Making Personalised Flight Recommendations using Implicit Feedback

Lorcan Coyle

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy

October 2004

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work. This thesis may be borrowed or copied upon request with the permission of the Librarian, University of Dublin, Trinity College.

The copyright belongs jointly to the University of Dublin, Trinity College and Lorcan Coyle

Lorcan Coyle

Dated: January 10, 2005

Acknowledgements

I would like to sincerely thank my supervisor Pádraig Cunningham for his support and friendship over the last three years. His help on the thesis was invaluable and his direction on the research was always spot-on.

I would like to thank Conor Nugent who generously offered to proof-read some of the more difficult chapters of this thesis. His criticisms were brilliant and his feedback added greatly to the thesis. Thanks also must go to Dónal Doyle who proof-read the CBML Chapter.

I would like to thank all the members of the MLG lab in Trinity College, with whom I shared many great discussions in both the lab and the pub. It is so easy to go to college every day when you work with friends like these: Nadia Bolshakova, Kenneth Bryan, Michael Carney, Patrick Clerkin, Sarah Jane Delany, Dónal Doyle, Chris Fairclough, Derek Greene, Marco Grimaldi, Conor Hayes, Deirdre Hogan, John Loughrey, Brian Mac Namee, Neil McDonnell, Conor Nugent, François-Xavier Pasquier, Matthew Sammon, Alexey Tsymbal, Robert Wall and Gabriele Zenobi. I owe a debt to all of these people, but especially to Conor Hayes who was a great influence on me in the first two years of my time in the MLG, Marco who was always there to discuss an idea over a pint or a coffee, Robert for his saint-like patience for answering any and all technical questions (and for introducing me to web-scraping), and the people I worked alongside on the Fionn/CBML project: Dónal, Sarah Jane, John and Conor Nugent. I would also like to thank all the people in Robert Gordon University in Aberdeen who made me feel at home there in summer 2003, especially Susan Craw and Stewart Massie.

Finally, I would like to thank my family for their love and support and to dedicate this thesis to them. Dad, now you can hold your head high in the club!

Lorcan Coyle
University of Dublin, Trinity College
October 2004

Abstract

As e-commerce has become more popular, the problem of information overload has come to the fore. Recommender systems that reduce the information overload problem are becoming more common. However, the problem with many recommender systems is that they are associated with a high cost of learning customer preferences (in terms of cognitive load). We describe the *Personal Travel Assistant* (PTA), a flight recommender application that uses case-based reasoning (CBR) to overcome these problems.

The PTA allows users to search multiple flights providers concurrently and recommends flights based on their individual travel preferences. These preferences are implicitly learned from observations of user behaviour. When the user purchases a flight, the PTA uses the selection of a preferred flight to discover and refine the user's overall travel preferences. These preferences are stored in a user-model as sets of cases representing their interactions, which are used to provide personalised recommendations.

The PTA makes recommendations taking into account the context in which the flights were offered. It uses features from the request to determine this context, e.g. the duration of the trip. We perform evaluations of contextual recommendations that support our view that user preferences change depending on the context of the session. We further improve recommendation accuracy by storing and personalising similarity measures in the user-model. The PTA alters the relative importance of features in the personal similarity measure based on implicit user feedback, e.g. increasing the importance of price at the cost of stop-over time in a multiple hop flight.

We also investigate cooperative components to extend our recommendation strategies. These allow users to reuse the information learned from other users when they encounter new situations. However, these techniques are not as successful as we had hoped. We discuss these components in relation to other work on collaborative recommendation and suggest that the standard approach is unsuited to the PTA's context-based recommendation strategy.

The strength of CBR in the e-commerce domain stems from its reuse of the knowledge base associated with a particular application. Since case data may be one aspect of a company's entire knowledge system, it is important to integrate case data easily within a company's IT infrastructure, providing in effect a case-based view on relevant portions of the company knowledge base. We describe CBML, an XML-based Case Mark-Up Language we have developed to facilitate

such integration.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In e-commerce the problem of information overload occurs when consumers are overwhelmed with choices. Many e-commerce applications use intelligent recommendation strategies to address this overload problem by attempting to learn each user's preferences. Recommender systems are used to suggest interesting and useful products and provide consumers with information to facilitate the decision-making process (Schafer et al., 2001). They facilitate personalisation on a site by tailoring the sales process to each individual user. Jeff Bezos[1] has said that if Amazon.com has 20 million customers, then it should have 20 million stores.

Personalised recommender systems work by generating a user-model that describes a user's preferences. User preferences can be discovered by asking the user to explicitly rate individual items or to explain the motivations behind making a particular selection. However, this kind of explicit feedback is associated with a high cost (in terms of cognitive load on the user) and runs the risk of boring or frustrating the user. This thesis introduces a recommender that attempts to overcome the problem of information overload while minimising the cognitive load it imposes on its users. It learns user preferences from observations of user behaviour alone — i.e. from *implicit feedback*.

We describe the *Personal Travel Assistant*, a recommender that takes a request from the user and returns a number of suitable flights. These flights are presented as an *offer-set*, ordered by recommendation score. The main motivation behind our research was to implement a recommender system that could make accurate personalised recommendations for its users. Our ultimate goal was for this system to be able to learn user preferences implicitly and apply them surreptitiously. We perform evaluations of our recommendation strategies by comparing these recommendation orderings with the actual purchases of users. If we can predict user selections with our recommendation ordering, then our strategies are successful.

[1]Jeff Bezos is the founder and current CEO of Amazon.com

1.1 The Personal Travel Assistant

The Personal Travel Assistant (PTA) is a recommender system that operates in the flights domain. It is capable of searching multiple real-world flight vendors concurrently for flights on behalf of a user. It has a simple web-interface that allows the user to make a flights request. The PTA takes this request and finds a set of suitable flights, which it presents to the user. Recommendation strategies are necessary in this application because of the large number of flights in an average sized offer-set (30 offers). In order to make accurate recommendations, the PTA should have a good knowledge of the travel area, be able to *learn* each user's travel preferences and apply them to the recommendation process. Every time a user interacts with the PTA, the PTA stores information about her requests and purchases. It uses this information to generate a user-model that represents that user's flights preferences. The PTA uses *case-based reasoning* to drive the user-modelling and recommendation processes.

1.2 Case-Based Reasoning

Case-based Reasoning (CBR) is an artificial intelligence methodology for solving problems by using or adapting solutions to old problems (Riesbeck and Schank, 1989). Previous experiences are retained as cases, which are usually described as problem-solution pairs. The problem represents the description of a previous situation, and the solution represents the action taken (or observed outcome) on that occasion. The collection of cases in the system is called the case-base. In case-based recommender systems the obvious cases are product descriptions. In this work cases represent flights and requests for flights.

CBR belongs to the lazy school of Machine Learning and thus has the defining characteristic of deferring processing to run-time. Whatever the run-time penalties associated with this, it does ensure that the data used for inference is as up-to-date as possible (Aha, 1997). This is of particular benefit in recommender systems, where data are scarce and where data are being updated continuously. The lazy learning approach of CBR also has the advantage that it can model local phenomena well compared to eager techniques that tend to focus on more global models.

