Improving Recommendation Ranking by Learning Personal Feature Weights[1]

Lorcan Coyle and Pádraig Cunningham

Department of Computer Science
Trinity College Dublin
{Lorcan.Coyle, Padraig.Cunningham}@cs.tcd.ie

Abstract. The ranking of offers is an issue in e-commerce that has received a lot of attention in Case-Based Reasoning research. In the absence of a sales assistant, it is important to provide a facility that will bring suitable products and services to the attention of the customer. In this paper we present such a facility that is part of a Personal Travel Assistant (PTA) for booking flights online. The PTA returns a large number of offers (24 on average) and it is important to rank them to bring the most suitable to the fore. This ranking is done based on similarity to previously accepted offers. It is a characteristic of this domain that the case-base of accepted offers will be small, so the learning of appropriate feature weights is a particular challenge. We describe a process for learning personalised feature weights and present an evaluation that shows its effectiveness.

1 Introduction

A particular challenge for e-commerce is to provide mechanisms that substitute for the ways in which the human sales assistant facilitates the sales process. An important component of this is the ability to identify the customer's preferences and highlight products and services that will satisfy the customer's requirements and preferences. This is particularly true in the travel domain. A dialog with a good old-fashioned business travel agent would contain phrases like; *"I presume you will want to go out on the first flight.", "You will want to return on the Friday evening.", "You will not want a stopover in Heathrow."* Ideally, an online Personal Travel Assistant will learn these preferences as well.

In this paper we describe such a system that uses CBR to rank offers returned in response to a travel request [6]. There are two types of cases in this system; session-cases and offer-cases. Session-cases represent previous user-interactions or sessions with the system and offer-cases represent individual travel offers. Session-cases can be viewed as request-offer pairs; the problem component of the case is made up of a previous travel request with some additional contextual information; the solution component is a reference to the selected offer (which is an offer-case) in response to that

[1] The support of the Informatics Research Initiative of Enterprise Ireland and the support of Science Foundation Ireland under grant No. 02/IN.1/I1111 are gratefully acknowledged.

that request. The idea behind this is that a user's preferences are encoded implicitly in the accepted offers to particular requests and that similar requests will lead to similar selections of offers. So the ranking is a two-stage process. The first stage is to find a previous session that contains a similar request to the current travel request. This session is assumed to be relevant to the user's current context. In the second stage, the current offers are ranked based on their similarity to the offer component of the retrieved session-case. This session-based recommendation approach is analogous to that used in Ricci et al.'s *DieToRecs* system [10]. Both systems rank presented items based on their similarity to items selected in response to similar queries in the past (twofold similarity) [11]. However, DieToRecs differs from our system in that it uses a mixed-initiative approach to elicit user preferences whereas we determine these preferences implicitly. We incorporate these preferences into the similarity measures used in the recommendation process. Some users will be very price conscious, others will be adverse to stopovers or long stopover times, and others will have preferences on departure times. Rather than ask users to weight the importance of these criteria we choose to learn this from past behaviour. There are two reasons for this, first, it places less cognitive load on the user. Second, it avoids the problem of asking users to assign numeric weights to criteria – a skill at which people are notoriously poor.

We use techniques along the lines of introspective learning as described in the past by Bonzano et al. [2], Branting [4] and Stahl [13, 14]. Introspective learning refers to an approach to learning problem solving knowledge by monitoring the run-time progress of a particular problem solver. The approach used here is failure-driven in the sense that an attempt is made to improve feature weights only in the case of a recommendation failure. This is done by decreasing the weights of unmatching features and increasing the weights of matching features. This will tend to push down the recommendation scores of offers that are not being taken up and pull up the scores of ones that are selected.

Section 2 discusses a number of feature weighting algorithms where user feedback drives learning. Section 3 describes our Personal Travel Assistant application and how CBR is used to recommend flights to users. In Section 4 we give a description of our feature weight learning algorithm. Section 5 presents results that show that weight learning improves recommendation accuracy. We discuss some future work in Section 6 and draw our conclusions in Section 7.

