execution into the reply that is presented to the user (e.g. as hyperlink or tooltip[7] menu).

3.3 Analysis Engine

The second large component of our system architecture is the analysis engine or analyser (on the right) that stores and processes instance data and results of the learning process. Usage data is collected by storing assertions about resources

[6] Even though WebServices are accessible via the HTTP protocol and XML based SOAP language, the interface is rather machine readable than suited for human interaction.

[7] A popup that appears to the event of the mouse hovering over a page element.

involved. RDF was specified by the W3C with the purpose of storing assertions about internet resources. Furthermore, it is the basis for more expressive languages like DAML+OIL [15] and DAML-S. Being able to extend the system with semantically founded learning methods led to the choice of a RDF storage facility as central data repository. The analysis component exposes itself as a WebService, to enable remote access to its functionality.

Internally, the RDF store is used by an extensible collection of learning and data mining components. Extracted preferences are stored to be retrieved by the orchestration engine that uses this data in order to personalize the collaboration applications. While we specified the analysis engine to be extensible, currently we focus on the integration of preference learning algorithms in the collaboration environment rather than developing own algorithms.

4 Use Cases and Applications

This section will introduce some use cases and sample applications currently being using our framework. We have chosen sample applications we believe to be particular useful to research and development groups, as we intend to use them in our own productivity environment. The scenario consists of a group of users accessing static resources (HTML pages) as well as dynamic resources (web applications) on the internet. The participants of the system also communicate vie email. The sample application should support the group of users in their tasks, preferably leveraging experience of other users.

4.1 Bookmark Recommendation and Annotation

Keeping bookmarks for interesting or useful web addresses is well known and supported by almost every web browser. Storing and synchronizing them against a repository that can be accessed from different locations (e.g. other computers, other users' accounts) is poorly integrated into and not often supported by standard web-browsers. The bookmarking process involves the active recognition of a valuable resource and some explicit action to add its URL to a local or shared repository. Our goal is to support the user group in the collection of such a repository.

Repeated visits to a resource are an indicator for the relevance of a resource. The collaboration system may therefore collect web usage information per user and recommend the addition of a repeatedly visited address to the repository. Determining often visited resources may be performed by analyzing the web usage information of the entire group.

The systems asks a user for explicit confirmation, prior to adding a resource to the repository, for various reasons:

- Resources can be manually categorized or the automatic (e.g. content based) categorization can be confirmed by a user.
- Even regularly accessed resources might hold no value for the repository, therefore the quality of the repository can be improved.

– A user may decide not to publish his/her usage information for privacy reasons.

The entire process is shown in figure 2: Web usage information (a) is collected by the extractor component (EX) in the proxy and stored in the analyser's data repository (b). Regular analysis of the data leads to recommendations (c) that are presented to the user via a summarizing web page, summarizing email (di) or direct recommendation woven into the reply (dii). Apart from recommending the addition to the repository, an explicit command may be woven into every page by the WebService orchestration engine (OE). In addition to statistical access information, the proxy is able to use another extractor to collect content information of a resource or associated meta data out of the reply by the remote server (e) and also store this information in the repository (b). Therefore, the system is able to learn a user's preferences based on contents. A recommended bookmark is added to the bookmark repository (g) (i.e. a centralized stand-alone WebService that is used by the collaboration environment) upon explicit confirmation by the user (f).

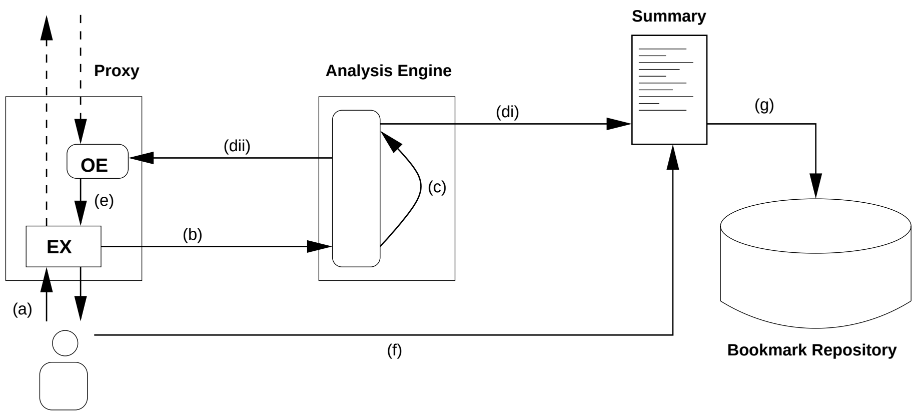


Fig. 2. *Bookmark recommendation*: (*a*) request, (*b*) collection of usage information (*c*) analysis leads to recommendation (*d*) presentation via *i*) summarizing web page/email *ii*) embedding, (*e*) collection of content features, (*f*) confirmation by user, (*g*) addition to repository.

The collaboration environment may as well be used to annotate [10, 9] web resources. A web application supporting the creation of RDF based annotations is embedded into web pages by the orchestration engine (OE). This application uses the data repository of the analyser component to store RDF based annotations to a resource. The annotations may be used to improve the web usage analysis performed by the analyser component, as annotations are additional content features of a resource.

Various aspects of this simple application hold the potential for dynamic adaptation to a user's preferences or context: The method of presentation (i.e. summarizing web pages, summarizing email or embedding recommendation facilities into replies), the choice of a particular repository WebService, the analysis features (e.g. integration of content features) or even the activation of the application. Those configuration features may be monitored and analysed by the system itself, allowing the system to learn a user's (or a group of users') configuration preferences. For example, the system may analyse how frequently a user confirms his/her recommendations or uses an embedded application to deactivate the component in question. On the contrary, the system may find similar users to access a newly introduced application and activate it as a recommendation.

4.2 Spam Filtering and Email Classification

The use of a proxy is not limited to the HTTP protocol. As a second domain for a sample application we have chosen email. The increasing flood of unsolicited bulk email (spam) has led to the integration of different spam filtering mechanisms into mail servers and client applications, based e.g. on bayesian classifiers. Figure 3 shows the integration of a spam filtering application into the collaboration environment.

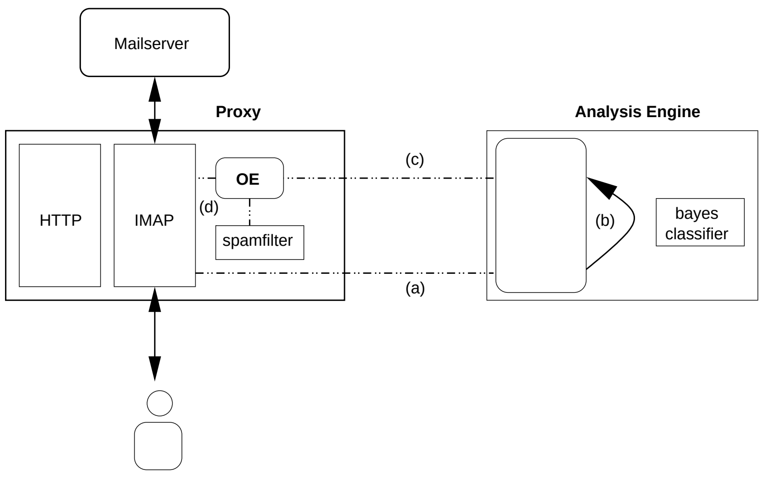


Fig. 3. Spam filtering, email classification is implemented by a second IMAP proxy, reusing the analyser and orchestration components.

Using an IMAP[8] proxy, the system is able to monitor the user interaction with a mail server (to move a message from the inbox to a mail folder called

[8] Internet Message Access Protocol

"spam" may be such an action). The actions taken by the user are stored in the analyser repository (a) as well as content information that may be gained by inspecting individual messages. The clustering functionality integrated into the analysis component may then be used to process this interaction and content information and then leads to the classification of a new message (e.g. by an integrated bayesian classifier) or an action recommendation (b). The spamfilter, run by the orchestration engine (OE) may use these recommendations or classifications (c) to issue commands to the mail-server via the IMAP proxy (d). Similarly, a content based clustering of messages could be used to sort messages into different mail folders. The system may leverage the actions of many users, learning spam classification on a larger sample base.

5 Conclusions and Future Work

In this paper, we have presented an overview of preference learning applications to support work on the internet, especially collaborative work. We have derived some fundamental requirements that a more generalized and extensible internet collaboration environment should satisfy, and have proposed design of such a system. Our system allows for the flexible integration of different web applications to support a workgroup in different everyday aspects of their internet based work. By using WebService technology, the system implements such web applications by *orchestrating* local or remote components. Adoption of the WebService paradigm also increases the flexibility in the deployment of the system and the extensibility thereof. By defining an extensible component for preference learning, the system supports the integration of recommendation techniques into its web applications.

Future work includes the extension of this implementation to gather experimental results for its usability and the performance questions raised by including SOAP calls. Furthermore, we will focus on the orchestration of compound service processes and the application of preference learning to the orchestration process, as already sketched in the bookmark scenario.

