Combining Case-Based and Similarity-Based Product Recommendation

Armin Stahl

German Research Center for Artificial Intelligence (DFKI) GmbH
Research Group Image Understanding and Pattern Recognition (IUPR)
Technical University of Kaiserslautern
Erwin-Schrödinger-Str. 57, 67663 Kaiserslautern, Germany
Armin.Stahl@dfki.de

Abstract. Product recommender systems are a popular application and research field of CBR for several years now. However, almost all CBR-based recommender systems are not case-based in the original view of CBR, but just perform a similarity-based retrieval of product descriptions. Here, a predefined similarity measure is used as heuristics for estimating the customers' product preferences. In this paper we propose an extension of these systems, which enables case-based learning of customer preferences and which also allows to incorporate collaborative recommendation techniques. Further, we show how this approach can be combined with existing approaches for learning the similarity measure directly. The presented results of a first experimental evaluation demonstrate the feasibility of our novel approach in an exemplary test domain.

1 Introduction

With the increasing success of e-Commerce web-sites, the development of intelligent *recommender systems* has become a popular field of research. Today, many e-Commerce sites are already deploying recommender systems to support their customers during the selection of a product that best matches their requirements and preferences. Depending on the type of offered products, the desire for such support can be explained by different issues:

- When being confronted with huge product databases, the search for a suited product can become very time consuming.
- When purchasing complex products (e.g. technical products like PCs, travels [**?**], insurance products [**?**]) customers do often not possess the expertise to select the optimal product with respect to their requirements.
- Some products cannot be described sufficiently by explicit and objective properties (e.g. books, music [**?**], videos [**?**]) but are selected on the basis of subtile aspects like the personal taste. Without recommendation a customer cannot estimate the personal value of such a product until purchasing it.

Since the requirements on the actual recommendation process are varying between different business scenarios, many different recommendation techniques

have been development during the last years (for an overview see [**?**,**?**]). In principle, two major approaches can be distinguished: *content-based recommendation* and *collaborative filtering (CF)*.

Content-based recommendation can deal with the first two issues, i.e. finding suitable products in large databases or advising customers when purchasing complex products. Therefore, the customer has to define his requirements on the searched product, e.g. by filling out a predefined query form. This information is then compared with the descriptions of the available products in order to identify a set of potential product candidates. If the comparison is based on exact match (e.g. by performing a simple SQL query), this is called *filter-based recommendation (FBR)*. An alternative to FBR, which often leads to unsatisfactory results, is *similarity-based recommendation (SBR)*. Here, the comparison between the query and the product descriptions is based on a specific similarity measure which also allows to rank retrieved products. As *Case-Based Reasoning (CBR)* provides powerful techniques for realising similarity-based retrieval it has become a popular technique for building SBR systems [**?**].

Collaborative filtering [**?**], on the other hand, is typically be used to deal with the third issue, i.e. to provide recommendations for products that cannot be described sufficiently by explicit properties. The basic idea of CF is to collect user ratings about seen or bought products and to use rating correlations between different users and products in order to recommend products. Hence, CF relies on a vast amount of user feedback before producing satisfactory recommendations.

In recent years several hybrid recommendation techniques which incorporate content-based and collaborative approaches have been developed [**?**], and some of them apply also CBR techniques [**?**,**?**,**?**,**?**].

In principle, a recommender system must possess knowledge about the customers' requirements and preferences and their relationship to the offered products. Generally, the following types of user needs can be distinguished [**?**]:

- hard requirements vs. preferences
- explicit vs. implicit preferences, i.e. is the preference explicitly expressed in the query or not
- general vs. individual preferences, i.e. is it a general preference of almost all customers or is it customer specific
- short-term vs. long-term preferences, i.e. is the preference only valid for the actual recommendation process or durable

While FBR can only treat the hard, explicit, individual and short-term preferences encoded in the query, SBR allows a much wider consideration of customer preferences. Here, similarity-measures (which may be customer specific) can be used to model almost all kind of preferences. Only implicit subtile preferences, that are difficult to express formally, can be treated exclusively by CF techniques.