1.3 Contributions of this Thesis

This thesis describes the PTA flights recommender system. During its development and operation several issues were encountered. The most interesting ones, which constitute the main contribution of this thesis, are:

- The implementation of a prototype application that allows users to search real-world flights providers concurrently

- The implementation of a user-modeller that elicits user preferences solely from implicit feedback
- The implementation of a two-stage context-based recommendation algorithm
- The incorporation of utility knowledge into the similarity measure
- An assessment of the usefulness of cooperative recommendation techniques in this domain
- The development of a representation format for CBR data called *CBML*

1.3.1 Implicit User-Modelling

If the PTA is to make successful flight recommendations to a user, it must have a grasp of that user's travel preferences. One of the problems with e-commerce in general is that users tend to get bored or frustrated easily, therefore a successful recommender should be able to learn preferences while minimising the cognitive load placed on the user.

The PTA uses feedback from users to generate the user-models that capture their preferences. Explicitly gathered feedback is characterised by a higher level of cognitive load and usually depends on asking users to prioritise criteria by assigning weights or by ranking in order of importance — a skill at which people are notoriously poor. The PTA attempts to infer user preferences from patterns in their behaviour, i.e. using *implicit* feedback. This feedback comes from the requests users make, and the flights they purchase. This thesis describes our attempts to model users' preferences from these data alone.

1.3.2 Context-Based Recommendation Algorithm

This thesis describes a recommendation algorithm that makes the assumption that user flight preferences change depending on the context in which the flights are presented. It uses the context in which an offer-set is sought to drive the recommendation process. This strategy uses features from the flight request as an indicator of context, e.g. whether a request is for a long distance flight or a shorter one. We performed evaluations of this technique that support the notion that user preferences in a session are similar to those learned in sessions with similar contexts in the past, i.e. that context should be an important consideration when making recommendations in this domain.

1.3.3 Incorporating Utility into Similarity

Since CBR uses solutions from *similar* problems in the past the definition of a good measure for calculating similarity is essential. In recent years, there has been some work done in the area of *introspective learning*, whereby the performance of CBR systems is improved by incorporating utility knowledge into the similarity measure by monitoring its run-time progress. We developed some of these techniques and incorporated them into the PTA application.

We describe techniques that correct the similarity measure (specifically the feature weights) in the event of recommendation failures. Recommendation failures occur when a user selects an offer that was not high on the recommendation ordering generated by the PTA. We have applied these improved feature weights in different ways and performed evaluations that showed significant improvements in recommendation accuracy.

1.3.4 Cooperative Techniques

Cooperative techniques are useful in recommender systems when a user-model might not have enough information to make an informed recommendation. The recommender system can use knowledge contained in similar users' user-models to make a better recommendation. Cooperative recommendation is especially useful when making recommendations to new users. We performed an evaluation of a standard cooperative recommendation strategy and suggest that it is not suitable for this application domain.

1.3.5 CBR Representation

Since case data may be only one aspect of a system's entire knowledge base, it is important to integrate case data easily within an application infrastructure, providing in effect a case-based view on relevant portions of the knowledge base. In Chapter 4 we describe Case Based Mark-up Language (CBML), an XML-based language we have developed to facilitate this integration. We detail its benefits in terms of extensibility, ease of reuse and interoperability. The language allows us to make the formal definition of the structure of our cases and similarity measures completely independent of the application code. In this way we allow the structure and definition of our cases to be described and modified easily.

1.4 Publications Related to this Thesis

The publications that are related to this thesis come in three flavours. These first three describe some of the personalisation techniques applied in the PTA application:

- Lorcan Coyle, Pádraig Cunningham and Conor Hayes: A Case-Based Personal Travel Assistant for Elaborating User Requirements and Assessing Offers, in S. Craw and A. D. Preece (eds.), Advances in Case-Based Reasoning, 6th European Conference, ECCBR 2002 Aberdeen, Scotland, UK, 2002, Proceedings, pp 505–518, Springer.

- Lorcan Coyle and Pádraig Cunningham: Exploiting Re-ranking Information in a Case-Based Personal Travel Assistant, in D. Aha (ed.), Workshop in Mixed-Initiative Case-Based Reasoning, Workshop Programme at the Fifth International Conference on Case-Based Reasoning 2003, Proceedings, pp 11–20.

- Lorcan Coyle and Pádraig Cunningham: Improving Recommendation Ranking by Learning Personal Feature Weights, in P. A. G. Calero and P. Funk (eds.), Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 Madrid, Spain, 2004, Proceedings, pp 560–572, Springer.

These next three publications describe our work on the development of Case-Based Mark-up Language:

- Lorcan Coyle, Conor Hayes and Pádraig Cunningham: Representing Cases for CBR in XML, in Expert Update Vol. 6, No. 2. pp 7–13, 2003.
- Lorcan Coyle, Dónal Doyle and Pádraig Cunningham: Representing Similarity for CBR in XML, in P. A. G. Calero and P. Funk (eds.), Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 Madrid, Spain, 2004, Proceedings, pp 119–127, Springer.
- Dónal Doyle, John Loughrey, Conor Nugent, Lorcan Coyle and Pádraig Cunningham: FIONN: A Framework for Developing CBR Systems. To Appear in Expert Update, 2004.

These final two papers describe a CBR solution to the problem of spam filtering. The case retrieval engine described in this work was also used in this application.

- Sarah Jane Delany, Pádraig Cunningham and Lorcan Coyle: An Assessment of Case-Based Reasoning for Spam Filtering, in L. McGinty and B. Crean (eds.) The Fifteenth Irish Conference on Artificial Intelligence and Cognitive Science (AICS'2004), Proceedings, pp 9–18.
- Sarah Jane Delany, Pádraig Cunningham, Alexey Tsymbal and Lorcan Coyle: A Case-Based Technique for Tracking Concept Drift in Spam Filtering, To Appear in Proceedings of the Twenty-Fourth Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (AI-2004).

1.5 Summary and Structure of this Thesis

In brief: Chapter 2 describes the operation of the PTA. Chapter 3 describes some case-based reasoning techniques that can be applied to the recommendation process. Chapter 4 describes our case representation format. Chapter 5 describes our recommendation strategies and Chapter 6 describes some of the interesting implementation aspects of this work. Chapter 7 contain evaluations of our recommendation techniques and the thesis is concluded with Chapter 8.

In more detail: Chapter 2 describes the operation of the Personal Travel Assistant in terms of an example interaction with the system. We follow this example throughout the thesis. This interaction involves a user making a travel request, receiving a set of flights and selecting one of these flights for purchase. This chapter discusses the need for a recommender system in the context of such an interaction.

Chapter 3 gives a description of the case-based reasoning methodology. It discusses some related research that will be applied in our recommendation strategies, including work in the areas of feature weight learning and cooperative CBR. It also describes four successful case-based e-commerce recommendation systems that have been deployed in recent years.

Chapter 4 contains an in-depth description of CBML. CBML is used throughout the PTA application and CBML snippets will feature in later chapters.