References

1. Burke, R., Hybrid Recommender Systems: Survey and Experiments, To appear in: User Modeling and User-Adapted Interaction, 2003
2. Carenini, G., Smith, J., Poole, D., Towards more Conversational and Collaborative Recommender Systems, *in Proceedings of The 2003 International Conference on Intelligent User Interfaces, pp. 12-18, 2003.*
3. Grimnes, G. A., Learning Knowledge Rich User Models From The Semantic Web, *in Proceedings of Doctoral Consortium, Proceedings of the 9th International Conference on User Modeling, 2003.*
4. Middleton, S. E., et al., Exploiting Synergy Between Ontologies and Recommender Systems, in Proceedings of The Eleventh International World Wide Web Conference (WWW2002), Semantic Web Workshop, 2002.

5. Beeferman, D., Berger, A., Agglomerative Clustering of a Search Engine Query Log. *in Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 407-416, 2000.*
6. Mobashar, B., Cooley, R., Srivastava, J., Automatic Personalization Based on Web Usage Mining, *in Communications of the ACM, vol. 43 (8), pp. 142-151, 2000.*
7. Joachims, T., Optimizing Search Engines Using Clickthrough Data, *in Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining (SIGKDD 2002), pp. 133-142, 2002.*
8. Resnick, P., et al., GroupLens: An Open Architecture for Collaborative Filtering of Netnews, *in Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work, pp. 175-186, 1994.*
9. Karger, D. R., Katz, B., Lin, J., Quan, D., Sticky Notes for the Semantic Web, *in Proceedings of the International Conference on Intelligent User Interfaces, pp. 254-256, 2003.*
10. Kahan, J. and Koivunen, M., Annotea: an open RDF Infrastructure for Shared Web Annotations, *in Proceedings of WWW10, pp. 623-632, 2001.*
11. Rashid, A. M., et al., Getting to Know You: Learning New User Preferences in Recommender Systems, *in Proceedings of The 2002 International Conference on Intelligent User Interfaces, pp. 127-134, 2002.*
12. Sharon, T., Lieberman, H., Selker, T., A Zero-input Interface for Leveraging Group Experience in Web Browsing, *in Proceedings of The 2003 International Conference on Intelligent User Interfaces, pp. 290-292, 2003.*
13. Cabri, G., Leonardi, L., Zambonelli, F., Supporting Cooperative WWW Browsing: a Proxy-based Approach, *in 7 th Euromicro Workshop on Parallel and Distributed Processing, pp. 138-145, 1999.*
14. Canny, J., Collaborative Filtering with Privacy, *In IEEE Symposium on Security and Privacy, pp. 45-57, 2002.*
15. Darpa Agent Markup Language, `http://www.daml.org`
16. Ankolenkar, A., et al., DAML services, `http://www.daml.org/services/`
17. Fielding, R., et al., RFC 2616 - Hypertext Transfer Protocol - HTTP/1.1

Weighting Predictors in Memory-based Collaborative Filtering

Jan-Hendrik Dörner

Falkstrasse 38b,
60487 Frankfurt am Main, Germany
`doerner@Project24.com`
`http://www.Project24.info`

Abstract. Memory-based collaborative filtering is a popular technique for personalizing content with heterogeneous preferences on the web intelligently. Most of these algorithms are based on its first, published by Resnick et al. in 1994. Unfortunately their algorithm has so far only an empirical justification.
This paper shows how two linear models can be the theoretical basis of memory-based collaborative filtering algorithms and explains why the performance measurement is an essential part thereof. It reveals the (here) existing dependency of generating individual predictions and weighting them.
The two models and two techniques to estimate them are compared using the EachMovie dataset. The setup for the test chosen retained the structure of the dataset. It was found that the benchmark algorithm of Herlocker et al. [1999] was improved significantly by considering the general preference of a product.

1 Introduction

The telecommunication industry spent several hundred billions of dollars for acquiring mobile licenses. These high spendings cannot be justified by future charges for regular telephone calls alone, but only by additional earnings from mobile data services. Since it is hardly possible to display the relevant information much more compact than on the web, the limited display of the mobile devices force them to get personalized. One method of personalization is called *collaborative filtering*.

Its basic assumption is that people share preferences and behavior [9]. Therefore, the preference or action of a particular user might be predicted well by other *similar* users. Since this kind of prediction is only based on comparing preferences and behavior, it can be used well for predicting complex products (such as books, jokes, movies, music or wine) where key product attributes are hard to identify and the preferences are heterogeneous, so that no single opinion (even that of an expert) can possibly lead to an ultimate advice for everyone.

2 Measure of performance

Personalization does not come for free. Therefore, (almost only) corporations use it on the web today and only when they hope to make money out of it. The *best* algorithm of personalization — from a corporations perspective — therefore maximizes the return on investment. Unfortunately the monetary return can often hardly be measured so that an appropriate *proxy* has to be used instead.

Several different measurements have been proposed to measure the quality of an algorithm [12].[1] As long as they are not totally equivalent, the best algorithm will depend on the measurement chosen. Therefore it is crucial to know the optimization criteria (i.e. the measurement) *before* developing an algorithm. For example, if the only focus is to minimize the calculation complexity and data requirement under maximum coverage[2], a good, but not necessarily accurate algorithm could simply return a constant or random profile. Since the criteria just mentioned are generally in direct conflict to the prediction error, an appropriate measurement should balance them all reasonably.

3 Model

The model presented in this paper is designed to predict preference values. In contrast to predicting behavior (which is most often of binary nature), memory-based[3] algorithms have shown to predict here more accurately than model-based[4] ones [2, 7]. A reason for this unexplained phenomenon will be revealed till the end of this section.

The focus in this paper is on minimizing the squared amount by which the estimated preferences differ from their real ones *(mean squared (prediction) error, MSE)*[5], while using only preference data and having no lower coverage nor higher computational complexity than a given base algorithm.

3.1 Memory-based model

The memory-based approach shows the basic idea behind collaborative filtering algorithms (see section 1) most clearly: It tries to predict the preference $\hat{u}^*_{p^*}$ of a particular user *(active user, U^*)* for a product p^* *(active product)* by weighting predictions $\hat{u}^*_{p^*,U}$ each based on only a single other user U. The weights $\omega_{U,\mathbb{U}}$

[1] There are two main classes of measurements [4]: One focusing on the order of different predictions, one on the quality of each prediction separately.

[2] The *coverage* is the percentage of the products for which a prediction can be given [7].

[3] *Memory-based* algorithms "operate over the entire user database to make predictions" [2].

[4] *Model-based* algorithms use the user database first "to estimate or learn a model, which is then used for predictions" [2].

[5] To compare this paper more easily with others the *mean absolute error (MAE)* is also stated.

measure how *similar* the active user is to an other user U within the considered neighborhood $\mathbb{U}$ (see eq. (1)).

$$\hat{u}^*_{p^*} = \sum_{U \in \mathbb{U}} [\omega_{U,\mathbb{U}} \cdot \hat{u}^*_{p^*,U}] \tag{1}$$

This is generalization of the approach of [2], allowing e.g. differences in the user variances as suggested in [5]. Two problems of determination have to be solved: The individual prediction $\hat{u}^*_{p^*,U}$ and the corresponding weight $\omega_{U,\mathbb{U}}$ for a given user U.

3.2 Modeling individual predictions

A simple approach to model the preferences for a product p of the active user y_p and another user u_p is to assume a linear relationship (M) between them.

$$u^*_p = \alpha_U + \beta_U u_p \tag{M}$$

An advanced approach is to assume a linear relationship between their deviation from the average product preference $\bar{t}_p$ calculated over all known preferences for the product p.

$$\begin{aligned} &\tilde{u}^*_p = \alpha_U + \beta_U \tilde{u}_p \\ \text{with} \quad &\tilde{u}^*_p = u^*_p - \bar{t}_p,\ \tilde{u}_p = u_p - \bar{t}_p \end{aligned} \tag{M*}$$

Since the advanced model (M*) can easily be transformed into the simpler model (M), we focus on the simpler one, knowing that similar results can be used for the advanced model as well.

Both approaches assume the parameter α_U and β_U to be constant for all products for any two users. α and β are estimated on in common rated products p, such that the model holds up as good as possible, i.e. the sum of the resulting errors $\varepsilon_{p,U}$ squared is as small as possible (eqn. (2)). One common technique to solve this kind of problems is called *ordinary least squares (OLS)*, which leads to the results shown in eqn. (3).

$$\min_{\hat{\alpha}_U, \hat{\beta}_U} \sum_p \Big[\underbrace{u^*_p - (\hat{\alpha}_U + \hat{\beta}_U u_p)}_{\varepsilon_{p,U}}\Big]^2 \tag{2}$$

$$\begin{aligned} \hat{\alpha}_U &= \overline{u}^* - \hat{\beta}_U\, \overline{u} \\ \hat{\beta}_U &= \frac{\hat{\sigma}_{U^*}}{\hat{\sigma}_U} r_U \end{aligned} \tag{3}$$

Here r_U denotes the correlation coefficient between the active user U^* and an other user U. Their average preferences $\overline{u}^*$, $\overline{u}$ and estimated standard deviations $\hat{\sigma}_{U^*}$, $\hat{\sigma}_U$ are calculated on products they rated in common.

