Exploiting Re-ranking Information in a Case-Based Personal Travel Assistant[1]

Lorcan Coyle and Pádraig Cunningham

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

Abstract. Intelligent software assistants are becoming more common in the e-commerce domain. We are working on a personal travel assistant. The goal of this application is to use case based reasoning to assist the user in arranging flights. It offers personalised service to its users and automatically learns their travel preferences. It stores these preferences in a user model that is directly related to the CBR process. It learns the user preferences by exploiting user feedback on sets of presented travel offers. When the user selects a preferred offer, the PTA establishes a preference ordering among the whole set. This ordering is calculated by measuring the similarity between the selected offer and each of the other offers. This ordering is used to rate these offers and store them in the user profile as cases. This ordering is also used to refine the user's overall travel preferences by altering their personal similarity measure.

1 Introduction

Most e-commerce stores sell products without the intervention of a human sales assistant. In the absence of human sales assistants there is a need for intelligent software assistants to facilitate the sales process. Most of these assistants help the user to search through a catalogue of products to find an appropriate item. The user makes an initial request for an item and is assisted by the sales assistant until the optimal item is presented to the user. The main goal of these assistants is to minimise the cognitive load on the customers. Assistants that bore or frustrate the customer risk driving them to competing stores, which are never more than one click away. One way to maximise the user's satisfaction is to provide a personalised service. This is achieved if the assistant learns and applies the user's personal preferences in its interactions with the user. These preferences can be discovered by asking the user to explicitly rate individual items or to explain their motivations behind making a particular selection, however we feel that this goes against the goal of reducing cognitive load. Hence our goal is to learn these preferences implicitly and apply them surreptitiously.

There is much work being done on learning a user's ranking criteria by observing their selections from a list of presented items [15,4,10]. Stahl uses a technique called *case order feedback* [15] and Branting uses a similar approach called *LCW* (Learning Customer Weights) [4]. These approaches compare the user's preferential ordering of a set of presented items with the recommender system's predicted ordering. They attempt to minimise the error between the predicted and observed ordering by altering the recommender system's similarity measures. One major advantage of using user feedback is that it incorporates utility into the similarity measure.

Our work centres on the development of a Personal Travel Assistant (PTA). This is an intelligent sales assistant that helps the user in making flight arrangements in the travel domain. This application is based on a scenario developed by the Foundation for Intelligent Physical Agents (FIPA) in the travel domain [8]. Our application takes a travel request from the user, forwards it to a number of real-world travel agents. These return a number of suitable offers and the PTA recommends the best of these to the user and assists in the selection of an optimal offer for purchase. This paper documents our approach towards the elicitation of individual user preferences to better complete this offer recommendation task.

There are a number of domain specific constraints that should be noted before we explain our implementation. The first is in dealing with the real-world travel providers; these take their requests in a pre-defined manner, there are a number of parameters that must be filled with each request. This restricts the type of requests the PTA can make on behalf of the user. The second is that there is a turn-

[1] The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged.

around time cost associated with making a request to a real-world broker; we have observed these delays to be as much as 30-40 seconds. These factors rule-out the use of a pure iterative conversational CBR approach since the request is not negotiable, and the user is unlikely to have the patience to continually refine a search until the optimal offer is found. To overcome this, we make the assumption that all users of the system are willing and able to make a broad initial travel request with the standard parameters required by online travel agents (i.e. origin, destination, departure date and number of tickets required).

Another major issue with our implementation is that we are targeting our application at mobile web users, e.g. WAP enabled phones and mobile-PDAs users. These devices are constrained in the amount of screen-size they have. This means that we have to be even more sensitive to the information overload problem and minimise the click-distance of the user to their optimal offer. This reinforces our desire to reduce cognitive load on the user. Ideally we would like to present the user with a small group of offers with maximal utility. They could then search through the offers by using a MLT type of approach [12] by using this initial set of offers as a jumping off point.