Chapter 5 describes the user-modelling techniques and recommendation strategies that were applied in the PTA. These strategies include our contextual recommender, our feature weight learner and our cooperative recommendation strategies. It uses example scenarios to describe these strategies.

Chapter 6 describes some novel or difficult aspects of the implementation of the PTA. It describes the PTA's user-interface, how it interacts with real-world flights providers and some of the implementation details related to CBML and our CBR implementation.

Chapter 7 discusses the advantages and shortcomings of our recommendation strategies. It contains detailed results comparing the strategies including statistical significances and discusses the implications of these results.

Finally Chapter 8 concludes the thesis and describes some further work that could be investigated.

Chapter 2

The Personal Travel Assistant

The original description of a Personal Travel Assistant was drawn up by the Foundation for Intelligent Physical Agents (FIPA) in 1998. This chapter describes their specifications and an earlier implementation of a Personal Travel Assistant. It describes the evolution of our system from these initial specifications and describes our final implementation of a fully functioning Personal Travel Assistant. We describe the operation of the PTA with an example user interaction. Finally we outline the necessity for intelligent personalisation techniques to reduce the cognitive load on the user.

2.1 Prior Art

2.1.1 FIPA's Personal Travel Market

The Foundation for Intelligent Physical Agents (FIPA) was formed in 1996 to produce software standards for heterogeneous and interacting agents and agent-based systems. FIPA described the PTA application as one of a set of benchmark scenarios to drive research on intelligent agents (FIPA, 1997b). Part of this proposal was the description of a Personal Travel Market. At one end of this market are a number of potential customers and at the other are a number of vendors selling travel services. Between them are a number of intelligent software agents who attempt to broker deals between the two:

Travel Service Agents These are agents operating on behalf of travel vendors, e.g. airlines, hotels, etc. They provide access to travel products. They are capable of responding to simple requests regarding the products they offer. They should also be able to complete the sales transaction if that is necessary. Simply put, they are responsible for delivering travel artifacts to the market.

Travel Broker Agents These are the middle-men of the personal travel market, they are responsible for taking high level requests from a user (e.g. "I want to take a holiday in Rome") and contracting with Travel Service Agents to compose a set of travel products, e.g. flights, hotels, car hire, etc.

Personal Travel Assistant The level of interaction required for a human user to interact successfully with a broker agent was seen as being rather tedious. The PTA was specified to make this interaction easier for the user. A PTA behaves as a customer's proxy in the market. It takes a travel request and negotiates with Travel Broker Agents to satisfy the user's requests. FIPA saw the PTA as being responsible for remembering and following the user's instructions and learning their preferences based on choices or feedback after the trip.

mPTA This is a lightweight agent capable of running on a mobile terminal (e.g. a WAP enabled mobile phone or PDA). Its function is to allow the user interact with the PTA while travelling. As the capabilities of these devices increase FIPA believed that more and more of the PTA's responsibilities could migrate to the mPTA.

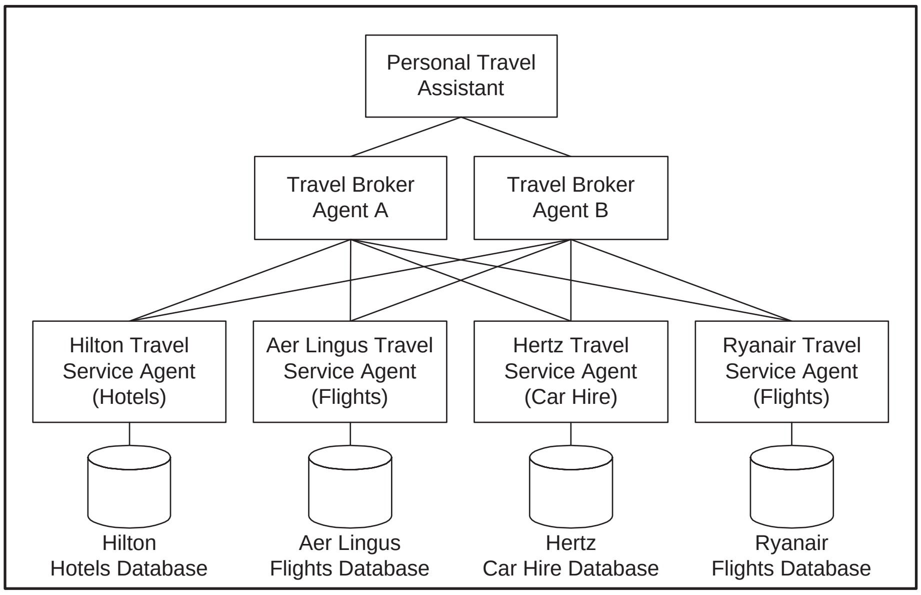

Personal Travel Assistant
Travel Broker Agent A
Travel Broker Agent B
Hilton Travel Service Agent (Hotels)
Aer Lingus Travel Service Agent (Flights)
Hertz Travel Service Agent (Car Hire)
Ryanair Travel Service Agent (Flights)
Hilton Hotels Database
Aer Lingus Flights Database
Hertz Car Hire Database
Ryanair Flights Database

Figure 2.1: FIPA's Personal Travel Market

Figure 2.1 shows a diagram of a FIPA-compliant Personal Travel Market architecture. The user accesses the Personal Travel Market using either the mPTA or the Personal Travel Assistant. The PTA sends requests to the Travel Broker Agents. These have contracts with various Travel Service Agents and use them to access travel products. The Travel Service Agents provide the travel products to the market and are effectively an interface into the Personal Travel Market for the travel vendors. Information flows vertically downwards through this chart in the form of requests and upwards in the form of offers. FIPA also specified a communicative language called

ACL (Agent Communication Language) with a *dialect* specifically defined for the travel domain (FIPA, 1997a), which these agents would use to communicate with each other. It was envisioned that the components of the Personal Travel Market would be created by individual vendors and that over time a real travel marketplace would emerge operating over FIPA standards.

2.1.2 The FACTS Personal Travel Market

The FACTS (FIPA Agent Communication Technologies and Services) project began working on an implementation of a FIPA-compliant PTM in 1998. Its primary objective was to validate the work of FIPA by constructing a number of demonstrator systems based on their proposed standards. They produced a number of papers describing their experiences in the use of FIPA agent technologies (O'Sullivan et al., 2000; Núñez-Suárez et al., 2000a; Núñez-Suárez et al., 2000b; Núñez-Suárez et al., 2000c). One of the participants of the FACTS project — Broadcom Eireann — designed and implemented a simple Personal Travel Agent (PTA). Their PTA used a combination of user specified travel preferences and reasoning based on previous interactions to allow the PTA to negotiate with Travel Broker Agents. Broadcom performed collaboration with researchers in Trinity College in Dublin to apply lazy learning techniques to the user-profiling task within the PTA (Waszkiewicz et al., 1999). However, their work was mainly proof of concept and the FACTS project wrapped up in 2000.

2.2 Our Personal Travel Market

Our original intention was the implementation of a FIPA-compliant Personal Travel Assistant and mPTA that could interact fully with a FIPA-compliant Personal Travel Market. Despite the early work on a Personal Travel Market, it was clear that publicly-accessible FIPA-compliant PTM was not going to be completed any time soon. Therefore we decided to continue with the development of a PTA application and implement a more modest version of the PTM ourselves. With this in mind we abandoned FIPA-compliant agent design and communication protocols.