3.3 Weighting individual predictions

Assuming the independence of the error terms of different users, it can be shown, that the expected squared error for the active product p^* is minimal if and only if the weights $\omega_{U,\mathbb{U}}$ have been chosen invers proportional to the corresponding expected squared errors [3].

$$\mathrm{E}\left[\left(\hat{u}^*_{p^*} - u^*_{p^*}\right)^2\right] = \min \Longleftrightarrow \omega_{U,\mathbb{U}} \sim \frac{1}{\mathrm{E}\left[\varepsilon^2_{p^*,U}\right]} \quad (U \in \mathbb{U}) \tag{4}$$

In other words, the weights should measure the expected prediction error and therefore depend on the model used.[6] When using an OLS-estimated linear relationship (M) the expected squared error $\mathrm{E}\left[\varepsilon^2_{p^*,U}\right]$ is well known to be

$$\mathrm{E}\left[\varepsilon^2_{p^*,U}\right] = \left(1 + \frac{1}{n_U} + \frac{(u_{p^*} - \overline{u})^2}{\sum_p (u_p - \overline{u})^2}\right) \frac{\sum_p \varepsilon^2_{p,U}}{n_U - 2} \tag{5}$$

where the user's average $\overline{u}$ and sums are calculated over the number n_U of products rated in common by the active user and the other considered user U.

Independently of the technique used for estimation, the neighborhood $\mathbb{U}$ should be chosen as large as possible, because every user considered reduces the error. Therefore the neighborhood should include all those users who have rated the active product p^* (necessary to make the prediction) and at least 3 products the active user has rated on (necessary to estimate the expected error).

3.4 Remarks

The presented model is based on a weighted average of linear predictions. The weights should be chosen inverse proportional to the corresponding expected squared errors (assumed to be independently). In the model therefore, the forecast of a single *perfect predictor* cannot be disturbed by other less perfect ones. This obviously correct result can never be achieved when the measured *similarity* between two people is bounded. As a bounded measurement, the correlation coefficient underweights therefore indeed more accurate predictors.

To use this model to predict behavior — which often implies binary data commonly found in e.g. ecommerce applications — other models of estimation and / or individual representation might be more appropriate.

4 Empirical Validation

In this section, the two previously described models (M) and (M*) are evaluated on the EachMovie dataset, provided by Compact Computers. Next to the

[6] For this reason, sequentially modifying and testing single modifications of an algorithm may not lead to an optimal solution and should therefore be avoided.

technique of *ordinary least squares (OLS)*, a second one developed in [5] is considered: Their algorithm — already used by e.g. [6] as a benchmark — fit the model (M) and can therefore be also applied to the model (M*).

4.1 Dataset

The EachMovie dataset is a widely used dataset for testing collaborative filtering algorithms [1, 2, 6, 8, 11]. The dataset is based on 72 916 users who were able to rate on 1 628 movies between February 1st. 1996 and September 15th. 1997. A total number of 2 811 983 individual preferences — based on a discrete rating scale of 0 to 5 *stars* — are known. 73% (252 876 votes) of all 0 star ratings were assigned to movies the users did not nor intended to see *(sounds awful)*.

The average votes per user is 38.56, the median is 20 (see Figure 1). As unequally as the votes are distributed among the users, they are among the movies (Figure 2): Most users evaluated the same movies.[7] Therefore the number of in common rated movies is higher than one would expect by a density of known preferences of 2.4%.

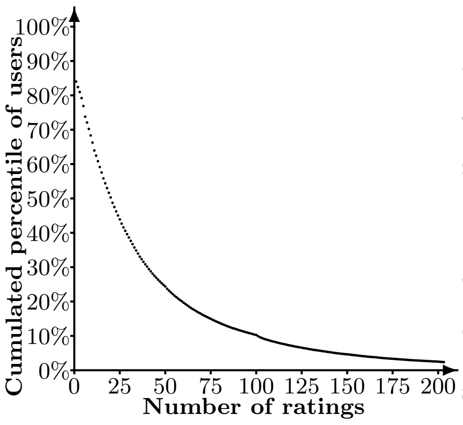


Fig. 1. Distribution of ratings among the users

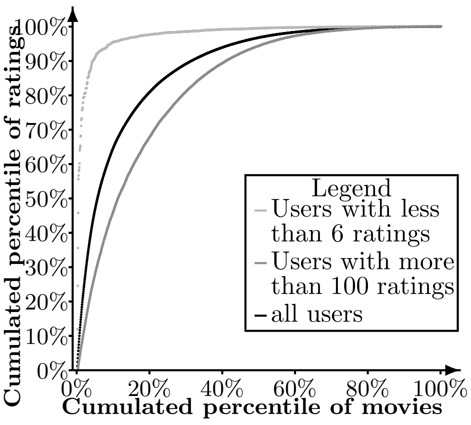


Fig. 2. Distribution of ratings among the movies

4.2 Setup

In order to perform accurate tests, all zero-star votes which are based on unseen movies and afterwards all users without any ratings were deleted from the dataset. 20% of those remaining users were used as a constant hold-out–sample

[7] The Figure 2 shows that 65% of the ratings are on 10% of the movies. This quota of ratings is higher (95%) for people with a few and lower (47%) for people with many ratings.

from which the active users were taken. In order to double the number of predictions, the other 80% were evenly divided into two datasets, used for generating the predictions.

To avoid an unrealistic bias in the tests, the setup maintained the unequal distribution of preferences found in the dataset in two aspects. First, three scenarios were used to examine the effect of the number of known preferences for the active user. The number of known preferences for the active user was chosen to be a small number (5) in the first, an average number (20) in the second and a large number (as many as possible, but not less than 20) in the third scenario. In any scenario, 6 other preferences were withheld to be predicted (see Table 1).[8]

Table 1. Scenarios

	Scenario 1	Scenario 2	Scenario 3
Number of known preferences	5	20	20 and more
Number of active user	9 228	6 005	6 005
Number of predicted movies per user	6	6	6
Number of databases	2	2	2

Second, the set of preferences known from the active user was randomly drawn out of his rated movies. In order to assure a realistic set, the probability of a movie preference to be known had been chosen proportional to the movies popularity in the dataset.[9] The movie preferences to be predicted were estimated under the remaining ones the same way, because users rated these movies most likely next.

4.3 Algorithms

The method of ordinary least squares (OLS) is compared with the one implied by an algorithm developed in [5]. Their algorithm[10] matches the simple model (M), where the parameters are given by

$$\begin{aligned}\hat{\alpha}_U &= \overline{u}^* - \hat{\beta}_U \overline{u},\\ \hat{\beta}_U &= \frac{\sigma_{U^*}}{\sigma_U} \cdot 1,\\ \omega_{U,\mathbb{U}} &\sim |r_U| \cdot \min\{50; n_U\},\end{aligned} \tag{H}$$

[8] To avoid a bias, all active users with less than $5+6=11$ $(20+6=26)$ ratings were excluded from the first (second / third) scenario.

[9] A total random choice would have lead to more predictions of more popular movies using less popular ones.

[10] This algorithm predicts significantly more precisely [3] than one based on [2], which e.g. is used as a benchmark in [1].

and the neighborhood $\mathbb{U}$ is limited to those 30 users U with the highest positive weights $\omega_{U,\mathbb{U}}$ which are based on the correlation coefficient r_U and the number of in common rated products n_U between the active user U^* and the corresponding user U. For robustness, the user averages $\overline{u}^*$, $\overline{u}$ and standard deviations σ_{U^*}, σ_U are estimated over all their known preferences.

Both methods of estimation are used for prediction with each of the two models (M) and (M*). The four resulting algorithms (see Table 2) were examined.[11]

Table 2. Algorithms examined

Based on	OLS	Herlocker et al.
Model (M)	OLS	H
Model (M*)	OLS*	H*

4.4 Results

The results of the four tested algorithms are shown in Table 3 and visualized in Figures 3 to 8. A Wilcoxon–Wilcox significance test showed that the order of the algorithms implied by the MAE[12] to be significant at the 1% level.[13]

$$\text{(worse)} \qquad \text{OLS} \prec \text{OLS}^* \prec \text{H} \prec \text{H}^* \qquad \text{(better)}$$

Table 3. Prediction errors

	Scenario 1		Scenario 2		Scenario 3	
Algorithm	MSE	MAE	MSE	MAE	MSE	MAE
OLS	1.5724	0.9755	1.3787	0.9113	1.4114	0.9208
OLS*	1.5592	0.9555	1.4096	0.9019	1.3503	0.8943
H	1.4853	0.9270	1.2066	0.8320	1.2151	0.8464
H*	1.3910	0.9064	1.1574	0.8260	1.1797	0.8393

[11] In all four algorithms occurred a *division-by-zero–errors* during calculations, whenever the variance on the in common rated movies vanished. In those cases, the non-active user was treated as non existent for the particular prognosis.