2 Feature Weighting based on User Feedback

There are a number of systems that use user feedback to assist in problem solving episodes. Mixed initiative CBR and conversational CBR systems use feedback to direct a search through a problem space, e.g. [5, 9, 12, 11]. Some learners attempt to incorporate a level of *utility* [1] into the similarity measures by looking at case order feedback [2, 4, 13, 14]. Utility is indicative of adaptability or usefulness to the current problem. We hope that by incorporating utility into the similarity measure in this way we will improve and personalise recommendations in our system.

Bonzano's et al ISAC system uses a form of *introspective learning* to improve its retrieval mechanism. Feature weights are updated in order to optimise problem solving performance. Stahl [13, 14] describes a *similarity teacher* that has knowledge of the utility function of what the system is trying to learn. This teacher goes through every retrieved set of cases and uses its utility function to calculate a similarity error. By minimising this error on a feature by feature basis, he attempts to learn the best feature weights for the problem-at-hand. Branting describes a method of learning feature weights by looking at customer selections from sets of presented items. This method is called *LCW* (learning customer weights), and involves boosting the ranking order of selected items by altering the feature weights. These techniques are broadly similar; in each of these approaches there is an attempt to learn a utility function by altering feature weights to improve retrieval accuracy. The learning techniques themselves are also similar, using a combination of failure and success-driven approaches.

Our system uses implicit user feedback – the final selection of an item for purchase by the user – to drive the learning process. It is a failure driven approach; if the system is making good recommendations there is no attempt to improve the retrieval mechanism. Because we only use the selection of a single item by the user as our retrieval mechanism we cannot look at the overall case order feedback as Stahl does, instead we use a technique more closely aligned with Branting's work. We also examine ideas from Bonzano's work with relation to the issue of contextual features. The fundamental differences between our work and other work in the area are:

- There is no a priori knowledge about the items being recommended (apart from their expected structure) as the items are being retrieved in real time. All that is known is that all items will completely satisfy the user's initial query.
- Each user of the system acts as her own similarity teacher and the learnt feature weights are stored in her personal profile.
- Learning is attempted on both stages of the recommendation process; session retrieval and final offer recommendation.

These techniques all concern introspective learning of feature weights, but there are alternative techniques available which we intend to evaluate with further work, e.g. feature selection rather than feature weighting. There has also been work done on the problem of learning local similarity measures – the similarity measure for each individual feature – e.g. [15], but in this work we have confined ourselves to the learning of the feature weights.

3 Recommending Travel Offers

The main purpose of the PTA is to take a user's request for flights, contract with real online flights brokers for travel solutions and recommend the best of these to the user. Since the flights come in from real, external sources, their details cannot be known in advance. For the purposes of a demonstration, consider the plight of a user making a request for a holiday trip from Dublin to Rome. On making the request the user is faced with choosing flights from a set of forty-nine offers (twenty-four outgoing and

twenty-five return flights). The following list of feature value possibilities illustrates the diversity of the outgoing offers set:

- Two carriers
- Two destination airports (Ciampino and DaVinci)
- Price ranging from €52 to €112
- Departing as early as 06:30 and arriving as late as 23:45
- Single flight trips and two hop trips. Among the multiple-hop set (of which there are twenty-three in this set) there are the following additional choices:
    - Four possible stopover airports
    - Stopover times ranging from two hours up to 12:30 hours

The size of this set is not atypical of the scenarios encountered by users of the PTA. In fact our users have average return-set sizes of more than 24 offers. Some requests yield much larger sets, e.g. London to Milan - 79 offers; Dublin to London - 73 offers. With this degree of freedom the idea that a single feature would override all others and offer the user the ability to manually search the set by sorting by a single feature is inadequate. This is why a recommender system is needed to reduce the offer set to a more manageable size. The remainder of this section describes the recommendation process in the PTA.

The PTA makes recommendations by looking at interactions the user has had with the system in the past. By using the selections the user made in similar sessions in the past we hope to make good recommendations in future sessions. To do this we need to store information about the user's habits. After every successful user interaction with the PTA (i.e. after the user has selected a flight and is forwarded on to the booking page), we record data about the request (e.g. origin, destination, departure date) in the form of a session-case. We also store a reference to the offer that was selected by the user. In this way, the request features represent the problem, and the selected offer the solution of the session-case. We represent offers as cases in the second stage of the recommendation process (offer-ranking). This allows us to rank the current set of offers based on their similarity to the offer-case referenced in the retrieved session-case.