The Travel Service Agents

The first deviation from the FIPA scenario was the decision to reduce the scope of the Travel Market to provide flights only. Since most of our target users are based in Dublin, we decided to focus on flights providers that fly out of Dublin. We implemented travel service agents for the two airlines that offer the most flights out of Dublin Airport; Aer Lingus and Ryanair. Between them, these airlines fly to more than 100 European cities.

Since airlines rarely allow direct access to their flights databases, we chose the option of using web-scraping techniques to get real-time flight data directly from their web sites. Web-scraping allows the travel service agents to interact with the flights web-site without the use of a browser, using scripts that automate the process. Our travel service agents are limited in that they can only deal with single-hop trips. This is because the underlying airlines only offer a point-to-point service, i.e. they only provide single-hop flights. These scripts allow the travel service agent to take a travel request and return a set of real flights. In addition to these flights, they also provide the information required for the PTA to forward a user onto the appropriate booking page should they wish to purchase a flight. The implementation details of these service agents and web-scraping techniques are described in more detail in Section 6.2.

The Broker Agent

We mentioned in our description of the personal travel market that the broker agent acts as the middle-man of the travel market. We implemented a broker agent capable of decomposing a PTA request into single-hop requests that can be processed by the travel service agents. It then recomposes the returned offers into a coherent form for the PTA. This decomposition/re-composition occurs whenever a user makes a request for a trip that requires more than a single-hop. The broker decomposes this into the individual hops, sends separate requests to the travel service agents and recomposes the single-hop offer-sets into a single multi-hop offer-set.

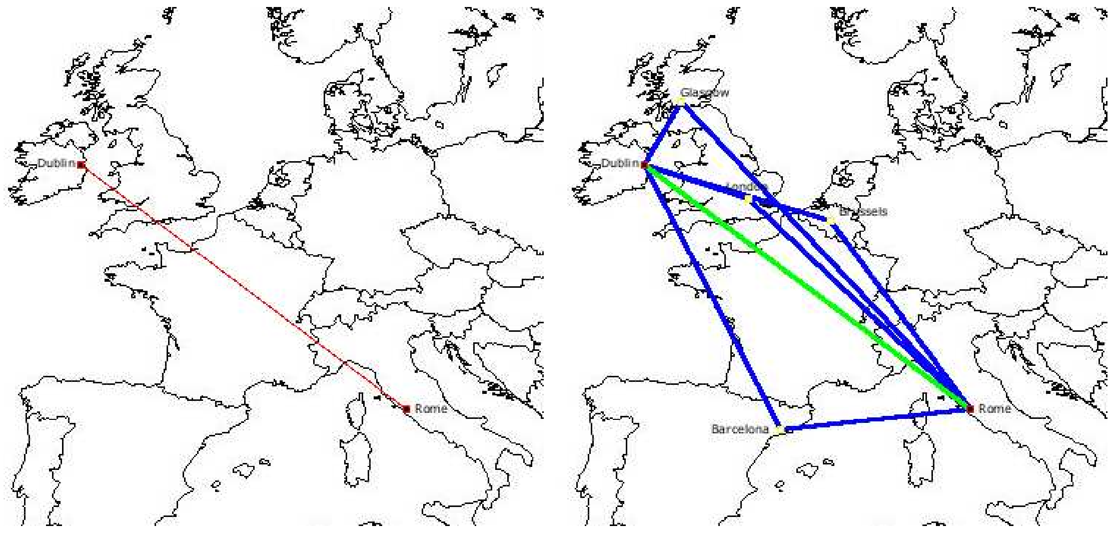


Figure 2.2: The Breakdown of a Request into Multiple Hops

In order to provide this service, the broker agent has knowledge about every origin-destination pair that is offered by the two service providers (it polls them for this information periodically). It uses these pairs to generate a *map* of the possible paths across Europe. When the broker receives a request from our PTA it uses this map to compose a list of the best (i.e. shortest) paths between the origin and destination. Each of these paths is separated into its constituent hops. These hops form the basis for a set of requests to the relevant service providers. Figure 2.2 illustrates that process in relation to a request for flights from Dublin to Rome. The map on the left shows the direct line between Dublin and Rome. The map on the right shows the possible paths between the two cities serviced by Aer Lingus and Ryanair. There are five possible paths in total; four two-hop Ryanair paths (Blue), one single-hop Aer Lingus (Green) path and no mixed paths (i.e. paths using hops from each airline).

When the service agents have responded to all the requests (or timed-out) the sets of hops are composed into full offers that satisfy the user's original query. This re-composition process involves the satisfying of simple constraints, e.g. the second hop must depart after the first lands (with enough time to check-in for the second flight), and yields an offer-set of departure flights and an offer-set of returning flights. Each of these offers will completely satisfy the user's original request.

2.3 The Personal Travel Assistant

We have implemented a single Personal Travel Assistant through which all users make requests for flights. We abandoned FIPA's idea of users accessing the PTA via a fat-client mobile-terminal based interface (the mPTA) and implemented a simple web-based interface. This decision was made out of frustration with the pace of development both in wireless terminal design, third generation network roll-out and migration of users to the wireless web. Having said this, our final web-interface is compatible with both fixed and mobile web users. In Section 6.1 we describe these problems and describe work done on developing a web interface for wireless users of the PTA.

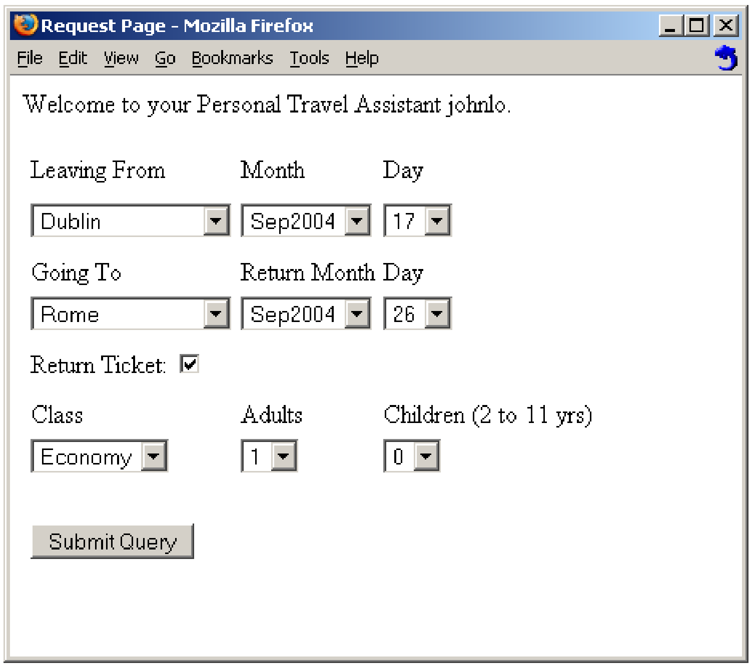
The user must register with the PTA before using it and logs into it before every session. This is to ensure that their interactions can be logged by the system. These data are used in the personalisation process, which is outlined in later sections. The following subsection describes the interactions between a user and the PTA by way of an example.