[12] The higher MSE but lower MAE of the OLS* compared to the OLS algorithm in the second scenario (see Figure 5) is due to a few *out-of-bounds prognoses*, caused by some almost vanishing standard deviations in the denominator.

[13] The significance level of 1% was achieved in the second scenario after tripling the number of predictions.

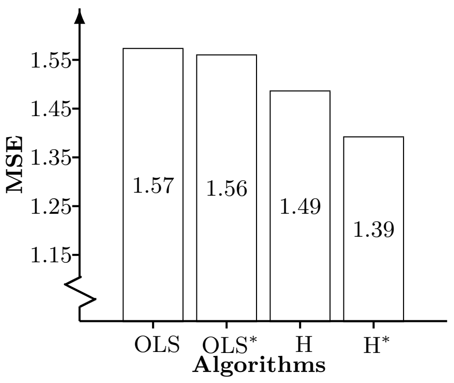


Fig. 3. MSE in scenario 1

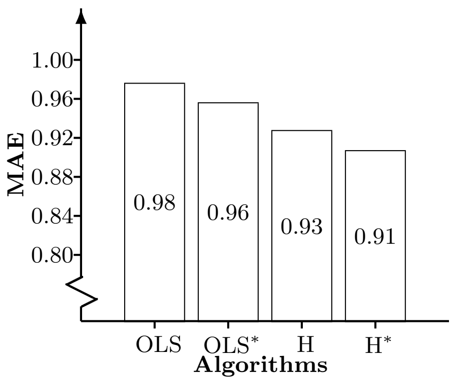


Fig. 4. MAE in scenario 1

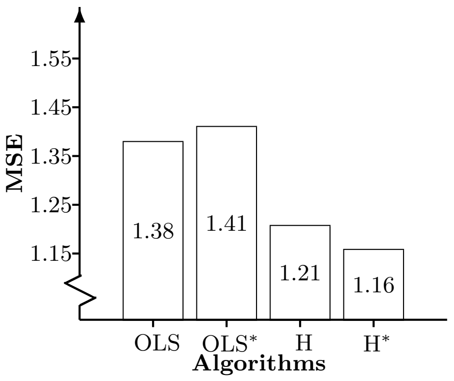


Fig. 5. MSE in scenario 2

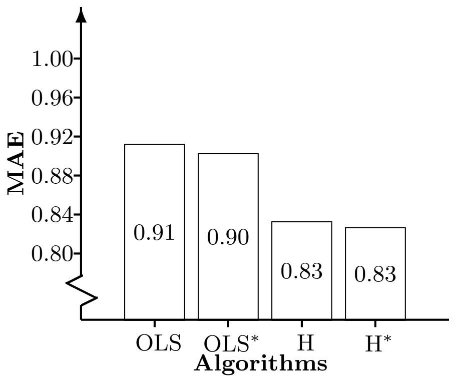


Fig. 6. MAE in scenario 2

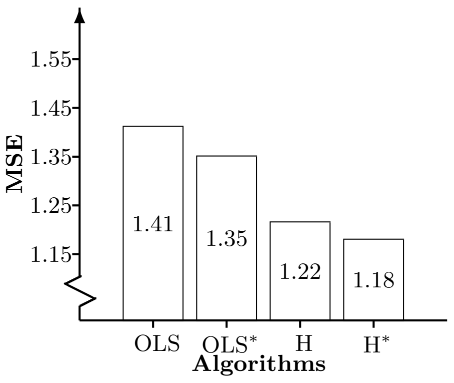


Fig. 7. MSE in scenario 3

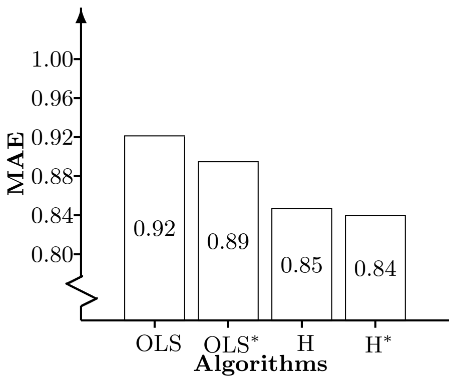


Fig. 8. MAE in scenario 3

Most surprisingly, the two OLS based algorithms performed poorly compared to the algorithm of [5], although theoretical constructed thoughts concluded otherwise. This is mainly due to the fact, that the OLS algorithms assume *perfect data* for their calculations: They react very sensitive to e.g. the estimated prediction error which can only be approximated with little data from a discrete zero to five star rating scale. It can therefore be concluded that the choice of algorithm depends on the data used, because of a trade off between the theoretical quality and robustness of estimation. Similar results were already found in [2]: Their tested linear model performed comparably poorly on binary data, but well on the EachMovie dataset.

In all cases, the advanced model (M*) was able to outperform the model (M) significantly without increasing its computational complexity. Therefore it seems reasonable to consider the advanced model (M*) as a basis for future algorithms.

The general decrease of the prediction error from the first to the second scenario is due to better knowledge of the active user and therefore better determination of accurate predictors and their errors. Because of the setup chosen (see section 4.2), the popularity of the predicted movie decreases simultaneously and so does the relevant neighborhood. For this reason, several better predictors are excluded and a set of less accurately predicting neighbors remain. This effect dominates from the second to the third scenario, so that the mean prediction error increases.[14]

5 Summary

Memory-based collaborative filtering is a technique for personalizing content on the web. Since finding the right algorithm is a problem of optimization, the objective function i.e. the measurement of its quality should be given *before* its development. In this paper, the algorithms primary objective was to minimize the squared error of the predicted preferences.

This objective, the model of weighting individual predictions (1), and the assumption of independence of the error terms from those predictions determine the weights for the model: The weights should be chosen inverse proportional to the estimated error of the corresponding predictions. This implies:

- The weights depend on the model of individual predictions and vice versa.
- A sequential testing of modifications which ignores this interdependency might lead to false conclusions.
- The (multiplicative) inverse similarity-measurement should be interpretable as an estimated prediction error of the individual prediction.

Analyzing the *EachMovie* dataset showed that the known preferences are sparse and not equally distributed among products nor among people. This

[14] The OLS* algorithm shows differently, because their primary problem — a few almost vanishing variances calculated on a few in common rated products — has been solved.

unequal distribution should be maintained in tests by appropriately selecting products and people. Using this data for testing four algorithms showed a trade off between *robust* and *correct* estimation of parameter. The unexpectedly weak performance of the OLS estimation can be attributed to the different nature of the data (i.e. a discrete and limited versus a continuous and unlimited scale). Due to its robust estimations, the estimation proposed in [5] showed to predict more accurately with the kind of data found in the EachMovie dataset. In this paper, their algorithm was significantly improved by applying it not on the preference data itself, but on its deviations from the average preferences of the particular products.

References

1. Asim Ansari, Skander Essegaier, and Rajeev Kohli. Internet recommendation systems. *Journal of Marketing Research*, 37(3):363–375, 2000.
2. John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th annual conference on Uncertainty in Artificial Intelligence (UAI–98)*, pages 43–52, Madison, Wisconsin, USA, July 1998. Morgan Kaufmann Publishers.
3. Jan-Hendrik Dörner. *Persönliche Empfehlungen mit Collaborative Filtering*. PhD thesis, Johann Wolfgang Goethe-University, Frankfurt am Main / Germany, 2002.
4. Jonathan L. Herlocker. *Understanding and Improving Automated Collaborative Filtering Systems*. PhD thesis, University of Minnesota, 2000.
5. Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM Press, August 1999.
6. Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering. In *The Proceedings of the SIGIR-2001 Workshop on Recommender Systems*, pages 14–21, New Orleans, Lousiana, USA, September 2001.
7. Mark O'Connor and Jon Herlocker. Clustering items for collaborative filtering. In *Proceedings of the twentysecond annual international ACM SIGIR conference on research and development in Information Retrieval*, Berkeley, August 1999. ACM.
8. David Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the sixteenth annual conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 473–480, Stanford, CA, 2000.
9. David M. Pennock, Eric Horvitz, and C. Lee Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *Proceedings of the 17th national conference on Artificial Intelligence (AAAI-2000)*, pages 729–734, Austin, Texas, July 2000.
10. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the conference on Computer supported cooperative work*, pages 175–186. ACM Press, October 1994.
11. Matthias Runte. *Individualisierte Angebote mit Collaborative Filtering*. PhD thesis, Christian-Albrechts-Universitt zu Kiel, Gabler, Wiesbaden, 2000.

12. Badrul M. Sarwar, Joseph A. Konstan, Al Bolchers, Jon Herlocker, Brad Miller, and John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of the ACM 1998 conference on Computer Supported Cooperative Work (CSCW)*, pages 345–354, Seattle, WA, November 1998.