We will illustrate the recommendation process by describing the steps taken in an example where a user makes a request for a flight from Dublin to Rome. The user logs into the PTA and submits a form containing details of the origin, destination, dates of travel and number of tickets required. The PTA decomposes this request into its constituent parts and forwards these on to a number of online travel brokers. It then composes the responses into a number of travel offers. These offers make up the current offer case-base.

At the same time, the PTA searches the user's session case-base for the session with the most similar request to the current one. It then uses the selected offer from that session to rank the offers in the current offer case-base. In this example, the retrieved session contained a request for a trip to Milan made by the user two months earlier. In that previous session, the user selected a cheap two-hop trip via London Stansted with a short stopover time. Therefore, the PTA will tend to recommend similar offers from the current offer case-base, e.g. cheap flights with a short stopover, preferably in Stansted.

Both stages use CBR and as such are dependant on the definition of good similarity measures. Traditionally, CBR systems have depended on domain experts to design similarity measures. However we have implemented a process whereby our users "teach" the system their personal preferences which are incorporated into both the session case-base and the similarity measures.

Unless the user constantly selects offers on the basis of a single feature (which we observe not to be true) it is important to gauge the relative importance of feature similarities in order to offer better recommendations. We describe the relative importance of features using feature weights in the similarity measure and describe our technique to learn them in the following section.

In summary, the recommendation process follows two stages:

i. Context-Matching: Finding the session with the most similar request from the user's session case-base and retrieving the selected offer from that session
ii. Offer-Ranking: Using that offer to rank the offers in the current session. The highest ranked offers are then presented to the user

4 Personalising Recommendations – Learning Feature Weights

As mentioned in Section 3, the recommendation process involves two similarity measures. We begin by describing our algorithm for learning feature weights for the second stage of recommendation, i.e. offer-ranking. The similarity between a previously selected offer, S and a current offer, C is given by the sum of the similarities of their constituent features (σ_f) multiplied by their respective weights:

$$Sim(S,C) = \sum_{f \in F} w_f \times \sigma_f(sf, cf) \quad \textbf{(1)}$$

Where F is the set of features that can occur in a case. We infer the relative importance of features by comparing our predicted recommendations with the actual selections of the users. When we run the recommendation process on a set of offers, we end up with an ordering across the set. By comparing the eventual selection of a preferred offer by the user with this ordering we can generate a recommendation error, E_{rec} as follows:

$$E_{rec} = \frac{index - 1}{N - 1} \qquad \forall N : N > 1 \quad \textbf{(2)}$$

Where N is the size of the offer set and *index* is defined as the ranking of the offer the user selected. However, if the offer is ranked equally with other offers, *index* is the lowest ranking of the equals (e.g. if the selected offer is ranked equal third with four other offers *index* is six – two higher ranked plus four equal ranked offers). In this way, a recommendation error of 0 would indicate that the recommender system recommended the selected offer above all other offers; this is to encourage the PTA to minimize the number of offers it must present to the user. Sessions with return flights have two retrieval accuracies, one for the outgoing and one for the return offers. The overall retrieval error for a user, E_{user} is the mean of their recommendation errors.

Table 1. Table 1 shows the selected offer from the previous session (*S*) and two of the offers from the current offer set (C_1 and C_2). C_1 was recommended above C_2 but the user selected C_2. Our algorithm should alter the feature weights so that C_2 would be ranked above C_1 if the recommendation process were executed again.

	S	C_1	C_2
Trip Details	Dublin->Milan	Dublin->Rome	Dublin->Rome
Price	*€50*	*€60*	*€85*
Stopover location	*London Stansted*	*London Stansted*	*London Stansted*
Stopover Time	*125 minutes*	*240 minutes*	*150 minutes*
Selected	N/A	not selected	selected