The most challenging task when building a recommender system is the acquisition of knowledge about the different kinds of preferences. While CF strictly relies on user feedback, SBR is applicable without any feedback by using a predefined similarity measure as heuristics. However, the quality of this heuristics influences the recommendation quality dramatically.

In this paper we propose a novel approach for learning customer preferences in content-based recommender systems. The approach combines SBR with the original idea of CBR, i.e. the reuse of collected experience knowledge. Therefore, it incorporates knowledge about successful recommendations of the past into the similarity-based product retrieval. We show that an additional optimisation of the underlying similarity measure yields in further improvements of the recommendation quality.

The advantage of our approach is its easy integration into state-of-the-art SBR systems. At the beginning the system can be applied with a standard similarity measure without relying on any user feedback. If feedback becomes available during usage it will enable the system to learn its users' preferences automatically over time leading to improved recommendation results. Moreover, our approach also provides the possibility to consider more subtile product properties and to include collaborative aspects into the recommendation process.

Section **??** starts with a short review of the functionality of SBR systems and existing approaches towards learning customer preferences. Section **??** then describes our novel approach which combines case-based learning with existing techniques for learning similarity measures. The results of an experimental evaluation presented in Section **??** demonstrate the feasibility of our approach. After discussing related work in Section **??**, we conclude with a summary and an outlook on future work.

2 Similarity-Based Product Recommendation Systems

Similarity-based recommendation systems have become a very popular CBR research area and numerous successful commercial applications are in use today. Surprisingly, on a closer look, most of these systems are not at all CBR systems in the traditional view of CBR since the used "cases" do not represent problem-solution pairs of the past but are typically just product descriptions.

2.1 Utility-Oriented Matching

The basic functionality of a SBR system is illustrated in Figure **??**. Given a customer *query* which describes the desired product properties, a CBR system applies a predefined similarity measure for comparing the query with the descriptions of all available products which are stored in a *product database (PB)*. Finally, a ranked set of the s most similar products (s is typically is set to 10) is presented as the *result set (RS)* to the customer[1].

If we look at this scenario, it becomes obvious that the system does not compare two problem descriptions like assumed by the traditional idea of CBR. Instead, it compares a problem—the query—directly with potential solutions—the products. This works well for product recommendation because here problems and solutions can be described by using the same vocabulary. However, it also

[1] In this paper we do not consider adaptation.

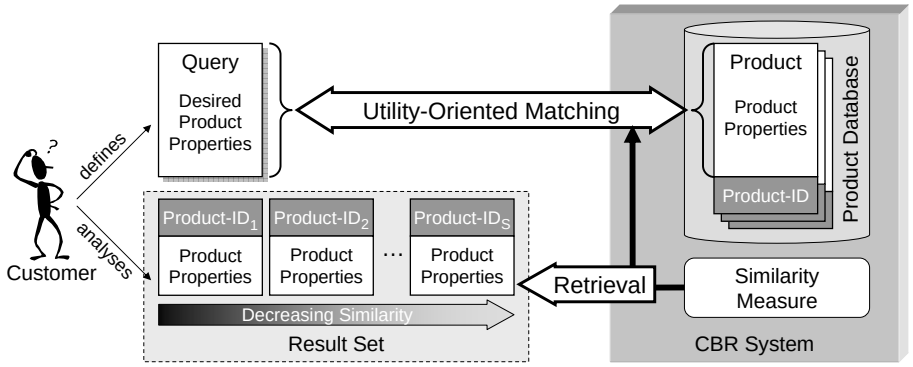

Fig. 1. Product Recommendation by Utility-Oriented Matching

restricts the features that can be used during the recommendation process to the information contained in existing product descriptions.

Traditional cases consisting of problem-solution pairs are not used at all in this scenario. Some authors have characterised this approach as *utility-oriented matching* because the similarity measure is directly used to approximate the utility of known solutions—here product descriptions—for a given problem [**?**].

In principle, the utility of a product description p_i with respect to a given query q can be characterised as the conditional probability that the product will be accepted by the customer—we denote this event as ω_i—given q, i.e. we may define a *utility function* u as follows by applying Bayes rule:

$$u(q,p) = P(\omega_i|q) = \frac{P(q|\omega_i) \cdot P(\omega_i)}{P(q)} \tag{1}$$

In a SBR system, a predefined similarity measure sim is used to approximate this unknown utility function u. Since it does not possess any other knowledge about the customers' preferences, the recommendation quality of such a system depends completely on the accuracy of this approximation.