User adaptive matchmaking [*]

Björn Fiehn
Rather Str. 4
Düsseldorf, Germany
bjoern.fiehn@web.de

Jörg P. Müller
Siemens AG, CT IC 6
Otto-Hahn-Ring 6
Munich, Germany
joerg.p.mueller@siemens.com

ABSTRACT

In electronic markets, matchmaking mediates demand and supply based on profile information. Since a matchmaking system is usually employed to support the decisions of a real person, it is desirable to have the user's preferences reflected in the actions of the matchmaking system. These preferences are usually modeled using a configurable system which may be adapted to the actual requirements and user's needs. The question, however, is, how can the system react to changing user preferences resp. how can the system be adapted to different user requirements. We follow an approach which incorporates explicit and implicit feedback into the matchmaking process and thus enables the user to react to the results presented by the matchmaking system. The User Adaptability Framework introduced in the course of this paper offers a general adaption solution with support for recording implicit and explicit user feedback, performing feedback analysis and generating analysis reports. We present several feedback analysis algorithms suited for different purposes which have been implemented using the User Adaptability Framework. In a subsequent evaluation these algorithms are eventually compared regarding to analysis performance and the evaluation results discussed.

Keywords

User-adaptability, user preferences, matchmaking, machine learning

1. INTRODUCTION

Matchmaking [12] [10] is the process which determines the relevance of an offered item compared to a given query in a specific domain. Situations in which the matchmaking process is typically used are market place scenarios where participants offer specific items and other users are interested in selecting the item which satisfies their requirements at best. In typical application scenarios the structure of the offered items and the specification of the requirement can reach a complexity so that a human user is not able to perform the process in an acceptable time. Of course not only the complexity drives the need for automated methods but also the wish to implement autonomous user agents which can act in the background according to the user preferences and are able to return with a result that satisfies the user's requirements.

As different users usually determine the relevance of an offer in different ways, a matchmaking system which forces a single preference model for all users will most probably suffer from acceptance problems as users want to see their own preferences reflected in the returned results.

This requirement leads to a number of questions:

1. How can user preferences be modeled so that a matchmaking system is able to incorporate these preferences in the matchmaking process?
2. Given a preference model, how can the actual user preferences be communicated to the matchmaking system?
3. How can the system cope with changing preferences?
4. What can be done if the user is not satisfied with the results returned by the system?

This paper deals with the above questions by first introducing the SieMatch Matchmaking Framework by Siemens AG [7] as an example for a highly configurable matchmaking system. In the context of [3] this system has been extended by the User Adaptability Framework, which provides a simple way to incorporate user feedback in the matchmaking process and which defines an architecture for feedback analysis algorithms to learn the actual user preferences without forcing the user to specify them explicitly. Using this framework, we will develop several analysis algorithms which improve the matchmaker configuration by processing the user feedback.

2. THE USER ADAPTABILITY FRAMEWORK

[*]The full version of this paper is available as the diploma thesis *Architektur and Basisalgorithmen eines Adaptiven Matchmakingsystems* (in german) at `Philipps-University of Marburg, Department of mathematics and computer science`

Preference Learning: Models, Methods, Applications. Workshop to be held as part of the conference KI-2003 2003 Hamburg, Germany.

As the underlying matchmaking system we use the SieMatch Matchmaking Framework developed by the Siemens AG. The SieMatch Framework follows a multi-dimensional matchmaking approch. It is based on the assumption that the match relevance of a candidate with regard to a given query (a centroid in the SieMatch terminology) is composed of one or more dimensions or "aspects" between the candidate and the centroid. An application of the matchmaker in the human resources domain might for example define the dimensions "Wage expectation", "Skills" and "Education" between applicants and openings. For each of these dimensions, the SieMatch Matchmaker will compute the relevance of the candidate for the given centroid. Using the dimension relevances, the matchmaker applies an aggregate function which computes the overall match relevance. Although the SieMatch Framework is able to use arbitrary aggregate functions, we restrict our focus to a weighted sum function which incorporates a user preference model based on the multi-attribute utility theory introduced in [14].

We will now provide a mathematical model for the matchmaking process, which will be used in the course of this paper to describe the developed framework and algorithms.

Let

- The application domain Dom, e.g. $Dom =$ "human resources".
- The set $C = C_{Dom}$ of candidates from the domain Dom.
- The set $Q = Q_{Dom}$ of centroids (queries in the matchmaking system) from the domain Dom.
- The recursively defined set $Dim = String \times \wp(Dim)$ of all dimensions with $String =$ set of all dimension labels. Thus, a dimension is either atomic or composed of several subdimensions and has a specific label describing its purpose.

For a fixed SieMatch configuration let:

- $D = \{D_1, ..., D_n\} \subseteq Dim$ be the dimensions between centroid and candidate where
  $$\forall D_i \in D : D_i = \begin{cases} (s_i, \{D_1^i, ..., D_{n_i}^i\}) & n_i\, sub-dim. \\ (s_i, \emptyset) & else \end{cases}$$
  and $s_i \in String$ is the unique dimension label, e.g. $s_2 =$ "soft skills".
- For every dimension $D_i = (s_i, \emptyset)$ a distance function $f_i : Q \times C \to [0; 1]$ to determine the match between candidate and centroid dimension.
- For every dimension $D_j = (s_j, \{D_1^j, ..., D_{n_j}^j\})$ an aggregate function $g_j : [0; 1]^{n_j} \to [0; 1]$ and $f_j(q, c) := g_j(f_1^j(q, c), ..., f_{n_j}^j(q, c)) \cdot k_j(f_1^j(q, c), ..., f_{n_j}^j(q, c))$.
- An aggregate function $g : [0; 1]^n \to [0; 1]$ for the computation of the overall result.
- For every dimension $D_i = (s_i, \{D_1^i, ..., D_{n_i}^i\})$ let $k_i : [0; 1]^{n_i} \to \{0, 1\}$ be defined as
  $$k_i(x_1, ..., x_{n_i}) = \begin{cases} 0 & \exists j : x_j = 0\, and\, D_j^i\, k.o. - crit. \\ 1 & else \end{cases}$$
  Let a similar function k be defined for the top-level dimensions D.

Using these notions, the overall distance between a centroid $q \in Q$ and a candidate $c \in C$ can then be computed as $f(q, c) = q(f_1(q, c), ..., f_n(q, c)) \cdot k(f_1(q, c), ..., f_n(q, c))$.

This work uses only a weighted sum aggregate function: $g(x_1, ..., x_n) = g_{w_1,...,w_n} = \frac{\sum_{i=1}^n w_i \cdot x_i}{\sum_{i=1}^n w_i}$. This is certainly a restriction, but empirical studies [13] show that this model yields satisfactory results.

The main goal of this paper is to introduce the User Adaptability Framework which has been developed during the diploma thesis [3]. In order to discuss the developed framework, we first need to explain what we understand by the terms "user adaptability" and "user feedback".

2.1 User adaptability

Definition 1. (User adaptability). Automated learning of the user's preferences concerning the computation of the matchmaking results so that the matchmaking system is able to produce results which reflect the user's requirements. The preferences can cover these aspects of the matchmaking system:

- *Dimension weights:* How important are the dimensions for the user with regard to the main result? How can these weights be be automatically fitted to the user's preferences without forcing him or her to specify the weights directly?
- *Selection of k.o.-criteria:* Which dimensions are k.o.-criteria for the user, that is, which dimensions should disqualify a candidate, if the dimension distance lies below a certain threshold?
- *Selection of the aggregate function:* How does the user determine the overall matchmaking result from the distances in the dimensions? Does the user prefer a different aggregate function than specified in the matchmaking configuration?
- *Selection of distance functions:* Are there distance functions which reflect the user's preferences better than the currently used ones?
- *Definition of the dimensions between query and candidates:* Determine the structure of the dimensions for the user. Is it possible that some elements of the profiles which haven't been regarded in the configuration are relevant for the distance computation?
- *Source of the candidates (Connectors):* Which candidate sources ("connectors" in the terms of the SieMatch Framework) deliver candidates that are preferred by the user? Is it possible to identify sources which usually provide the system with candidates with a low rating?

This work concentrates on an adaptive dimension weighting and selection of k.o.-criteria in the matchmaking configuration. All other aspects will be discussed shortly in section 5 but have not been realized, mostly due to the limited time of the diploma thesis.

The user adaptability should be realized by recording explicit and implicit feedback of the user to the matchmaking results presented by the matchmaker.