Section 2 describes our application and how CBR is used in the recommendation of travel offers to the user. It also mentions the possibility of using a collaborative CBR approach to improve problem coverage. Sections 3 and 4 describe the automatic learning techniques in this application. First we describe how implicit user feedback is used to rank travel offers (Section 3) and then we describe how overall user preferences are learned by the PTA based on these rankings (Section 4). Finally, in section 5 we conclude the paper and outline some further work that may be done to further this research.

2 Description of the Application

This section describes the architecture of our PTA application and is described in more detail in [6]. Our PTA assists the user in the planning and selection of flights and acts as the user's proxy when dealing with real world travel providers. It takes travel requests from users and negotiates with broker agents for suitable travel solutions[2]. In order to become efficient at its job, it should be able to learn the user's travel preferences and apply this knowledge when brokering offers. Since the PTA acts on behalf of many users it must build a user model or profile for every user. This profile is made up of two types of information, a representation of the user's preferences with respect to previously viewed travel offers, and a representation of the user's overall travel preferences. With every user-interaction the set of viewed travel offers is increased and the overall travel preferences are refined. This refinement is explained in greater detail in the next section.

The user interacts with the PTA via a simple web interface. A user interaction consists of a dialog between the user and the PTA. This dialog begins with the user making an initial travel request. This returns a large number of matching travel offers. These offers are ranked and filtered by the PTA according to the user's preferences. This reduced set of offers is presented to the user who selects their preferred offer. The PTA then connects the user to the broker that made the original offer and bows out of the transaction.

The CBR Solution to Recommending Offers

Personalisation on the PTA is achieved using CBR. The PTA builds a model of each user of the system. This model consists of two parts; a set of cases representing previously viewed travel offers and user's responses to them; and a description of the user's overall travel preferences in the form of similarity measure information. The personalisation process can be broken down into three parts: the ranking of received offers with respect to users' preferences; the storage of presented offers and their ratings (as new cases); and the refinement of users' profiles with respect to their feedback on presented offers.

When a user makes a new travel request, the PTA compiles a case base of similar offers to the current query and sends the request to the broker agents. As offers are returned, they are individually converted into target cases and passed into the CBR module. The CBR module returns the most similar case(s) to the target case. This case will have a user rating attached to it and the PTA uses this to give a recommendation score to the current offer. When all received offers have been scored in this way, the ones with the highest recommendation scores ones are sent to the user as recommended travel offers.

[2] See Appendix A for a description of how the broker interacts with real-world travel brokers.

The next stage of the interaction involves the user selecting an offer or making a modified request. Assuming that the user selects a flight and proceeds to book it, we can draw some inferences about their preferences with regard to the currently presented offers. This inference is explained in more detail in section 3. Finally, we attempt to draw inferences from this selection into the user's overall travel preferences. This is described in more detail in Section 4.

Collaborative CBR
CBR is only useful in solving a problem if the case base offers good coverage for that problem. If a user encounters a problem outside their previous experience we borrow cases from her neighbours and use a collaborative CBR [11] approach. This should increase the probability of finding a good solution. We use this collaboration approach only when the user's own case base provides inadequate coverage. We calculate problem coverage by looking at the degree of similarity between the current problem case and the most similar cases from the user's own case base. A high level of similarity would indicate good coverage for the current problem. It should be noted that this approach will remove the bootstrap problem encountered by new users.

3 Learning User Preferences on Presented Items

In the last section we discussed how we intend to recommend new offers to users by using a CBR approach. This approach depends on the user rating all viewed offers. However, since one of our goals in this application is the reduction of cognitive load on the user we want to measure the user's ratings implicitly. We do this by observing the user's selections from the presented set and establishing a preference ordering between them. We then use this ordering to estimate ratings for each offer. This type of feedback works at a local, case level and tells us little about the users overall travel preferences. There follows a description of our approach to learning local user preferences by observing user feedback to presented sets of cases. We then describe how we use this ordering to rate presented offers. We define the preference ordering as follows: If the user prefers offer *i* (O_i) to offer *j* (O_j), we define the following order:

$$O_i > O_j \tag{1}$$

Consider a set of offers that were presented to the user $S = \{O_1, \ldots, O_n\}$, If the user selects the i^{th} offer (O_i) we can determine the following preference order:

$$O_i > \{O_1, \ldots, O_{i-1}, O_{i+1}, \ldots, O_n\} \tag{2}$$

However, this tells us nothing about the relationship between the offers that weren't selected by the user. By looking at the similarity of each of the offers to the selected offer we can estimate a preferential relationship between them. If we reorder the set of offers according to each offers similarity (*Sim*) to O_i, we get the following preference relationship:

$$O_1 > O_2 \text{ if } Sim(O_1, O_i) > Sim(O_2, O_i) \tag{3}$$

This estimation is a good indication of the user's real preference ordering because our similarity measure incorporates offer utility (more of this in Section 4). Confirmation of this will require evaluation with real user data. Now that we have a preference ordering we need to map it onto real ratings. Our system uses a rating scale of zero to ten, where ten is the highest possible rating score. The offer with the highest similarity, $Sim_{highest}$, is given a rating score of ten and, the offer with the lowest similarity, Sim_{lowest}, is given a score of zero. The other offers are mapped onto the scale using the following formula:

$$Rating_i = 10 \times \frac{Sim_i - Sim_{lowest}}{Sim_{highest} - Sim_{lowest}} \tag{4}$$

Since the PTA made a broad search for travel offers on behalf of the user, we can expect to get a large number of offers. Even after passing these offers through our recommendation system we can expect to have a large set of recommended offers. In order to maximise coverage of the set of all offers we propose to impose a diversity constraint [3] on this initial set (it should be noted that we only intend to impose this constraint on the initial set of offers). The PTA then allows the user to either accept an

offer or use a *more-like-this* (MLT) approach such as those proposed by [12,5,1] that will allow the user to search through the available offers to find the optimal offer. By recording the iterative selections made by the user in their search, we can rate all offers that have been viewed. We intend to supplement this MLT strategy with a *less-like-this* (LLT) strategy [12], e.g. a user could ask for more offers that are more like offer j (O_j) and less like offer k (O_k). We use the same ratings function as with the pure MLT strategy but we change the underlying similarity functions. We create a new similarity function ($Sim_{i\ not\ j}$), which is equal to offer i's similarity to offer k (the least preferred offer) minus its similarity to offer j (the preferred case), i.e.:

$$Sim_{i\ not\ j}(O_i, O_j, O_k) = Sim(O_i, O_j) - Sim(O_i, O_k) \quad (\mathbf{5})$$

We need to consider the possibility that none of the brokers returned an optimal offer. In this case our PTA will have to broaden the original travel request. In doing this we force the user to complete a new travel request and to endure the delay of waiting for the brokers to fetch a new batch of offers. Clearly this inconveniences the user but we see it as an inherent problem with the domain.

It is important to once again mention that the target platform for this application is the wireless terminal. As the abilities of these mobile devices increase it might be appropriate to distribute some of the CBR tasks to the client side (like [9,17]), e.g. the MLT search algorithm. The PTA could then be responsible for the computationally expensive data maintenance and automated learning tasks. It would recommend a large set of offers and send them to the client, which would allow the user to hone in on the optimal one. At this point, the client would return the user's selections and the transaction would continue as before.

4 Automated Learning of Overall User Preferences

Section 3 described our approach to learning the preference relationship between the travel offers presented to the user. It showed how we use this relationship to rate offers implicitly. However, it should be noted that this approach operates on a local basis and cannot tell us anything about a user's underlying preferences with respect to why one travel offer is preferable to another. By altering these preferences, we will improve our similarity retrieval mechanism. This section describes how we use local user preference relationships to learn these overall user preferences at the feature level.