In the machine learning point of view, the SBR task can be characterised as a n-class classification problem, where n represents the number of available products. Each product description p_i represents a prototype of class ω_i and can be characterised as a point[2] in a high dimensional feature space where each feature corresponds to a product property. In order to recommend the optimal product for a given query q, the query has to be classified as belonging to one of the classes ω_i. This leads to a huge number of decision boundaries between all classes with their corresponding prototypes p_i (see Figure **??**a)). In our example, q would be classified as ω_1, because it lays in the corresponding Voronoi cell of ω_1 (marked grey), i.e. p_1 is considered to be the optimal product. In order to recommend a set of products, the utility functions $u(q, p_i)$ for all products p_i, i.e. the posterior probabilities $P(\omega_i|q)$, have to be estimated (see Figure **??**b)).

[2] We do not consider generalised cases.

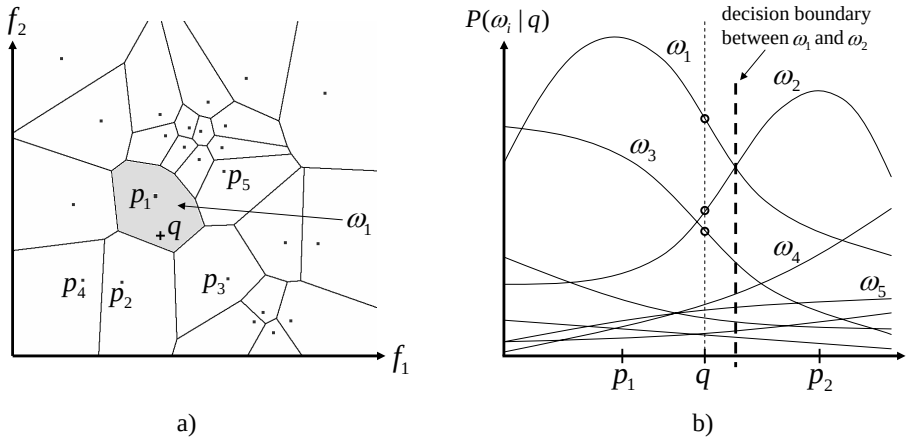

Fig. 2. Similarity-Based Product Recommendation as Classification

For a given set of product descriptions p_i, in SBR the decision boundaries are completely determined by *sim*. If *sim* is conform to the Euclidean Distance, for example, in the 2D case the decision boundaries can be visualized by a simple Voronoi diagramm with respect to the points p_i as shown in Figure **??**a).

2.2 Dealing with Customer Preferences

However, because of the complexity of customer preferences, in practice standard similarity measures such as the Euclidean Distance will result in a poor approximation of u. In principle, u will be determined by different kinds of preferences with different locality in the problem space:

1. the different importance of general product properties, e.g. "the price is usually much more important than the colour"
2. preferences concerning different values of product properties
   (a) independently from q and other properties, e.g. "black cars are generally preferred over white cars"
   (b) depending on q but independently from other properties, e.g. "if a black car is desired, a dark blue car will likely be preferred over a yellow car"
   (c) depending on other properties, e.g. "black BMWs are mostly preferred over red BMWs"
3. product specific preferences that are independent from q (in the probabilistic view this is the prior probability $P(\omega_i)$ of class ω_i in formula (1)), e.g. "the silver BMW 320i is a very popular car and is generally preferred over many other similar cars"

With common similarity measures supported by CBR tools [**?**], influence 1 can be modeled with global *feature weights* and influences 2a) and 2b) can be modeled with *local similarity measures*. However, in particular the definition of

accurate local similarity measures is a very time consuming task. The consideration of influences 2c) and 3) would require more sophisticated measures requiring a modeling effort that is usually not tolerable in practice. But the more serious problem of defining an accurate similarity measure is the a-priori completely missing or only partially available knowledge about the customers' preferences.