To improve our recommendation accuracy we must alter the feature weights in such a way as to ensure that the selected case is ranked higher than the other offers. To this end we calculate the local similarity difference, $\Delta\sigma_f$ for cases ranked lower than *index*. The local similarity difference for the feature of a case (c_i) is the difference between its local similarity score and that of the selected case. We then find out which cases ($w_{learn}(C_1) = true$) could be ranked lower if we altered the feature weights:

$$w_{learn}(C_i) = false \text{ if } \Delta\sigma_f <= 0 \; \forall f \in C_i \tag{3}$$

$$\text{where } \Delta\sigma_f = \sigma_f(s, c_{index}) - \sigma_f(s, c_i) \tag{4}$$

The local similarities and similarity differences for the example in the last section are shown below in Fig 1. We are now faced the decision of which case to use to drive learning. We have achieved good results when using the highest ranked learnable case (i.e. with $w_{learn}(C_1) = true$) to drive learning. Returning to our example scenario, we can use offer C_1 to improve the ranking of C_2 with respect to C_1. We use the following equations to change the weights on price and stopover time:

$$w_f = w_f \times increment \quad \text{if } \Delta\sigma_f > 0 \tag{5}$$
$$w_f = w_f / increment \quad \text{if } \Delta\sigma_f < 0$$

The *increment* parameter was set to 1.1 in the results presented here. Weights are then normalised and the whole process is repeated until the algorithm passes an iteration limit or one of the following stopping criteria is met:

$$index == 1 \tag{6a}$$

$$w_{learn}(Ci) == false \; \forall i < index \tag{6b}$$

Criterion 6a indicates that we have reached the overall optimal solution and that no further learning is necessary; criterion 6b indicates that no further improvement is possible. As we iterate through the algorithm we check for an improvement (i.e. a reduction) in *index*. If this occurs we save the feature weights as the current best. When one of the stopping criteria is met, we save the current best weights for that session (session-weights).

There are two ways we can use these session-weights. The PTA could store a single set of weights for each user (user-weights) and could incorporate the session-weights

into the user weights after every completed session. Alternatively, it could store a reference to these session-weights with the session-case in such a way that the ranking preferences are viewed as part of the context of the session. We have performed experiments that confirm that it is better to store session-weights with each session-case than to store user-weights for each user (these results are presented in Section 5). This reflects the fact that people have different ranking and selection criteria under different contexts.

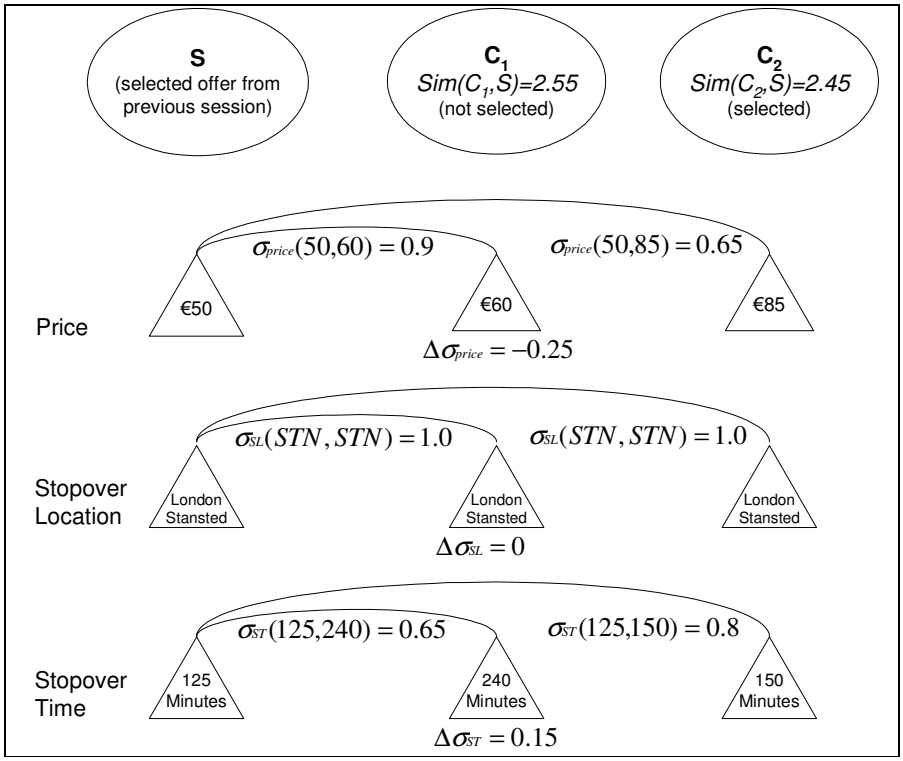

Fig. 1. This figure shows the local similarity scores for the Dublin to Rome example. Details are shown for the features price, stopover location and stopover time. In this example the user selected the second highest recommended feature so *index* is 2. The similarity differences for case C_1 are shown beneath its feature values, i.e. $\Delta\sigma_{price}$=*-0.25*.