Before we talk about how our system learns overall user preferences, we should give some information as to the structure of a travel offer case. Our travel offer case contains n features, the most important of which are the origin, destination, travel time, distance, number of hops and price. In order to calculate similarity between a query and a case (or between two cases) we use a similarity measure that combines the n local similarity measures (at the feature level) and feature weights (that represent the relative importance of features):

$$Sim(O_j, O_k) = \sum_{i=1}^{n} w_i \times Sim(f_i^j, f_i^k) \quad (\mathbf{6})$$

We mentioned in the introduction that our user profile contains the user's overall travel preferences. This information is a representation of that user's similarity measure. This information is stored in CBML format [7], an XML-based case representation language. CBML allows us to store the case structure and content in separate documents. It also allows for the creation of a user-profile document. This document allows us to specify certain similarity measure parameters for each user, i.e. feature weights and local similarity function parameters.

To optimise this function we can learn and alter the feature weights (w_i) and the local similarity measures. There has been much work done in this area [14,15,16,4,10] that is applicable to our application. Branting describes a procedure for learning users' preferred feature weights with respect to item recommendation by observing their selections from return sets. This approach can be summarised as follows: whenever a user makes a request for an item, the system recommends a set of items. These items are ordered by their recommendation score, and one or more of them is considered optimal by the system. If the user selects a non-optimal item, the system's prediction was flawed. To correct this, the system alters its feature weights such that the systems recommendation of optimal item matches the user's selection. Stahl et al. approach is one step further and attempts to learn both the feature weighting and local similarity measures.

Over time users' profiles diverge from the average preferences of the population of the system depending on their individual preferences. The CBML specification allows user profiles to be inherited

from one another. We take advantage of this by creating a default profile that represents the average preferences of the total user population. We allow new users to inherit from this and as their preferences are learned they will diverge from these and their profiles will contain further values. Fig. 1 shows an example CBML user profile documents and illustrates this divergence.

```
<domain name="pta">
    <profile username="default">
        <relevance>
            <feature path="/traveloffer/source" value="2"/>
            <feature path="/traveloffer/price" value="8"/>
            <feature path="/traveloffer/origin" value="5"/>
            <feature path="/traveloffer/destination" value="5"/>
            <feature path="/traveloffer/departuredate" value="8"/>
            <feature path="/traveloffer/distance" value="1"/>
            <feature path="/traveloffer/flighttime" value="1"/>
            <feature path="/traveloffer/hops" value="12"/>
        </relevance>
        <similarity>
            <feature name=”/traveloffer/hops/hop/carrier”>
                <!—similarity information -->
            </feature>
        </similarity>
</profile>
<profile username="Coyle" extends="default">
    <relevance>
        <feature path="/traveloffer/source" value="0"/>
        <feature path="/traveloffer/price" value="12"/>
        <feature path="/traveloffer/hops" value="10"/>
    </relevance>
</profile>
<profile username="Newbie" extends="default"/>
</domain>
```

Fig. 1. This figure shows an example CBML user profile document. This document describes the feature weights for 3 users, *default*, *Coyle* and *Newbie*. The default profile specifies feature weights on eight features. Both Coyle and *Newbie* extend the default profile. User *Coyle* has overridden three of the feature weights from the default profile. User *Newbie* has never used the system before and doesn't override any of the default profile. Similarity information for the default user for the feature named *carrier* is shown without detail because of the lack of space.

One of the main issues with learning similarity measures is that it is very CPU intensive. Stahl [15] and Branting [4] use error functions to describe the deviation between the predicted case order and the observed case order. This error is minimised by altering the similarity measure parameters. This can be done iteratively, e.g. [4] or by using evolutionary algorithms, e.g. [16,10]. We decided to use genetic algorithms because of their ability to search for an optimum in complex search spaces. Although they may not find the exact optimum, if a cut-off point is imposed a good solution is usually found quickly.