In our previous work we have proposed to apply a specific machine learning approach which allows automatically learning of feature weights and local similarity measures based on user feedback [**?**,**?**,**?**]. We have shown that this approach also allows the incorporation of partially known background knowledge into the learning process [**?**]. However, in particular the learning of local similarity measures is generally susceptible to overfitting if not enough user feedback is available. Moreover, the approach does not provide a solution for the consideration of all above enumerated kinds of preferences.

3 Case-Based Learning of Customer Preferences

In this section we present an alternative approach for learning customer preferences which avoids some of the problems of the previously described approaches.

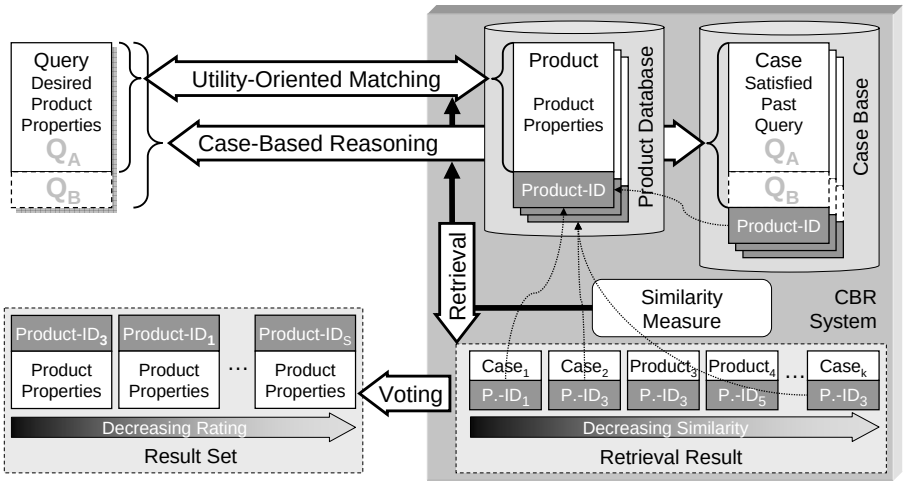

Fig. 3. Case-Based Learning of Customer Preferences

The basic idea of this approach is illustrated in Figure **??**. At the beginning of its life cycle, the extended recommender system will behave like a standard SBR system, i.e. it will perform utiltiy-oriented matching on the given PB. However, any time a customer has selected a product that is acceptable for him (e.g. if s/he orders the product), his query (optionally together with additional information, see Section **??**) will be stored in the case base CB together with the product-ID of the selected product. These records now represent actual cases in the

traditional view of CBR; the combination of a problem description—the query—and a corresponding solution—the accepted product. And more important, such cases contain implicit knowledge about the customers' preferences.

During following recommendation sessions, this knowledge can be used to estimate u more accurately as it is possible with a predefined similarity measure sim alone. Therefore, the current query q is not only matched against PB, but also against CB by using sim. The corresponding merged *retrieval result* $RR := (r_1, \ldots, r_k)$ where $r_i \in PB \cup CB$ with $sim(q, r_i) \geq sim(q, r_j)$ for all $i < j$ is then used to perform a k-NN classification [**?**] of q.

3.1 Voting Strategy

Because in the recommendation scenario one is not only interested in the most probable class, but in the s most probable classes, the definition of a fixed k is not suited for our approach. If RR contains only r_i that correspond to $t < s$ different classes, the system will not be able to recommend s different products. Hence, k has to be determined dynamically after each retrieval process. It will be set to the smallest possible value k, so that RR includes exactly s different classes, respectively products.

Now RR can be used to generate the required result set RS consisting of s different product proposals. The ranking of these products within RS will be determined by some voting strategy. Different voting strategies are commonly used in k-NN classification, e.g. majority voting, weighted voting [**?**] or advanced and adaptive approaches [**?**,**?**]. In our experiments (see Section **??**) product p_x is ranked over product p_y (written as $p_x \succ p_y$) according to the following weighted majority voting rule, where all $r_i^{\omega_x}$ correspond to the same class ω_x:

$$p_x \succ p_y \Leftrightarrow \sum_{r_i^{\omega_x} \in RR} sim(q, r_i^{\omega_x}) > \sum_{r_i^{\omega_y} \in RR} sim(q, r_i^{\omega_y}) \tag{2}$$

Such a simple voting strategy may lead to overfitting problems as long as only view cases have been collected because the corresponding products will then have a much higher probability to be recommended compared with still unbought products. More adaptive strategies which take the cases' class distribution into account might outperform the proposed strategy.