Definition 2. (Explicit feedback). Explicit feedback gives the user the possibility to express his or her opinion concerning the presented matchmaking results. We distinguish two basic types of explicit feedback:

- *Qualitative feedback*: The user associates a number $x_{\mathrm{M}} \in [-1;1]$ to the matchmaking result. We distinguish different modes M of qualitative feedback: $M \in \{MA, MR, CA, CR\}$ which provide the number x_M with a semantic meaning
  - *MA (Match, absolute)*: The user gives a direct rating of the quality of the matchmaking result.
  - *MR (Match, relative)*: The user rates the returned result in relation to his or her impression of the candidate (e.g. the returned match is far too good for the candidate).
  - *CA (Candidate, absolute)*: An alternate rating of the candidate, i.e. the match result the user expects the matchmaker to return for this candidate.
  - *CR (Candidate, relative)*: The user rates the candidate in relation to the matchmaking result (the opposite statement as MR feedback).
- *Ranking orders*: The possibility to arrange the candidates (usually a subset) according to the user's preferences without actually specifying the concrete match relevances. This is probably a more natural way of giving feedback than associating an abstract number to a candidate.

Definition 3. (Implicit feedback). Observing the user's actions concerning the returned matchmaking results and candidates. Some of the actions which may be performed can give a hint on the user's preference regarding the candidates, as for example deleting a candidate or inviting a candidate to a job interview. In the simplest form of implicit feedback, these actions can be associated to qualitative feedback.

The User Adaptability Framework integrates all presented kinds of feedback and provides the application developer with an API that enables a simple integration of the framework into existing matchmaking applications. The framework is divided into a matchmaking independent part and the Adaptive Matchmaking Component which uses the common api to implement a matchmaking-specific adaption solution. The matchmaking independent part consists of the following components:

- *Action history:* A component that enables the recording of arbitrary user actions throughout the program flow. The design is kept in such a way, that almost no assumptions on the structure of the actions were made, except that an action must have a name and can have a type and one or two serializable parameters. The component offers several Enterprise Java Beans based interfaces to define, store and process user actions in the application server.
- *Application profile management:* Delivers the basis for storing the user preferences and context data required for feedback analysis.
- *Analysis and Report:* Provides the possibility to perform an analysis in the application server using a session bean that has been parameterized using the *Command* design pattern [2] to support different kinds of analysis algorithms and the cooperation of multiple algorithms. Additionally, the component contains an interface for generation of analysis reports. These reports are be used to return the result of the feedback analysis to the user or to pass the intermediate result of an algorithm to the succeeding algorithm.
- *Miscellaneous helper components*, e.g. an access tool for the EJBs that enables to develop the algorithms independently of the actual execution place, that is, inside the application server or at the client.

2.2 Related work

Concering the matchmaking topic, much research has been done by Sycara and Klusch who introduced the LARKS matchmaker in [10]. Multidimensional matchmaking on which the work of this paper is based has been covered by Veit, Müller, Weinhardt in [12]. The GRAPPA framework presented in [12] integrates negotation role definitions and is in many aspects different from other solutions: flexible definition of demand and supply profiles, arbitrary description schemes for the matchmaking logic including nested structures and an open design of the framework which enables the easy integration of new matchmaking algorithms.

This paper extends the matchmaking approach by enabling user-adaptable preferences. In [5] Guo and Müller follow an information-theoretic approach to elicit utility functions automatically using user feedback. This approach has been extended and applied to the matchmaking problem. Other methods used to learn user-preferences presented in this paper involve genetic algorithms. [6] and [4] provide good introductions to these subjects.

3. LEARNING ALGORITHMS

The User Adaptability Framework supports pluggable and exchangeable analysis algorithms for different kinds of feedback. As one of the main results of [3] several algorithms for qualitative and ranking order feedback have been developed which. These algorithms will be presented in this section.

We use a genetic algorithm approach (see [6][4]) as a basis for two analysis algorithms. This method was chosen because it offers a flexible way of analyzing feedback of different kinds and can be easily extended for future analysis algorithms.

3.1 Analysis of qualitative Feedback: Qualitative Feedback Analyzer

The first analyzer based on the genetic algorithm is suited best for qualitative feedback in the mode *Candidate, absolute*. The basic idea behind this algorithm is to try to generate a dimension weighting that implies a minimal deviation from the given user feedback. To support the other kinds of qualitative feedback as well, we use the heuristic conversion described in [3], appendix B.

Hypothesis representation

A hypothesis $h \in H$ about a possible matchmaker configuration consists of a representation of the dimension weights and the choice of k.o. criteria. The weight and the k.o. criterion marker for each dimension are converted into a bit string.

3.1.1 Fitness function

To determine the quality of a hypothesis the genetic algorithm uses a fitness function which computes the fitness $f(h)$ for a given hypothesis h. In our case, the fitness function must determine how good the matchmaking results produced under a given hypothesis fit to the feedback given by the user. Of course, the analysis algorithm should be careful with proposing k.o.-criteria, since these criteria are a powerful way of disqualifying candidates. We use a rather conservative way of dealing with k.o.-criteria, i.e. we accept only k.o.-criteria in a hypothesis where there is not a single instance of user feedback that does not conform with the selected k.o.-criterion. We use the following feedback function: Let

- $U \subseteq Q \times C$ the set of centroid / candidate pairs rated by the user.
- $f_{user}^{CA} : U \rightarrow [0; 1]$ the mapping that associates the *Candidate, absolute* feedback to a centroid / candidate pair. Feedback in other modes will be converted using the respective conversion rules ([3], appendix B).
- $f : U \rightarrow [0; 1]$ the mapping that computes the total distance between the centroid and the candidate using the matchmaking configuration.
- $f_h : U \rightarrow [0; 1]$ do., using the matchmaking configuration proposed in hypothesis h
- P the current population of hypothesis

Let the fitness function $F : P \rightarrow [0; 1]$ be defined by

$$F(h) := min \left\{ \frac{\sum_{u \in U} |f_{user}(u) - f_h(u)|}{|U|}, K(h) \right\}$$

and

$$K(h) := \left\{ \begin{array}{ll} 0 & falls\, \exists u \in U : k_h(u) = 0 \wedge f_{user}(u) > 0 \\ 1 & sonst \end{array} \right.$$

The function F computes the deviations between the matchmaking results produced using the weights and k.o.criteria from the hypothesis h and the ratings (i.e. *Candidate, relative* feedback) given by the user. $K(h)$ is used to disqualify a hypothesis with an illegal k.o.-criterion.

3.1.2 Genetic operators

The analyzer uses a slightly modified version of the standard *Mutation*, *Probabilistic and Elitist selection*, and *Crossover* operators to generate a best-fitting hypothesis. Additionally, in order to take care of the special requirements regarding the dimension weights and the k.o.-criteria, we use:

- *Weight normalization:* Normalizes the weights w_i in the hypothesis h so that $\sum_{i=1}^{n} w_i = 1$ with $w_i \in [0; 1]$
- *"Hypothesis tamer":* Deletes the k.o.-criteria of a hypothesis that has been disqualified due to a false k.o.-criterion. A k.o.-criterion is called "false" if it leads at least once to disqualifying a candidate that was not ranked with a maximum distance by the user.

3.2 Analysis of ranking orders: Ranking Order Feedback Analyzer

The algorithms presented in the preceding text were only applicable to qualitative feedback. As the second way of giving explicit feedback, the User Adaptability Framework allows the user to specify ranking orders of the candidates for a given query.

It is the goal of the algorithm that analyzes the user-given ranking orders to generate a dimension weighting and selection of k.o.-criteria under which the ranking orders given by the user are preserved, i.e. the candidates are returned in the same sequence as specified by the user.

The RankingOrderFeedbackAnalyzer algorithm is also based on the genetic algorithm. The main difference to the already presented UserRatingFeedbackAnalyzer algorithm is the special fitness function which provides a measure of how good the ranking orders are preserved using the configuration proposed by the hypothesis.

3.2.1 Input of the algorithm

The input for this analysis algorithm is:

- $R_1, ..., R_m \subseteq \{((q, c_1), ..., (q, c_n)) \in (Q \times C)^n; n \in \mathbb{N}_{>1}\}$ the ranking orders specified by the user. A ranking order $R_i = ((r, c_1), ..., (r, c_n))$ contains candidates c_j that have been put into a ranking order with regard to the common centroid q.

3.2.2 Selection of the fitness function

Crucial for a successful application of a genetic algorithm to search for a suitable configuration is the selection of the fitness function. We chose the following way to determine the quality of a hypothesis h:

- Determine the number of neighbor permutations necessary to transform the ranking order generated under h into the ranking order given by the user divided by the number of maximum permutations.

If we have more than one instance of ranking order feedback, we compute the arithmetic mean of the above value.

The computation of the fitness of a hypothesis h consists in addition to the value mentioned above of two more components:

- *Square deviation of the original configuration weights:* Computes the square deviation between the weights w_i in hypothesis h and the original weights w_i^{orig} in the currently used configuration. This ensures that the new configuration deviates as little as possible from the original configuration.
- *Conservative treatment of k.o.-criteria:* If h contains a k.o.-criterion, so that a at least once a candidate that has been disqualified because of this criterion does not appear at the end of a ranking order, this hypothesis is discarded.