2.3.1 An Example Interaction with the PTA

User johnlo is looking for a return flight to Rome. He logs into the PTA and is presented with a request page. Figure 2.3 shows a screen shot of this page. The origin and destination fields contain a drop-down list of every city served by the travel service providers. This list is updated automatically by the PTA if a service provider goes offline, or adds or removes destinations. The travel and return dates allow the user to specify the day of departure and return.

This page contains a number of fields that must be completed before a request can be made. Table 2.1 shows the mandatory parameters that must be filled by the user on making a travel request to the PTA. These are the same parameters that are requested by any travel web page. johnlo makes a request for one economy class return flight ticket from Dublin to Rome leaving on the 17th of September and returning on the 26th of September.

When johnlo has completed this form he submits it to the PTA, which forwards it to the broker agent. The broker agent decomposes this into a number of sub-requests, forwards these to


Request Page - Mozilla Firefox
File Edit View Go Bookmarks Tools Help

Welcome to your Personal Travel Assistant johnlo.

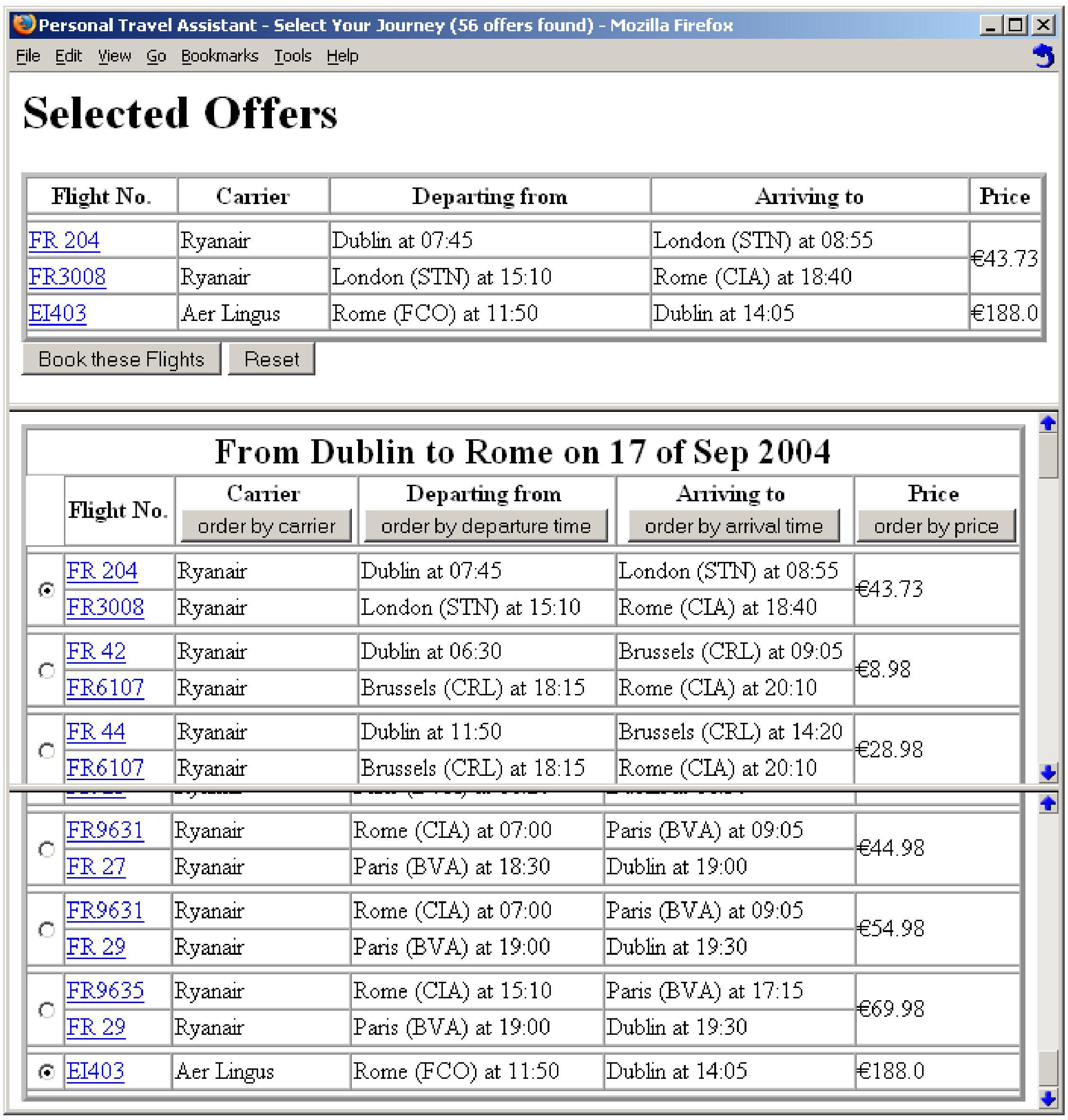
Leaving From: Dublin
Month: Sep2004
Day: 17
Going To: Rome
Return Month: Sep2004
Day: 26
Return Ticket: ✓
Class: Economy
Adults: 1
Children (2 to 11 yrs): 0

Figure 2.3: The Flight Request Page

Table 2.1: Mandatory Flight Request Parameters

Parameter	Description
Origin	The city of departure
Destination	The destination city
Flight Type	Whether the ticket is one-way or return
Departure Date	The departure date of the outgoing flight
Return Date	If this is a return ticket this is the date of the return flight otherwise it is empty
Number Of Passengers	The number of Adult and Child tickets the user requires.

the service agents and compiles two offer-sets (one outgoing and one returning) of valid flights for the user. The PTA takes these flights and presents them to the user in the next screen — the offers page. Figure 2.4 shows a screen shot of the offers page.


Personal Travel Assistant - Select Your Journey (56 offers found) - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Book these Flights
Reset
order by carrier
order by departure time
order by arrival time
order by price

Selected Offers

Flight No.	Carrier	Departing from	Arriving to	Price
FR 204	Ryanair	Dublin at 07:45	London (STN) at 08:55	€43.73
FR3008	Ryanair	London (STN) at 15:10	Rome (CIA) at 18:40	
EI403	Aer Lingus	Rome (FCO) at 11:50	Dublin at 14:05	€188.0

From Dublin to Rome on 17 of Sep 2004

	Flight No.	Carrier	Departing from	Arriving to	Price
●	FR 204	Ryanair	Dublin at 07:45	London (STN) at 08:55	€43.73
	FR3008	Ryanair	London (STN) at 15:10	Rome (CIA) at 18:40	
○	FR 42	Ryanair	Dublin at 06:30	Brussels (CRL) at 09:05	€8.98
	FR6107	Ryanair	Brussels (CRL) at 18:15	Rome (CIA) at 20:10	
○	FR 44	Ryanair	Dublin at 11:50	Brussels (CRL) at 14:20	€28.98
	FR6107	Ryanair	Brussels (CRL) at 18:15	Rome (CIA) at 20:10	