Learning the Request Similarity Measure

We use a different similarity measure for finding the most similar previous request. We intend to apply the same learning techniques for this similarity measure. However there is a fundamental difference in how this learning is driven. As we use a failure driven approach, we can only trigger learning on the request similarity measure when we are unable to learn an optimal set of feature weights in the offer recommendation stage, i.e. when we are unable to find a set of feature weights such that *index = 1*. When this occurs, we believe that the problem is not with the learning algorithm but

with the most similar request; i.e. that the context of the most similar request was different from the context of the current request. We search through the user's sessions to see if there was another session that would have yielded a better recommendation and attempt to alter the request feature weights to improve the recommendation score of the selected offer. This mechanism is more computationally expensive than simply learning feature weights at the offer granularity, and care must be taken to ensure that a change in the measure does not affect the accuracy of earlier sessions. We are currently in the process of implementing this algorithm and so have no results to present at this point.

5 Results

The PTA has been up and running since December 2003, but due to the nature of the domain, there is a dearth of sessions. This is because the average user will only make a few requests every year. To overcome this, we created a number of travel scenarios and asked people to complete them using the system. One such scenario was to make plans for a holiday to one of a list of destinations for any duration between five days and fourteen days. These scenarios were chosen to guarantee a large number of possible solutions with diversity in the offer sets. Each user was given six scenarios to complete.

The emphasis on this evaluation was to make the data as realistic as possible. With this in mind, users were given the freedom to choose their own destination and were allowed to reject a whole set of offers and make a totally new request if they were not happy with any of the offer set. The key point to note is that selected offers were considered by the user to be the genuinely preferred offer from the presented set. Many users had also completed sessions on their own initiative, and purchased real offers. These "real" sessions are included in our evaluations.

We use an offline-technique to evaluate our approach to feature weight learning. This involves simulating interactions with the system using the PTA's history of user-interactions. In the first evaluation, we go through every session in the history and calculate a set of session-weights using the techniques outlined in Section 4. We store these weights with the session-case.

The evaluation proceeds as follows: we use a leave-one-out approach whereby we remove one session from the user's session case-base and treat it as a new (unseen) session. This session contains a travel-request and a set of offers that were viewed by the user as well as her eventual selection of a preferred offer. We retrieve the most similar session-case from the user's session case-base. With the referenced offer-case and the optimal set of session weights that we learnt for that session we can calculate a ranking order on the current set of offers. By comparing this ordering with the user's actual selection we calculate a recommendation error for that session. We do this for every case in the user's session case-base and calculate an average recommendation error for each user. We plot the recommendation accuracies against a baseline accuracy for each user of the system in Fig 2. The baseline is calculated by using the same techniques as above except that we do not use session weights but weight features

equally in the ranking process. Fig 2 shows that the recommendation process using learnt session-weights is significantly more accurate than using equal weights for every feature.