The fitness function $F : P \to [0; 1]$ is defined as follows:

$$F(h) = min\left\{0.95 \cdot f(h) + 0.05 \cdot g(h), K(h)\right\}$$

with

$$g(h) = \frac{n - \sum_{i=1}^{n}(w_i - w_i^{orig})^2}{n}$$

and

$$K(h) = \begin{cases} 0 & \text{at least one k.o.-crit. violates a ranking order} \\ 1 & else \end{cases}$$

3.2.3 Genetic operators

We use the same operators as in the preceding algorithm.

3.3 Analysis of mutual information: Mutual Information Feedback Analyzer

A completely different way of analyzing qualitative user feedback is the mutual information approach. This statistical method tries to reveal a correlation between the positive and negative rating of the user and the value of the candidates in each dimension. The algorithm developed here extends the adaptive weight determination method proposed in [5] and applies it to the matchmaking situation.

The mutual information

$$H(A, B) = \sum_{a \in A}\sum_{b \in B} P(A = a, B = b) \log \frac{P(A = a, B = b)}{P(A = a)P(B = b)}$$

introduced by Shannon in [9] measures the reduction in the uncertainty in the prediction of the value of A, $E(A)$, if the value of the second variable B is already known.

The basic idea of this analysis algorithm is to compute $H(A, B)$ between the direction of the user feedback F and the distance D_i of each dimension and thus determine how a good or bad value in a single dimension correlates with positive or negative user feedback.

Using these mechanisms we take the following approach:

- Let $U \subseteq Q \times C$ be the set of all centroid / candidate pairs rated by the user.
- Let $f_{user} : U \to [-1; 1]$ the mapping that associates the qualitative feedback in the mode *candidate, relative*[1] to every $u \in U$.
- The distances occurring in a dimension D_i will from now on not be regarded absolute but relative to the average distance in this dimension for all matched candidates. To differentiate between the actual dimension value and the discrete value, we use the notation D_i for the original and $\tilde{D}_i$ for the discrete value.
- For a distance function $f_i : Q \times C \to [0; 1]$ let $f_i^{relative} : Q \times C \to [-1; 1]$ be defined as the relative distance respective to the average dimension distance of dimension D_i

  $$f_i^{relative}(x) := \begin{cases} \frac{f_i(x) - r_i}{1 - r_i} & \text{if } f_i > r_i \\ \frac{f_i(x) - r_i}{r_i} & \text{if } f_i < r_i \\ 0 & else \end{cases}$$

  with $r_i \in (0; 1)$ the reference distance for dimension D_i.
- The feedback F given by the user will also be treated discretely so that only positive and negative feedback is regarded. Notation: $\tilde{F}$.
- Computation of the mutual information $H(\tilde{D}_i, \tilde{F})$ between each dimension D_i and the *candidate, relative* feedback F given by the user (D_i and F are being regarded as discrete random variables, to match with the above presentation).
- Computation of $h_i := \begin{cases} \frac{H(\tilde{D}_i, \tilde{F})}{E(\tilde{F})} & \text{if } E(\tilde{F}) \neq 0 \\ 0 & else \end{cases}$ as the mutual information relative to the uncertainty in the user feedback $E(\tilde{F})$.

Thus, h_i is the fraction by which the uncertainty in the user feedback is reduced if it is known, if D_i is above or below average. Unfortunately, the mutual information $H(\tilde{D}_i, \tilde{F})$ alone does not not make a proposition about the actual direction of the correlation. [3] shows an example, where $H(\tilde{D}_i, \tilde{F})$ is identical for two opposed situations.

To avoid this problem, we take the feedback and the old weights into account. The new weights $\tilde{w}_i$ are computed as follows: Let

- $w_i \in [0; 1]$ be the weight of dimension D_i in the original configuration
- $\forall i \in \{1, ..., n\}$ let $s_i : U \to [-1; 1]$ be defined as $s_i(u) := sign(f_i^{relative}(u) \cdot f_{user}(u))$ with $f_i : Q \times C \to [0; 1]$ the i-th distance function of the matchmaking configuration
- Then $\hat{w}_i$ is defined as

  $$\tilde{w}_i := w_i + k \cdot \frac{H(\tilde{D}_i, \tilde{F})}{\sum_{j=1}^{n} H(\tilde{D}_j, \tilde{F})} \cdot \frac{\sum_{u \in U} s_i(u)|f_{user}(u)}{|U|}$$

[1]if the qualitative feedback is not available in this mode, the corresponding conversion functions are applied

Using the common factor k the new weights are fitted so that the square deviation between the relevances under the new weighting w_i and the feedback in mode *candidate, absolute* is minimized. Accordingly, k is determined by a extremal value computation so that the following sum is minimized:

$$\sum_{u \in U} \left(\frac{\sum_{i=1}^{n} \hat{w_i}(k) \cdot f_i(u)}{\sum_{i=1}^{n} \hat{w_i}(k)} - f_{user}^{CA}(u) \right)^2$$

4. EVALUATION

4.1 Underlying model

We used an automated evaluation method in order to measure the quality and performance of the developed algorithms. The following algorithm describes the steps taken to evaluate the feedback analysis algorithms:

Let

- n_{user} be the number of virtual users of the matchmaking system
- $q \in Q$ be a fixed centroid
- C with $n_{cand} := |C|$ be the set of available candidates
- $M \subset F \times A$ be the set of tuples $(f_i, a_i) \in F \times A$ with

  $$F = \text{the set of feedback generator algorithms}$$

  and

  $$A = \text{set of analysis algorithms}$$

Then the evaluation algorithms consists of the following steps:

```
FOR i := 1 TO $n_{user}$ DO

  1. For the current user generate a configuration $conf_{user}$
     with randomly chosen dimension weights and perform
     the matchmaking between the candidates from $C$ and
     the centroid $q$ using that configuration.
  2. Generate a configuration $conf_{system}$ with equally
     weighted dimensions.
  3. $\forall (f_i, a_i) \in M$ perform the following steps:
     (a) Generate the feedback using algorithm $f_i$ and create
         the instances of the analysis algorithms $a_i$
     (b) Perform the analysis in the application server
     (c) Change $conf_{system}$ according to the analysis results
     (d) Perform the matchmaking using the adapted configuration $conf_{system}$
     (e) Compute the deviations between the relevances
         computed by the matchmaking system using the
         configurations $conf_{user}$ and $conf_{system}$

END FOR
```

Since the User Adaptability Frameworks supports arbitrary combinations of the algorithms which do not all make sense and to keep the evaluation effort in a sensible range, we restricted the evaluation to a selected number of algorithm combinations:

- The UserRatingFeedbackAnalyzer, RankingOrderFeedbackAnalyzer, MutualInformationFeedbackAnalyzer algorithms separately
- UserRatingFeedbackAnalyzer + RankingOrderFeedbackAnalyzer

4.2 Experimental results

The following graphs present the results obtained in the automated evaluation. We measured the change in the deviation between the computed relevances (continuous line) and the dimension weights (dashed line). To gain better comparability, the deviations were normalized on the maximum deviation, i.e. the deviation before the analysis (1.0 = deviation from the weights without analysis; 0.0 = no deviation between computed weights and given weights).

The results show that the qualitative feedback analysis yields the best results, the genetic algorithm User Rating Feedback Analyzer being the most promising method (fig. 1 and fig. 4).

The Ranking Order Feedback Analyzer algorithms appears to be the worst analysis algorithm (fig. 2). However, regarding its functioning, the result is not at all surprising. This algorithm optimizes the configuration that all user-given ranking orders are preserved and the generated configuration has the minimum deviation from the original configuration. In other words, it assumes that the original configuration is already usable and may need only slight improvements.

Combining the User Rating Feedback Analyzer and the Ranking Order Feedback Analyzer algorithms shows that the results of the first algorithm improve after the application of the ranking order analysis (fig. 3).

5. CONCLUSION AND OUTLOOK

In the preceding chapters we introduced the User Adaptability Framework which is able to adapt the configuration of an arbitrary matchmaking system to the preferences concerning the dimension weighting and selection of k.o.-criteria of the user. This adaption is done by the analysis of implicit feedback which has been recorded by observing the user's interaction with the application and by feedback which has been given directly by the user.

The User Adaptability Framework provides the infrastructure for adaption solutions by offering reusable components such as preference and profile management and a framework for analysis algorithms. Although this paper focuses on the matchmaking-based application of the User Adaptability Framework, its universal applicability is demonstrated in [3] which contains an completely matchmaking-independent example that uses the framework.