Finally, the actual product descriptions have to be retrieved from the product database in order to generate the final result set to be presented to the customer. This is necessary because the retrieval result might contain only cases which do not contain the product description itself but only the product-ID.

3.2 Learning Additional Case Indexes

Up to now, we have assumed that queries consist of the same attributes that are used in the original product descriptions (denoted as Q_A in Figure **??**). As already described, this is a precondition for applying utility-oriented matching. However, by learning cases of successful recommendation sessions, this is no

longer strictly necessary. One may enable the customer to ask also for additional product properties that are not contained at all in the original product descriptions (part Q_B of the query). Typical examples of such additional query items are more subtile (e.g. "I want a very sporty car") or functional requirements (e.g. "I want to use my PC mainly for gaming"). In principle, a fixed set of such additional features may be considered in the query interface or the interface may provide the option to enter some free text to be processed by textual CBR techniques.

These additional desires of the customer cannot improve the recommendations if the case base is still empty. However, the more cases that contain such information are stored in the case base, the higher will be the influence of this information on the recommendation results. In principle, the queries of the customers are then used to implicitly index the products automatically by using additional features which would be to expensive to be done manually by domain experts. At some point, this statisctial information could also be extracted automatically from the case base in order to extend the product descriptions stored in the product database. Another possibility is the incorporation of collaborative features (e.g. user profiles) [**?**] in part Q_B of the query.

3.3 Acquisition of Training Data

As typical for a supervised learning approach, the acquisition of accurate training data is crucial. In our approach, we assume that the customer states some query (this query might also be the result of a sales dialog [**?**]) and in the case that he accepts one of the proposed products (e.g. because s/he orders it), this data is used to create a new training example, i.e. a new case. To control the learning process, one may choose one of the CBL algorithms [**?**]. For example, when applying the CBL2 algorithm, one would store a new case only if the ordered product was not recommended as the optimal product.

However, in general it cannot be guaranteed, that the resulting case represents an optimal query-product pair. Maybe there are other products in PB that the customer has not seen, but that s/he would definitely prefer. This means, we will only get *relative utility feedback* [**?**] about the utility of the products included in the result set[3]. If the system proposes s different products $p_1, \ldots, p_s$ and the customer orders p_3 we only get evidence, that it holds $u(q, p_3) \geq u(q, p_i)$ for all $i \leq s$, but we do not obtain any information about the absolute value of $u(q, p_3)$. However, this information would be necessary in order to ensure that the learned case alone represents accurate knowledge about u.

This situation is less problematic if the retrieval set contains the optimal case with high probability, also if it is not ranked correctly. Hence, the quality of the predefined similarity measure which determines the initial result sets is crucial in order to restrict the noise in the training data. However, learning of extremely noisy training examples is generally unlikely because then the customer would not have ordered the product.

[3] Here we assume that the customer analyses each product contained in the result set.

To guarantee a minimal quality of the used similarity measure it is possible to apply machine learning, too. In [**?**,**?**,**?**] we have presented an algorithm for learning similarity measures which can handle the kind of relative feedback that we obtain in the recommendation scenario. Hence, this feedback can also be used to optimise the similarity measure in parallel or a-priori to learning new cases in order to reduce the noise in the training data.

4 Experimental Evaluation

In order to evaluate our novel approach we have performed some first experiments in a simulated product recommendation scenario.

4.1 Test Domain

As test scenario we have chosen a used cars domain consisting of 100 descriptions of different used cars which we have extracted from a real world online used cars market. Each car is described by 4 numeric and 4 symbolic attributes, such as price, power, color, year of construction, etc. For a more detailed description of the used test domain see [**?**].