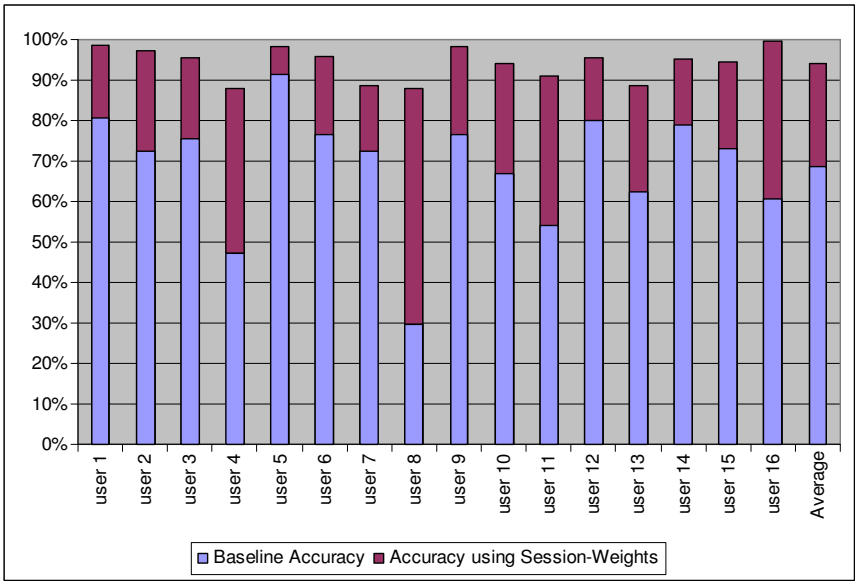

Fig. 2. A comparison of the baseline recommendation accuracy against the session-weight recommendation accuracy for each of the sixteen users. The average accuracies of all users are shown in the far right column. The improvement in accuracy is statistically significant at the 99.99% confidence level.

Our second evaluation assesses the value of user-weights in the recommendation process. User-weights are an amalgamation of session-weights and provide a better level of generalization. The user-weights are the average of each user's session-weights as calculated in the previous evaluation. A comparison of the recommendation accuracies in the previous evaluation (i.e. baseline and session-weight recommendation accuracies) against user-weights is shown in Fig 3. This shows that session-weights offer a significant improvement in recommendation accuracy over user-weights. We see the fact that session-weights are more valuable than user-weights as proof of over-generalization and a confirmation of our premise that context is an important element in this domain.

Over-fitting

It is important at this point to mention the problem of over-fitting. Previous research has shown that feature weighting algorithms tend to over-fit the data [8]. We believe that over-fitting is unavoidable in this domain due to the lack of data. However, our leave-one-out evaluation shows that the learned weights are still better that the starting position of equal weights. As more data is collected, we intend to perform an evalua-

tion of the level of over-fitting that is occurring in our algorithm and to attempt to minimise it.

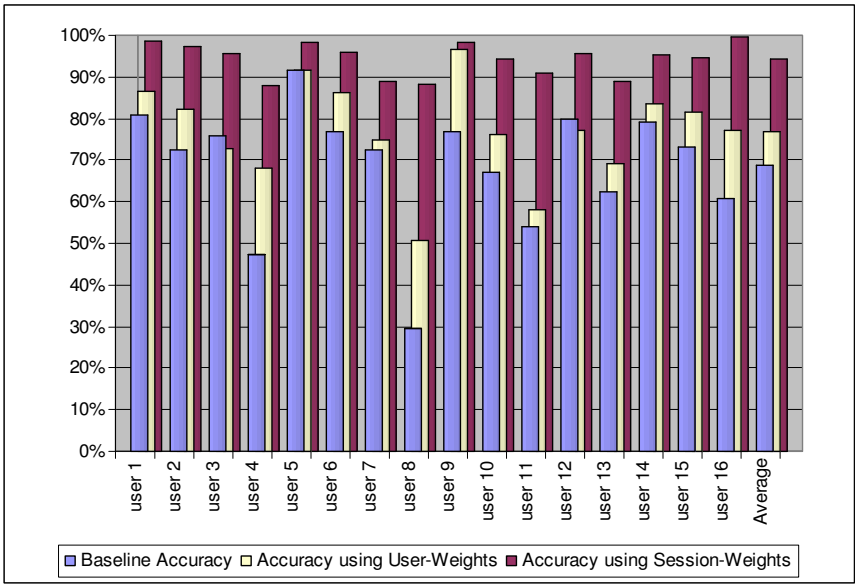

Fig. 3. A comparison of the baseline recommendation accuracy, the session-weight recommendation accuracy and the user-weight recommendation accuracy for each of the sixteen users. The average accuracies of all users are shown in the far right column. The improvement of learning session-weights over user-weights is statistically significant at the 99.99% confidence level.

6 Future Work

We intend to develop our approach to making good recommendations to users of our PTA system. We are currently implementing the mechanism for learning the request feature weights and hope to use this to further improve our recommendations. We intend to implement further techniques including Collaborative CBR and learning the local similarity measure (in the same manner as work done by Stahl [15]).