We introduced several feedback analysis algorithms suited for various purposes and presented evaluation results which

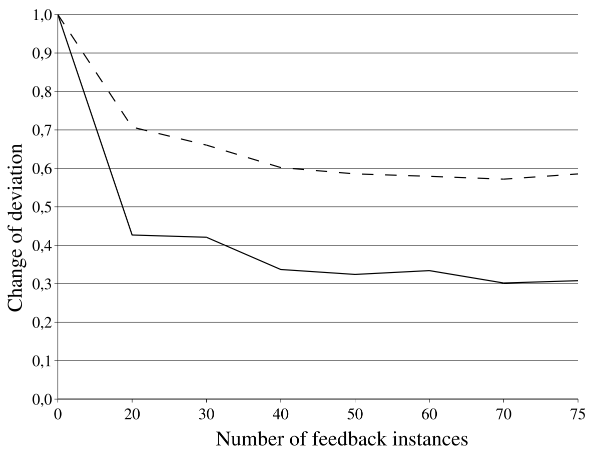


Figure 1: Reduction of the deviation between the relevances and the dimension weights (User Rating Feedback Analyzer)

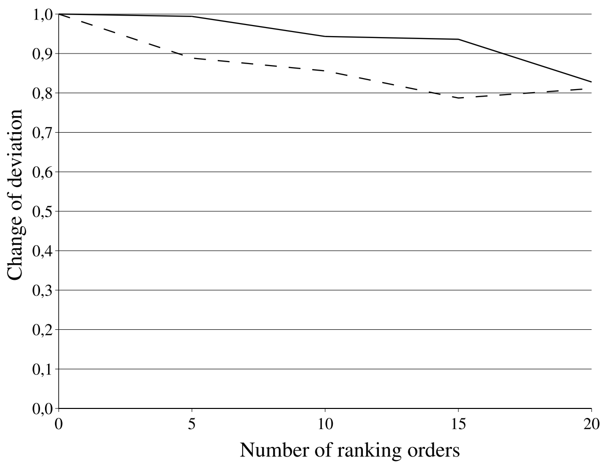


Figure 2: Reduction of the deviation between the relevances and the dimension weights (Ranking Order Feedback Analyzer)

show that these algorithms work satisfactory. These results also show that the combination of feedback analysis algorithms which process different kinds of feedback (e.g. qualitative feedback and ranking orders), which is a main functionality of the User Adaptability Framework, yields a result which is better than the application of only one analysis algorithm.

These algorithms have to be regarded as sample implementations of feedback analysis algorithms as the User Adaptability Framework was designed in such a way that new algorithms can easily be plugged in.

Integrating the User Adaptability Framework into matchmaking-based applications promises a broader user acceptance as the framework offers the possibility for the user to optimize the matchmaking behavior of the application according to his preferences. This optimization eventually leads to an improved matchmaking quality perceived by the user.

However, a few aspects have not yet been covered which may yet lead to interesting applications of the User Adaptability Framework. These aspects will in the following be shortly presented.

- *Automated selection of a configuration for a centroid:* The current version of the SieMatch Framework uses a common configuration for all queries of a user in a domain (e.g. human resources). There are, of course, situations when several configurations in one application are necessary. As for example in a human resources scenario when openings for a secretary and a department manager have to be assigned. Obviously, most personnel consultants would consider different aspects for the candidates for those openings. An interesting extension of the adaption idea would be to support the automated selection of the correct specialized configuration for a given centroid.
- *Learning of alternate aggregate functions:* This work assumes that the total relevance of candidate is composed of a weighted sum of all distance results. Giving up this assumption eventually leads to new possibilities for feedback analysis. With the help of appropriate algorithms, the complete aggregate function could be designed adaptive, so that more elaborate aggregate functions can evolve.
- *Learning of alternate distance functions:* The previously discussed case of adaptive aggregate functions can of course be transferred to the distance functions themselves. This work assumes that the distance functions return correct values for the aspects between candidate and centroid and offers method to optimize the weightings of these aspects. There may be situations when either the existing distance function do not return satisfactory results or no distance function is available at all. In this situation, there are at least two ways of learning the distance functions by analyzing user feedback:
    1. *Composing the distance functions of base distance functions:* As proposed in [11] if a specialized distance function for a dimension is not available, the system might try to break that dimension down into sub-dimensions and apply base distance functions available in the matchmaking system (e.g. free-text or number range comparisons). Thus, a dimension obtains a substructure with several sub-dimensions that produce, with the help of an aggregate function, the overall result.
    2. *Learning of general distance functions:* If the above approach is not applicable, this more general method tries to learn the complete distance function by analyzing the user feedback. This general problem of classifying data belongs to the field of information retrieval.

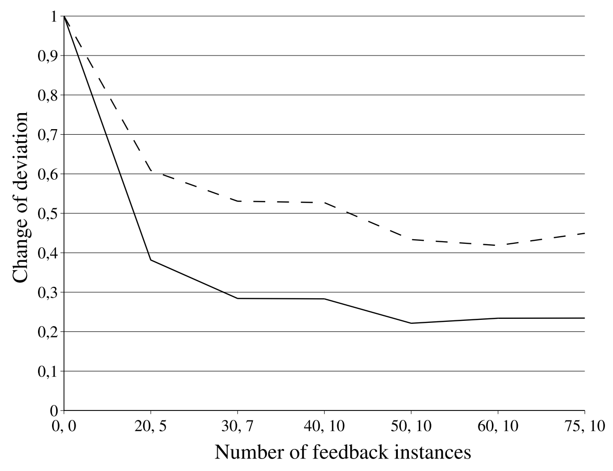


Figure 3: Reduction of the deviation between the relevances and the dimension weights (User Rating Feedback Analyzer and Ranking Order Feedback Analyzer)

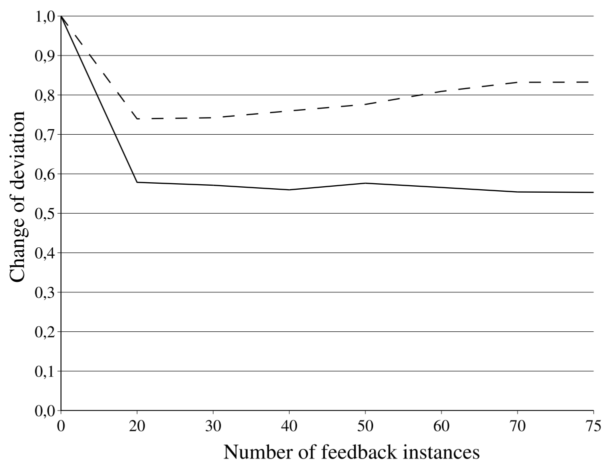


Figure 4: Reduction of the deviation between the relevances and the dimension weights (Mutual Information Feedback Analyzer)

- *Adaptive transformation of the results of a distance function:* Assuming that a distance function produces correct results for the dimensions between candidate and centroid, one could ask the question if the distribution of that values is always correct for every user. A conservative user might tend to devaluate a candidate in a specific dimension much faster than a less conservative user. A user-adaptive system can try to automate this process for the user by selecting the appropriate transformation function (e.g. threshold or distribution functions).

6. REFERENCES

[1] Ester, M., Sander, J., Knowledge Discovery in Databases. Techniken und Anwendungen, Springer, Heidelberg, 2000

[2] Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns, Addison Wesley, 2000

[3] Fiehn, B., Architektur und Basisalgorithmen eines adaptiven Matchmakingsystems (in german), Diploma thesis, Philipps-University of Marburg, Department of mathematics and computer science, 2003

[4] Goldberg, D. E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, Mass., 1989

[5] Guo, Y., Müller, J.P., Automated Utility Elicitation in Agent-based Negotiation, Internal paper, Siemens AG CT, Intelligent Autonomous Systems, 2002

[6] Mitchell, T., Machine Learning, McGraw-Hill Education (ISE Editions), 1997

[7] Müller, J. P., Jasny, R., Astor, J., Guo, Y., Veit, D., Kleegrewe, C., SieMatch: Intelligent Sourcing and Matchmaking in Electronic Marketplaces, Internal presentation, Siemens AT, CT IC6, 2002

[8] Schäfer, R., Rules for Using a Multi-Attribute Utility Theory for Estimating a User's Interests, Online-Proceedings des 9. GI-Workshops: ABIS-Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen, 2001

[9] Shannon, C. E., A Mathematical Theory of Communication, Bell Systems Technology Journal, 27, 1948, S. 379-423

[10] Sycara, K., Lu, J., Klusch, M., and Widoff, S., Matchmaking Among Heterogeneous Agents on the Internet, Proceedings AAAI Spring Symposium on Intelligent Agents in Cyberspace, Stanford, USA, 1999

[11] Veit, D., Müller, J. P., Schneider, M., Fiehn, B., Matchmaking for Autonomous Agents in Electronic Marketplaces, Proceedings of the Fifth International Conference on Autonomous Agents (Agents 2001), S.65-66, ACM, Montreal, 2001

[12] Veit, D., Müller, J. P., Weinhardt, C., Multidimensional Matchmaking for Electronic Markets, Proceedings of the Third International Symposium: From Agent Theory to Agent Implementation (AT2AI-3), Volume 2, S. 713-718, Vienna, 2002

[13] Veit, D., Electronic Negotiations and Multidimensional Matchmaking - an Agent based Approach, PhD. thesis, University of Karlsruhe, 2002

[14] Winterfeldt, D. von, Edwards, W., Decision Analysis and Behavioral Research, Cambridge, England, Cambridge UniversityPress, 1986