	Flight No.	Carrier	Departing from	Arriving to	Price
○	FR9631	Ryanair	Rome (CIA) at 07:00	Paris (BVA) at 09:05	€44.98
	FR 27	Ryanair	Paris (BVA) at 18:30	Dublin at 19:00	
○	FR9631	Ryanair	Rome (CIA) at 07:00	Paris (BVA) at 09:05	€54.98
	FR 29	Ryanair	Paris (BVA) at 19:00	Dublin at 19:30	
○	FR9635	Ryanair	Rome (CIA) at 15:10	Paris (BVA) at 17:15	€69.98
	FR 29	Ryanair	Paris (BVA) at 19:00	Dublin at 19:30	
●	EI403	Aer Lingus	Rome (FCO) at 11:50	Dublin at 14:05	€188.0

Figure 2.4: The Offers Page

The offers page is divided into three frames — the top frame shows the users selections of preferred flights and contains the submission button that will signal to the PTA that they wish to book those flights. The second and third frames contain the outgoing and returning flights respectively. The flights in these frames may be ordered by carrier, price, departure time and arrival time. johnlo has selected a two-hop flight to Ciampino airport via Charleroi airport (outside Brussels) and a single-hop return flight directly from DaVinci airport to Dublin (Ciampino and

DaVinci are two airports in the Rome area; their airport codes are `CIA` and `FCO` respectively).

Providing that he is happy with these flights, johnlo can submit the form by clicking the 'Book these Flights' button (otherwise he can go back and make a different request). This prompts the PTA to forward the user onto the booking web sites of the appropriate vendor. The PTA's part in the transaction is concluded at this point and the user is forwarded onto the booking page of the vendor's own web site to continue the transaction directly. This is advantageous because it avoids the security, privacy and legal issues that would have been inevitable if the PTA was designed to go through the booking process on behalf of the user. Appendix A describes another example user interaction with the PTA, which includes screen shots of the logon and hand-off screens.

We mentioned that the PTA logs information about the user's interactions. At the successful conclusion of a user-interaction, i.e. a user-interaction that ends with a user purchasing a flight (or pair of flights) the PTA stores a *session* object describing the interaction. The most important elements of this session are:

Request The request that the user made for flights

Offer-sets The sets of flights that were presented to the user, i.e. an outgoing offer-set and an incoming offer-set

Offer selections The actual selection of preferred flights by the user

In this way the PTA maintains a history of each user's interactions with it.

2.4 The Information Overload Problem

The offer-set of flights received by johnlo in the example interaction was quite large. In fact there were 56 flights in total (30 outgoing and 26 return flights). Obviously this is far too many to be compared easily, and if we look at Figure 2.4 we see that it is only possible to see four incoming and outgoing flights on the screen at one time. Not only is the size of the offer-set large, but it is also quite diverse. In fact, the outgoing offer-set has the following characteristics:

- There are two carriers — Aer Lingus and Ryanair
- There are two destination airports available in Rome — Ciampino and DaVinci
- The cheapest flight is €8.98 and the most expensive is €148.00
- Flights depart Dublin as early as 06:30 and arrive in Rome as late as 23:45
- There is one single-hop flight and 29 two-hop flights available. Among the two-hop flights there are the following additional choices:
  - There is a choice of five possible stopover airports (in Barcelona, Brussels, Glasgow, London and Paris).

    – Stopover times in these airports range between 90 minutes and 12 hours 40 minutes.

The size of this set is not atypical of the scenarios encountered by users of the PTA. On analysis, our users are presented with offer-sets with an average size of 30 flights and many requests yield much larger sets, e.g. a Dublin to London offer-set was observed containing 83 flights. With the size and diversity of these offer-sets it is inadequate to expect the user to manually search an offer-set for a preferred flight. Strategies are needed to reduce this information overload and make it easier for the user to find the most suitable flights. The solution to information overload is to improve the quantity and quality of information presented to the user. Two approaches to achieve this are *information filtering* and *information visualisation* (Shneiderman, 1986; Ware, 2000):

Information Filtering The PTA could present a reduced set of flights

Information Visualisation The PTA could reduce the volume of information (rather than the volume of flights) presented to the user or to cluster the flights according to suitable criteria

The PTA prototype does not use information filtering at all and it uses a very simple form of information visualisation by arranging the flights in a table and allowing the user to order them by feature values (e.g. by price). More intelligent presentation techniques than these would be necessary in a mature implementation of the PTA application. These presentation techniques will be more successful if the PTA proves capable of learning and applying each user's flight preferences to the process of recommending and presenting flights.

2.5 Conclusion

In this chapter we described FIPA's Personal Travel Market Scenario. This Scenario proposed an intelligent software agent called the Personal Travel Assistant whose job it was to guide and support the user in finding and booking travel items. We have implemented a Personal Travel Assistant that broadly follows their original specification. We used an example scenario to describe how this agent takes a user request for flights and contracts with software agents to find suitable flights. We showed how the offer-sets of flights that satisfy a request may be very large and how it is unsatisfactory for a user to have to trawl through these flights to find their preferred offer. We argue that a recommendation process is necessary to reduce the burden on the user. We propose to use Machine Learning techniques to accomplish this goal; specifically Case-Based Reasoning. Chapter 3 describes the Case-based Reasoning methodology and its application to recommendation systems.

Chapter 3

Case-Based Reasoning

Case-based reasoning (CBR) is an artificial intelligence methodology for solving problems by using or adapting solutions to old problems (Riesbeck and Schank, 1989). Previous experiences are retained as cases, which are usually described as problem-solution pairs. The problem represents the description of a previous situation, and the solution represents the action taken (or observed outcome) on that occasion. The collection of cases in the system is called the case-base, and is the main source of knowledge in the system.

The main advantages of CBR are its flexibility, extensibility, ability to handle incomplete data and ability to learn incrementally. CBR belongs to the lazy school of Machine Learning (Aha, 1997). Lazy learning algorithms defer inference computation until they receive a request for the solution to a problem. Whatever the run-time penalties associated with this, it does ensure that the data used for inference are as up-to-date as possible. This is of particular benefit in situations where data are scarce and where data are being updated continuously. The lazy learning approach of CBR also has the advantage that it can model local phenomena well compared to eager techniques that tend to focus on more global models.

Table 3.1: The Indexed Features of a Flight Offer Case

Feature	Description
Origin	The airport of departure
Destination	The destination airport
Departure Time	The time of day that the flight departs at
Time Of Flight	The total time between departure and arrival
Price	The price of the flight
Number Of Hops	The number of flight segments between the origin and destination
Stopover Location	The stopover airport (if the case represents a multi-segment flight)
Stopover Time	The amount of time to be spent in the stopover airport

CBR is commonly used in recommender systems, e.g. Ricci et al.'s *DieToRecs* travel recommender (2003), Smyth and Cotter's *PTV* recommender (1999), Shimazu's *ExpertClerk* system

(2002) and Burke's *Wasabi* personal shopper (1999). In case-based recommender systems the obvious cases are descriptions of the commodities on sale. These cases might describe package holidays, hardware configurations or as we will describe in this work, flight offers and requests for flights. Table 3.1 shows the indexed features in a flight offer case used in the PTA application.