Collaborative CBR

Because of the lack of session information, we intend to investigate collaborative techniques to improve our recommendations. When a user makes a request, and the retrieved session's similarity score is below a threshold, we look to the user's neighbours for a better match. If a similar request is found from a neighbour's session

case-base we use their experience to recommend offers from the user's current session. This is especially useful for new users.

The main issue with this approach is the determination of a user's neighbours. We intend to do this by comparing the feature weights of each of the users and group users together by virtue of the similarity between their weights. We will strengthen these groupings if collaboration leads to good recommendations and vice versa. This solution is appropriate for users with a rich history and well learned weights; however we are still faced with the problem of determining neighbours for new users. To solve this we intend to allow new users access to the collective case-base of the system.

Altering the similarity measure

The focus of this paper has been on the learning of feature weights, but there is also scope to learn local similarity measures. In fact this is also happening in this evaluation to a small extent. The PTA uses a taxonomy difference function to capture the relationships between geographical locations (the origin and destination features in the offer-cases use this representation). However, due to the configuration of this taxonomy there will never be diversity in similarity among the set of current offers, since all airports in a city are at the same level in the taxonomy. To allow us to perform learning at this level, we reorder the taxonomy by boosting the selected feature value above its siblings, thus incorporating a measure of utility into the local similarity function.

This is only one example of how local similarity functions can be altered to incorporate utility; another way is to alter the sensitivity to difference. Many of the features in this domain use numeric difference as the basis for similarity calculations, by altering the user's sensitivity to difference we can implement further personalisation. The similarity graph for the price feature is shown in Fig 4; by changing point {200, 0} to {100, 0} we would focus the price similarity measure on cases with differences of less than €100. A more in depth description of our representation of similarity measures is given in [7].

7 Conclusions

This paper described our approach to learning personalized feature weights in an online travel recommendation system. The key motivation in developing a good recommender system is to combine good recommendations with a low cognitive load on the user. We have achieved good recommendation results by implementing personalised profiles, with learning algorithms specific to each user. We minimise cognitive load by using implicit feedback to drive this learning. Our recommendation process is based on Case-Based Reasoning, and we learn a user's profile in two ways; by adding cases to their case-base with every interaction, and by learning optimal sets of feature weights for each interaction.

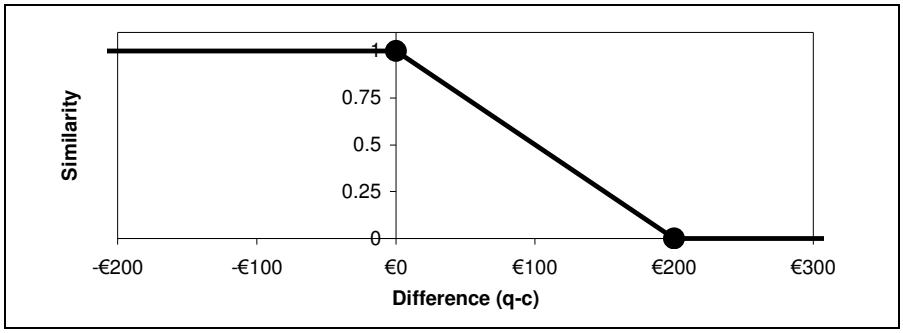

Fig. 4. Fig 4 shows the relationship between a difference in price between S and C. A difference of greater than €200 results in a similarity of zero.

Our motivation for learning in this way is that users enter each interaction with a different context. In the offer recommendation process, the defining context is the request itself, it is clear that a user's preferences with respect to a long haul flight will be quite different than for a short trip, e.g. price may become less important, and comfort may be the defining feature. It is these preferences that we are trying to learn. We have performed evaluations of our techniques with real users of the system that show a highly significant improvement in recommendation accuracy with our learning algorithms.

One interpretation of our techniques is that they are geared towards reducing the return set size in response to a request. If we present a subset of the total number of offers we cannot offer a guarantee that our system will present the most suitable offer in the first retrieval. For this reason we see the potential for our techniques to operate in parallel with other recommendation strategies such as comparison-based recommendation [9] and diversity boosting [3] in a mature recommendation system.