Since we were not able to perform an experiment with real world customers, we have simulated imaginable customer preferences with a manually defined similarity measure sim_U consisting of specific feature weights and specific local similarity measures for each attribute. Of course, such a model is not sufficient for simulating the actual behaviour of real world customers. On the one hand, it does not simulate the inconsistencies between the individual preferences of different customers that would occur in the real world. On the other hand, it does also not model all kinds of preferences discussed in Section **??** (2c and 3 cannot be modeled with such a kind of similarity measure). However, it is sufficient for a first prove of concept of our approach.

4.2 Experiments

In order to evaluate the capability to learn the simulated customer preferences in the described test domain, we have performed several experiments where we have applied the proposed case-based learning approach and/or our previous algorithm for learning feature weights [**?**]. In principle, each experiment consisted of the following steps:

1. create empty case base CB, empty feedback set FB, and initialize standard similarity measure sim with uniform weights
2. select a set of training queries $Q_{train} := (q_1, \ldots, q_{10000})$
3. for each $q_i \in Q_{train}$ do
   (a) generate result set $RS_i := (p_1, \ldots, p_{10})$ consisting of 10 product descriptions p_j by following the procedure described in Section **??** and by using q_i, sim, CB and the static product database PB

(b) determine *preferred product* $pp_i := \arg\max_{p_j \in RS_i} sim_U(q_i, p_j)$

(c) generate feedback $FB_i := (pp_i, (\bar{p}_1, \ldots, \bar{p}_9))$ where $\bar{p}_l \in RS_i \setminus pp_i$

(d) store feedback, i.e $FB := FB \cup FB_i$

(e) optional: learn feature weights from FB and update sim accordingly

(f) create a new case c_i from q_i and the product-ID of pp_i

(g) optional: insert c_i into CB by applying CBL1 or CBL2

(h) if $i \in \{5, 10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000\}$ then evaluate the recommendation accuracy on query test set Q_{test} using sim_U

By different combinations of the optional learning steps 3(e) and 3(g) we have generated the following five experiments:

SIM: Exclusively learning of feature weights by using the relative utility feedback FB which only expresses that the pp_i are more useful than all other p_j contained in the respective result sets RS_i.

CBL1/2: Exclusively applying case-based learning algorithms CBL1 (each c_i is stored) or CBL2 (c_i is stored only if $pp_i \neq p_1$ holds) (cf. [**?**]).

SIM-CBL1/2: A-priori learning of feature weights using the feedback of the first 5/10/25/50 queries and activation of CBL1/2 starting from query 51.

Each experiment was repeated with 5 different, a-priori randomly generated training query sets. For the evaluation of the achieved recommendation accuracy a static set of 250 randomly generated test queries Q_{test} was used to compute 4 different quality measures:

mpp-**in-**x**:** The average percentage of recommendation sessions, where the theoretically *most preferred product* $mpp = \arg\max_{p_i \in PB} sim_U(q, p_i)$ was contained in the first $x \in \{1, 3, 10\}$ recommended products.

avg-*mpp***:** The average position of *mpp* in the result sets.

4.3 Results

Figure **??** summarizes the results of the experiments SIM and CBL1/2. The left chart shows the achieved improvements concerning the *mpp*-in-x measures. For the exclusive optimisation of *sim* one observes a rapid ascent of all learning curves where about 10 training queries are sufficient to achieve the maximal improvements, e.g. for the *mpp*-in-10 measure an increase from 52% to 81%.

In contrast, the learning curves of the CBL experiments show much slower improvements of the recommendation quality. However, after 1000-2500 training queries case-based learning starts to outperform similarity measure learning and achieves significantly better results after 10000 queries, e.g. for the *mpp*-in-1 measure over 40% (compared to about 29%). This is not surprising since the case-based learning approach is able to learn the preferences encoded in the local similarity measures of sim_U which cannot be modeled with feature weights. However, surprisingly the differences between the CBL1 and CBL2 are very