3.1 The CBR Methodology

The CBR methodology is often described in terms of the R4 cycle described by Aamodt and Plaza (1994). Figure 3.1 shows a diagram of this system. This cycle contains four phases:

Retrieval The most similar case(s) to the problem description is retrieved from the case-base.

Reuse The solution(s) from the retrieved case(s) is used to suggest a solution

Revision If the suggested solution was unsuitable for use, it is repaired as best as possible.

Retention Once the new solution is confirmed, the problem-solution is added to the case-base as a new case.

We will describe each of these phases in the context of a recommender system in more detail in the remainder of this section.

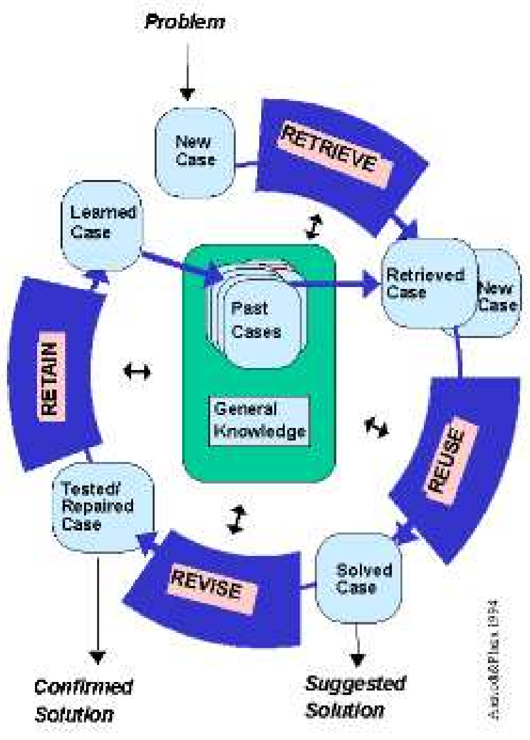


Figure 3.1: Aamodt and Plaza's 4 Stages of CBR (1994)

3.1.1 CBR Retrieval

The retrieval stage is generally held to be the most important phase of CBR as the retrieved cases will be used in the subsequent phases. Before retrieval proceeds, the case-base is organised into a structure that supports retrieval. This is called a *case-memory.* The query or *target case* is then presented to the case-memory to trigger reminding or retrieval. In recommender systems it is common to treat a query for a product as a target case and retrieve the most similar items in the case-base to be presented to the user.

The simplest and most common retrieval algorithm is the k Nearest Neighbours (k-*NN*) retrieval algorithm. This is a flat search algorithm — the case-memory is effectively unstructured. k-NNs have the advantage that they do not require any training period. However, they are slow because the case-base must be completely searched every time retrieval is executed. k-NN retrieval proceeds by sequentially comparing the query case to each case in the case base, and returning the k most similar cases. The major drawback with k-NN is its scalability. Retrieval time increases linearly with the size of the case-base. Thus k-NN is ill suited to domains where query response time is important.

There has been much work on the improvement of the k-NN algorithm. Lenz et al. (1998) introduced the Case-Retrieval Net (CRN), an efficient memory model that uses a bottom-up approach to calculate similarity. CRNs take advantage of redundancy in feature values and are tolerant of missing or unknown feature values. In case-bases where such circumstances exist they allow significant improvements in retrieval time. In Section 6.4.1 we give a fuller description of CRNs and in Section 6.4.2 we describe our implementation of a CRN-inspired k-NN retrieval algorithm.

3.1.2 CBR Reuse

The output of the retrieval phase of CBR is a proposed solution (or group of solutions in the case of k-NN retrieval). This solution is applied to the current problem, i.e. reused. In order to reuse a previous solution the CBR system may need to adapt it to take account of differences between the problem specifications. Several techniques are used in CBR for adaptation. In classification areas it is common to reuse the solution of the proposed case without adaptation (this is known as *null adaptation*). At the other end of the scale are more knowledge-intensive adaptation processes, e.g. the problem of tablet formulation (Craw et al., 1998). Watson (1997) suggests that adaptation is the Achilles' heel of CBR. He argues that complex adaptation is knowledge intensive, and since CBR is often applied to problems that are not well understood, complex adaptation will not be possible.

3.1.3 CBR Revision

Because the solution proposed in the reuse phase of CBR is not always correct, there is need for validation of the solution. This validation occurs in the CBR revision phase. Revision is often triggered if the suggested solution fails to solve the problem at hand. In effect the CBR system is corrected to account for this failure. In case-based recommender systems, the revision phase is often triggered by user-feedback.

3.1.4 CBR Retention

The CBR Retention phase involves applying the lessons learned from the solution of a new problem. In general this means the addition of a new case to the case-base. However, retention can also involve adjusting the CBR system in other ways, e.g. by altering the similarity measure or adaptation rules. In many CBR systems the retention phase is only triggered after proposed solutions fail.

3.2 The Knowledge Contained in CBR Systems

Richter has identified four different ways in which knowledge can be represented in a CBR system (1995; 1998). He has named these *knowledge containers* and they have met wide acceptance as a natural organisation of knowledge in CBR. Richter's knowledge containers are:

The Case-base This is the case data

The Vocabulary Used This is the description of the structure and semantics of the case data — the domain model

The Similarity Measure This describes how similarity between cases is calculated (used in the retrieval phase). We describe similarity measures in more detail in Section 3.3.

The Solution Transformation This describes how solutions to previous cases are adapted to solve new problems (used in the reuse phase).

The knowledge for the vocabulary, similarity measure and solution transform is structured and used at compile time and the knowledge in the case-base plays its role only at run time. Richter suggests that this is the major advantage of CBR because the knowledge acquisition of cases is easy. Although the acquisition of knowledge in the other containers is more difficult to obtain, shifting knowledge from the case-base to another container can lead to significant improvements in the system. In Section 3.4 we will describe work that has been done in shifting knowledge from the case-base to the similarity measure.

In Chapter 4 we describe an XML-based CBR representation language that allows the representation of three of these knowledge containers — the case-base, the vocabulary and the similarity measure.

3.3 Similarity Measures

The retrieval phase of the CBR cycle is dependant on the definition of a *similarity measure*. A similarity measure is a function to calculate the similarity between two cases. In cases made up of attribute-value pairs, the similarity is generally held to be related to the similarity of their shared features. Functions that calculate the similarity between two values of a particular feature are called local similarity measures. An amalgamation function describes how local similarity values are combined to generate an overall or global similarity between two cases. Amalgamation functions often use weights to define the relative importance of features in the global similarity calculation. Similarity between two cases Q and C is usually defined as the sum of the local similarity values multiplied by their relative importance:

$$Sim\,(Q, C) = \sum_{f \in F} w_f \times \sigma_f \tag{3.1}$$

where σ_f is the local similarity measure for feature f and w_f reflects its *weight* (or importance). F is the set of all features in Q.