Branting's work on learning feature weights in combination with a MLT search strategy yielded results that suggest that there is little to be gained in observing the user's choices beyond the first iteration of the search [4]. Since our PTA uses a similar approach to this, we should investigate his claim. When the PTA presents an initial set of offers to the user they are a diverse set. Assuming that the user asks to see offers that are more like a particular item, our next set of presented offers will be far less diverse. This suggests that we have most to learn from the user's selection from the first set of offers and supports Branting's results. However, because subsequent sets of offers have lower diversity (since we only impose diversity constraints on the initial set of presented offers) it is possible that we can learn some important preferences with respect to individual features. This is because it is probable that the offers will differ in a very small number of feature values. We propose to do some further investigation into this hypothesis.

5 Conclusions and Future Work

We have been working on a Personal Travel Assistant, an intelligent sales assistant that helps the user in making flight arrangements on behalf of its users. The main task of the PTA is the recommendation of travel offers to the user. The PTA uses case based reasoning to complete this task by comparing new offers with similar previously viewed and rated offers. If the most similar previous offer has a high

rating the new offer will be recommended to the user by the PTA. This task is carried out according to the user's travel preferences. User's travel preferences are made up of two parts: a case base representing previously viewed and rated travel offers; and a CBR profile document describing the user's similarity function (i.e. its component parts: its feature weights and local similarity function parameters). There are three parts to the recommendation task:

- The learning of a user's preference ordering in a set of presented offers
- The implicit rating of viewed offers based on this ordering
- The optimising of overall user preferences based on preference orderings

Since we aim to reduce cognitive load on the user (and due to screen size constraints) we want to automatically rate offers without explicit user feedback. Section 3 describes our approach to the learning of the preference ordering. It then goes on to describe how this ordering is used to implicitly rate the presented offers. These offers and their ratings are then stored in the user's profile as travel-offer cases.

Section 4 describes our approach to the refinement of overall user preferences by using an approach similar to [15,4]. This user-feedback approach to refinement guarantees the incorporation of utility into the similarity measures.

Future Work

Our PTA application is currently capable of accepting requests from users and retrieving travel offers from a single real world broker. We intend to create interfaces to at least one more broker and to provide a user friendly front-end (the current front end is extremely basic, and we are having some problems with handing off the transaction with the user to the broker). Once we have users in the system we will be able to harvest data and do an evaluation. First of all we should mention that we intend to hide our recommendation scores from the user and present offers by departure time (this will make the user blind to our underlying recommendations). Our evaluation will involve a number of tests:

- User Selections: Over time, the observed user selections should closely match our predictions.
- Click Distance: Low click distances to the optimal offer (the final offer chosen by the user) should be indicative of user satisfaction.
- Stabilisation of overall user preferences: if our user preferences stabilise over time, we can be confident that they are representative of the user's real preferences

The automated learning processes and CBR approach to recommending offers described in this paper assume that user's preferences are constant and static. Context is not taken into account by the PTA at any point in a user interaction. It may be appropriate to look into incorporating context into the transaction. For example, if we look at the amount of time between the request time and the travel time we can get an idea of the urgency of the user. This could be an important feature to bear in mind when resolving a request and recommending an offer. By targeting the mobile device we may also have access to more context specific information, e.g. the user's physical location or information relating to the user's schedule that may be stored on the device's calendar.

We mentioned in section 2 that a collaborative CBR approach would be used to share travel-offer cases when a user's case coverage is low. The sharing of cases between users has been well documented e.g. [11]. Less work has been done in the area of sharing the more subtle similarity preferences. Some work in this area has been done by [13] in this area of sharing similarity relationships. We intend to investigate her approach and use it to share elements of our user profile documents. We mentioned in Section 4 that user profiles are inheritable and that all users inherit their base values from the default user. We would hope to combine her approach with the inheritance of user profiles and the grouping of similar users into explicit neighbourhood groupings.

We mentioned in Section 2 that our iterative-search approach could potentially harvest a large number of cases from every user interaction with the PTA. We intend to put some effort into case base maintenance and examine the possibility of only storing a subset of the presented offers from each interaction. This will be especially significant if the cases are extremely similar.