References

1. Bergmann, R., Richter, M. M., Schmitt, S., Stahl, A., Vollrath, I. (2001). Utility-Oriented Matching: A New Research Direction for Case-Based Reasoning. Proceedings of the 9th German Workshop on Case-Based Reasoning, GWCBR'01, Baden-Baden, Germany. In: H.-P. Schnurr, S. Staab, R. Studer, G. Stumme, Y. Sure (Hrsg.): Professionelles Wissensmanagement. Shaker Verlag. pp. 264-274.
2. Bonzano, A., Cunningham, P., Smyth, B. (1997) Using introspective learning to improve retrieval in CBR: A case study in air traffic control. Proceedings of the 2nd International Conference on Case Based Reasoning (ICCBR-97), David B. Leake, Enric Plaza (Eds.), LNCS 1266 Springer pp 291-302 1997.
3. Bradley K. & Smyth B. (2001). Improving Recommendation Diversity. Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, D. O'Connor (ed.) pp85-94, 2001.
4. Branting, L. K. (2003). Learning Feature Weights from Customer Return-Set Selections. The Journal of Knowledge and Information Systems (KAIS) 6(2) March (2004)

5. Burke, R., Hammond, K., & Young, B. (1997). The FindMe Approach to Assisted Browsing. IEEE Expert, 12(4), pages 32-40, 1997.
6. Coyle, L., Cunningham, P. & Hayes, C. A Case-Based Personal Travel Assistant for Elaborating User Requirements and Assessing Offers. Proceedings of the 6th European Conference, ECCBR 2002, Susan Craw, Alun Preece (eds.). LNAI Vol. 2416 pp. 505-518, Springer-Verlag 2002.
7. Coyle, L., Doyle, D., & Cunningham, P. (2004) Representing Similarity for CBR in XML. To appear in the proceedings of the 7th European Conference on Case Based Reasoning, ECCBR 2004.
8. Kohavi, R., Langley, P., Yun, Y., The Utility of Feature Weighting in Nearest-Neighbor Algorithms. , 9th European Conference on Machine Learning ECML-97, Prague, Czech Republic. Poster session.
9. McGinty, L., & Smyth, B. (2002). Comparison-Based Recommendation. Proceedings of the 6th European Conference, ECCBR 2002, Susan Craw, Alun Preece (eds.). LNAI Vol. 2416, pp 575-589, Springer-Verlag, 2002.
10. Ricci, F., Mirzadeh, N. & Venturini, A. (2002). ITR: a case-based travel advisory system. Proceedings of the 6th European Conference, ECCBR 2002, Susan Craw, Alun Preece (Eds.). LNAI Vol. 2416, pp 613-627, Springer-Verlag, 2002.
11. Ricci, F., Venturini, A., Cavada, D., Mirzadeh, N., Blaas, D. & Nones, M. (2003). Produce Recommendation with Iteractive Query Management and Twofold Similarity. Proceedings of the 5th International Conference on Case-Based Reasoning, ICCBR 2003, Kevin D. Ashley, Derek G. Bridge (Eds.). LNCS Vol. 2689, pp479-493 Springer 2003.
12. Shimazu, H. (2001). ExpertClerk: Navigating Shoppers' Buying Process with the Combination of Asking and Proposing. Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI-01, Seattle, Washington, USA.
13. Stahl, A. (2001). Learning Feature Weights from Case Order Feedback. Proceedings of the 4th International Conference on Case-Based Reasoning, ICCBR 2001, David W. Aha, Ian Watson (Eds.). LNCS Vol. 2080 pp502-516 Springer 2001
14. Stahl, A. (2002). Defining similarity measures: Top-Down vs. bottom-up. Proceedings of the 6th European Conference, ECCBR 2002, Susan Craw, Alun Preece (Eds.). LNAI Vol. 2416, pp 406-420, Springer-Verlag, 2002.
15. Stahl, A., Gabel, T. (2003). Using Evolution Programs to Learn Local Similarity Measures. Proceedings of the 5th International Conference on Case-Based Reasoning, ICCBR 2003, Kevin D. Ashley, Derek G. Bridge (Eds.). LNCS Vol. 2689, pp 537-551, Springer 2003.