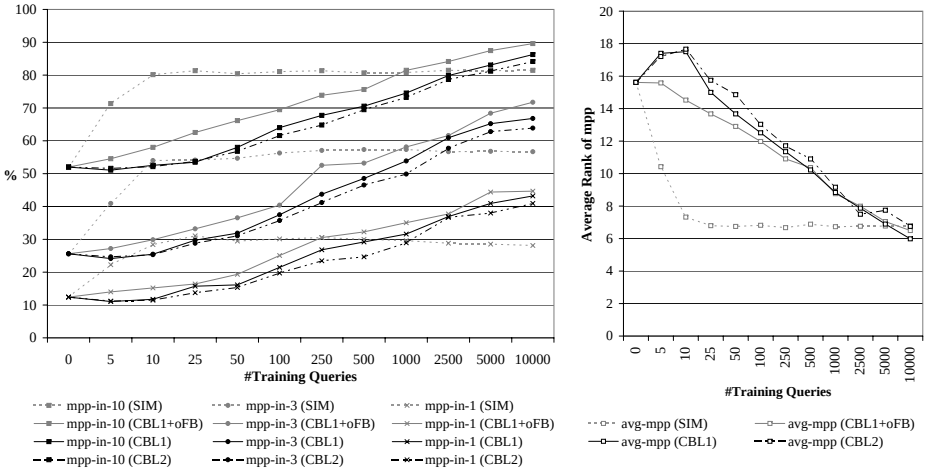

Fig. 4. Results of Experiments SIM and CBL1/2

small, even though the average number of stored cases is significantly lower in the CBL2 experiment (6032 compared to 10000 in CBL1).

In order to be able to evaluate the impact of noisy feedback, we have performed an additional CBL1 experiment with optimal feedback (CBL1+oFB) by using *mpp* instead of *pp* in step 3(b). While the CBL1+oFB learning curve shows continuous recommendation improvements from the beginning, in the realistic experiments CBL1/2 the improvements achieved with less then 50 training queries are quite small or even negative. This can be explained with overfitting which will be amplified by the noisy training data in CBL1/2 and becomes more obvious in the avg-*mpp* measure (right chart of Figure **??**).

Figure **??** shows the results of the experiments SIM-CBL1/2. Here, the first 50[4] queries were used to exclusively learn the similarity measure in order to improve the feedback quality for the subsequent case-based learning process.

The achieved results clearly show the advantage of the combination of both learning techniques. On the one hand, a-priori optimisation of *sim* ensures much faster performance gains compared with applying case-based learning alone. Now only 250 training queries are sufficient to outperform the results of SIM. This also leads to increased robustness against overfitting, since the negative impact of a too small case base is compensated by the preliminary improvements achieved by optimising *sim*. However, the avg-*mpp* curves clearly show that the overfitting effect is still present (see peak at 100 training queries) as long as the case base contains less than 50 cases[5]. Although overfitting is more prominent in SIM-CBL2, in general the differences between SIM-CBL1 and SIM-CBL2 are almost not recognisable. This is all the more surprising because the average number of learned cases is further reduced (4930) compared to experiment CBL2.

[4] according to the results of SIM even 25 queries would be sufficient

[5] Note, that the first 50 queries were not used for learning cases.

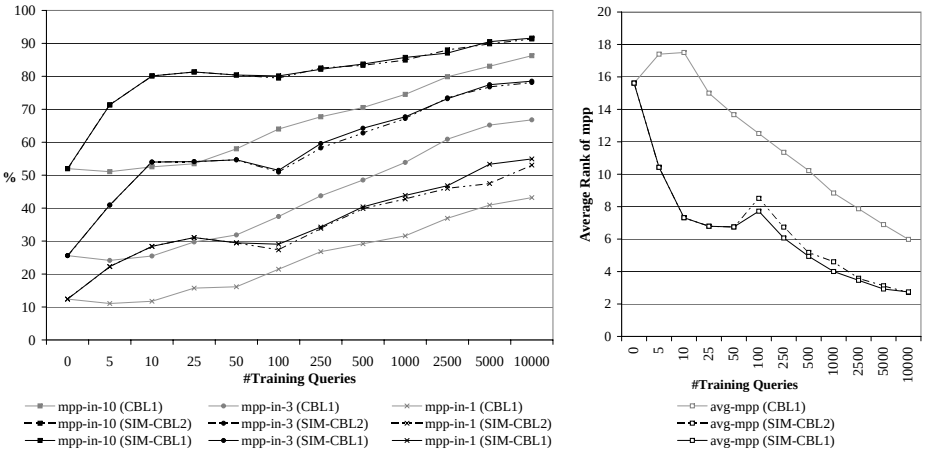

Fig. 5. Results of Experiments SIM-CBL1/2

On the other hand, the finally achieved recommendation accuracy is significantly higher compared with the results that can be achieved with each learning technique alone. This becomes particularly obvious in the avg-*mpp* quality measure. While each learning technique alone was able to decrease the average rank of *mpp* from about 16 to 6, the combination of both approaches finally achieves an average rank smaller 3. This would allow to decrease the size of the returned result sets significantly, e.g. if displayed on mobile devices [**?**].