Appendix A: Connecting to the real world flights providers

Our broker component takes a travel request from the PTA and sends it to a number of competent real world travel agents to resolve. The PTA uses a technique known as web-scraping [2] to interact with real world travel providers. This involves faking an interaction between a browser and an online flights provider. The PTA fills in fields of a website travel request form with values specified by the user and

sends it to the web server as a regular HTTP request. When the response is received, the HTML is parsed to get the travel offers.

There are a number of problems with this technique, if the e-store changes its presentation format in any way we will have to re-engineer the PTA's expectations of the web-server's response. More importantly is the problem of allowing the user to purchase an offer they are happy with. One way to do this would be to maintain the user's credit card and delivery details on the PTA but this has serious privacy implications. Fortunately, we are able to maintain sessions on behalf of the user as we make the original travel requests. When the user eventually decides to purchase a flight we can forward them onto the next webpage in the transaction with the e-store. To follow the sales-assistant metaphor, it could be said that our assistant has helped the user find the product they were looking for, brought her to the checkout and stepped aside to allow her to pay for the goods.

References

[1] Aha, D. & Breslow, A. Refining Conversational Case Libraries. Proceedings of the 2nd International Conference on Case-Based Reasoning, ICCBR97, David B. Leake, Enric Plaza (Eds.). LNAI Vol. 1266, pp 265-278, Springer-Verlag, 1997.

[2] Ball, C. Screen-scraping with WWW::Mechanize. http://www.perl.com/pub/a/2003/01/22/mechanize.html

[3] Bradley K., Smyth B. Improving Recommendation Diversity, Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland (2001).

[4] Branting, L. Karl. Learning Feature Weights from Customer Return-Set Selections.

[5] Burke, R., Hammond, K., and Young, B. 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., Cunningham, P. & Hayes, C. Representing Cases for CBR in XML. In Proceedings of 7th UKCBR Workshop, Peterhouse, Cambridge, UK.

[8] Foundation for Intelligent Physical Agents. Personal Travel Assistance Specification (Document No. XC00080). Carried forward from FIPA 1997 Specification 4 V1.0.

[9] Hayes, C., Cunningham, P., & Doyle, M. Distributed CBR using XML. In Proceedings of the KI-98 Workshop on Intelligent Systems and Electronic Commerce, number LSA-98-03E. University of Kaiserslauten Computer Science Department, 1998.

[10] Jarmulak, J., Craw, S. & Rowe, R. Genetic algorithms to optimize CBR retrieval. In Proceedings of the 5th European Workshop on Case-Based Reasoning. LNAI Vol. 2416, pp 421-435, Springer-Verlag, 2000.

[11] McGinty, L. & Smyth, B. Collaborative CBR for Real-World Route Planning. In Proceedings of the 2001 International Conference on Artificial Intelligence (IC-AI'2001) Las Vegas, Nevada

[12] McGinty, L., & Smyth, B. 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.

[13] Minor, M., Wernicke, M. The Exchange of Retrieval Knowledge about Services between Agents. Wissensmanagement 2003: pp. 295-302

[14] A. Stahl. Defining similarity measures: Top-Down vs. bottom-up. In Proceedings of the 6th European Conference on Case-Based Reasoning. Springer, 2002.

[15] A. Stahl. Learning Feature Weights from Case Order Feedback. Proceedings of the 6th European Conference, ECCBR 2002, Susan Craw, Alun Preece (eds.). LNAI Vol. 2416, pp 502-516, Springer-Verlag, 2002.

[16] A. Stahl, T. Gabel. Using Evolution Programs to Learn Local Similarity Measures.

[17] Watson, I. & Gardingen, D. (1999). *A Distributed Case-Based Reasoning Application for Engineering Sales Support*. In, Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99), Vol. 1: pp. 600-605. Morgan Kaufmann Publishers Inc. ISBN 1-55860-613-0