5 Related Work

The work that is most related to the approach presented in this paper are the results of the DIETORECS project [**?**,**?**]. In this project, an advanced travel recommendation system which combines case-based and CF-based techniques has been developed. This system also uses the combination of a raw product database and a case base containing so-called *session-cases*. These cases describe recommendation sessions of the past, containing stated queries, selected travel components, and also collaborative features.

The major difference compared to our work is the kind of use of the two databases. In the DieToRec system, on the one hand, the product database is used for an initial filter-based retrieval which requires conversational techniques in order to obtain useful result sets. On the other hand, the case base is used only to determine the ranking of the previously selected cases by using collaborative techniques. Moreover, the system does not optimize the similarity measure required to retrieve session-cases.

An early algorithm which integrates case-based learning with optimisation of the required similarity measure is the CBL4 algorithm [**?**]. However, this algorithm is designed for simple classification tasks and requires absolute utility

feedback [**?**] about the correctness of the solution proposed by the CBR system. Hence, it is not applicable in the product recommendation scenario.

Other work which deals with learning of user preferences is described in [**?**,**?**,**?**]. However, none of these approaches applies a combination of case-based and similarity measure learning as proposed in this paper.

6 Conclusion and Future Work

In this paper we have presented a novel approach for learning customer preferences in content-based recommender systems. This approach extends the functionality of existing similarity-based recommender systems by applying case-based learning in combination with similarity measure learning. On the one hand, optimising the similarity measure directly improves the accuracy of the underlying k-NN classifier [**?**]. On the other hand, it also improves the quality of the absolute utility feedback required by the case-based learner.

Although in this paper we have focused on product recommendation, the approach is not restricted to this application scenario. It is also suited to learn other types of user preferences, e.g. like occurring in knowledge management domains where users are interested in getting advice about available knowledge resources (e.g., documents, web sites, pictures) with respect to their individual information needs [**?**].

The advantage of our approach is its broad applicability and its compatibility with already successfully applied SBR systems. Moreover it allows an automatical extension of the set of features used to characterise products or information ressources and the incorporation of collaborative techniques. The results of the presented experimental evaluation show the principal ability of our approach to learn customer preferences from easy to acquire customer feedback.

For future work we are planning to perform a more realistic evaluation by using a more sophisticated model of the customers' preferences including all kinds of preferences discussed in Section **??** and also nondeterministic behavior. In such a scenario the learning task is generally more challenging, however, here the case-based approach should also outperform solely learning of the similarity measure more clearly due to its less restricted hypotheses space. In such an extended evaluation it would also be interesting to investigate the impact of learning additional product features.

We also plan to further improve our approach. On the one hand, advanced voting strategies which incorporate statistical information about the learned cases might allow to model the prior probabilities $P(\omega_i)$ explicitly in order to improve the recommendation accuracy. Moreover, this might also help to reduce the risk of overfitting for small case bases. Generally, we plan to investigate the potential of the generation and incorporation of statistical models into the recommendation process with the increasing number of collected cases. On the other hand, smarter learning policies than CBL2 (e.g. such as CBL3 [**?**]) can help to reduce the size of the case base while maintaining or even improving the recommendation accuracy. This is important in order to minimise retrieval

times. Another interesting issue would be the application of our advanced similarity measure learning algorithm which allows an optimisation of local similarity measures [**?**,**?**,**?**].

Last but not least, we want to investigate whether our approach is also suited to be used in domains where products can customized [**?**,**?**].

Acknowledgements

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) under the IPeT (01 IW D03) project and by the federal state Rhineland-Palatinate under the project ADIB (Adaptive Provision of Information).