Although the term similarity measure is often used in the literature to describe the function that calculates the similarity between two cases, we will use either the term *global similarity measure* or *similarity profile*. We do this to avoid confusion between global and local similarity measures.

In recommender systems pure similarity retrieval is not always the most suitable policy. McClave and Smyth (2001) suggest that recommendation based purely on similarity to a query case is inadequate in some domains. They propose that a set of proposed items should be both maximally similar to the query case and maximally different from each other. Bradley and Smyth (2001) describe a Personal Computer recommender system where PC specifications are represented as cases. The user makes a query for a PC and the system returns a set of configurations that are suitable (i.e. similar to the query case) while allowing the user a broad choice between items (i.e. diverse).

3.4 Learning Similarity

Although knowledge acquisition in the case-base container of a CBR system is performed lazily, it is possible to incorporate knowledge into the similarity measure in an eager manner. There has been much research on the improvement in the similarity measure in this way (Wettschereck et al., 1997).

Smyth and Keane (1993), Fox and Leake (1995), and Bergmann et al. (2001) propose that the traditional view that *'similar problems have similar solutions'* needs to be overcome. They discuss the notion of adaptability or *utility* and propose that the retrieval phase of CBR should retrieve the most adaptable or useful case for the current problem rather than the most similar. Figure 3.2 shows a diagram showing how CBR retrieves a similar solution (Retrieve) and uses it to generate a suitable solution (Reuse). By incorporating utility knowledge into the global similarity measure we should reduce the distance between $S1$ and S thus reducing the effort that would be needed in the subsequent reuse and revise CBR phases. This is useful because these phases are usually seen as being more costly and complex than the retrieval phase.

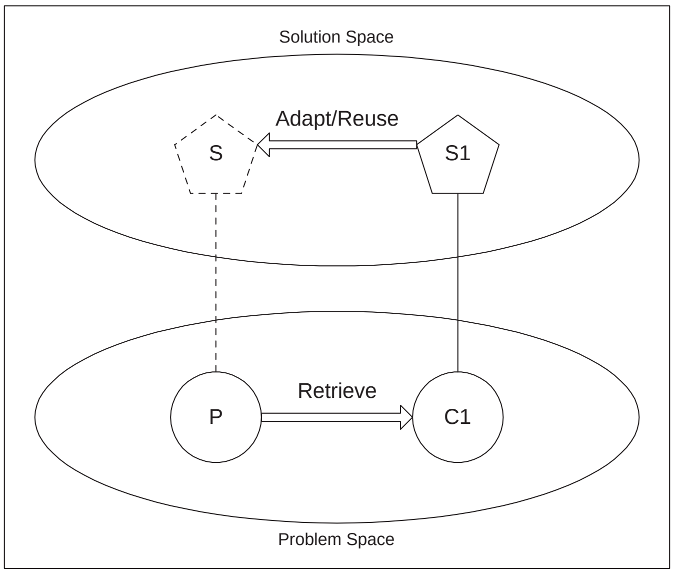


Figure 3.2: The CBR Transformation

Introspective Learning refers to an approach to learning problem solving knowledge by monitoring the run-time progress of a particular problem solver (Fox and Leake, 1995; Leake et al., 1995; Oehlmann et al., 1995). Many researchers use introspective learning techniques in CBR systems by examining and using recommendation failures to incorporate utility into the similarity measure. This leads to improvements in the overall performance of the system.

Case-based recommender systems are usually used to order a list of products or items (represented as cases). Stahl (2001; 2002) and Branting (2001) 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. (Stahl, 2001; Stahl, 2002; Branting, 2001), or by using evolutionary algorithms, e.g. (Jarmulak et al., 2000). In the remainder of this section we describe some other work that has been done in the area of introspective feature weight learning. These techniques are broadly similar; in each of these approaches there is an attempt to improve retrieval accuracy by incorporating utility into the global similarity measure.

3.4.1 Bonzano and Cunningham's Introspective Learning Techniques

Bonzano and Cunningham have developed an intelligent agent that provides decision support to human air traffic controllers in solving the problem of aircraft conflict resolution called *ISAC* (Bonzano, 1998). ISAC uses introspective learning techniques to determine the relative importance of parameters, i.e. to calculate optimal feature weight values. In doing this it improves its retrieval mechanism. In this way feature weights are updated in order to optimise problem solving performance. They use learning policies that *push* good cases towards the target case and *pull* bad cases from the target case by altering the feature weights. Their evaluations suggest that failure-driven policies are more effective for learning good feature weights. Their introspective approach uses a training data-set to drive learning which is then validated on a test data-set.

3.4.2 Stahl's Utility Learning Techniques

Stahl (2001; 2002) describes a framework for learning global similarity measures. The framework is centred around a 'similarity teacher' that has knowledge of the utility of solutions for a given problem. The similarity teacher uses its utility knowledge to correct the ordering of a set of cases retrieved by the CBR system in response to a query. A 'similarity learner' uses this corrected case order to improve the feature weights in the global similarity measure. Stahl suggests that such a framework would be suited to the e-commerce domain whereby the user could act as a limited similarity teacher (limited since the user would only provide a partially corrected case order).

Stahl and Gabel (2003; 2004) extend this framework with an approach for learning knowledge-intensive local similarity measures using evolutionary algorithms.

3.4.3 Branting's Learning Customer Weights

Branting (2001) 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 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 system's recommendation of an optimal item matches the user's selection. Branting calls this approach *Learning Customer Weights* (LCW).

3.5 Cooperative CBR

Some case-based systems distribute the case knowledge of the system among a number of reasoning agents, e.g. (Plaza et al., 1997; McGinty and Smyth, 2001). In such scenarios it may be productive to share case data between agents. This is especially useful when one agent's case knowledge is an imperfect description of the application domain and as such is inadequate to solve a problem. Cooperative (or collaborative) CBR techniques attempt to facilitate the efficient sharing of case data between agents.

Plaza et al. (1997) describe two modes of cooperation among homogeneous agents that solve problems using CBR: *distCBR* and *collCBR*. Their agents benefit from cooperation with agents that have better knowledge than themselves with relation to a given problem. distCBR approaches delegate authority to other peer agents to solve the problem rather than allowing the originating agent to solve the problem. collCBR approaches maintain the authority of the originating agent to solve the problem and use the knowledge of other peer agents to solve the problem.

The important considerations in using these approaches are in deciding when it is appropriate to seek cooperation among agents. It is essential to be able to accurately estimate the *competence* (Smyth and McKenna, 1998) or *justification* (Armengol et al., 2004) of both the originating agent and its peers to solve a problem a-priori. If the problem-space is assumed to be smooth, and the similarity measure to be reflective of case utility, then the assumption that similar problems yield similar solutions holds. This assumption could be used to gauge agent competence a-priori, whereby an agent who has experience of a similar problem to the current one is assumed to be competent to solve the current problem.

The cooperative CBR techniques developed by Plaza et al. (1997), Plaza and Ontañón (2001) and Ontañón and Plaza (2001) assume that each of the agents is attempting to solve a classification problem (marine sponge identification) where there is a single correct solution. Collaboration in recommender systems uses a different approach; the assumption is that agents represent user's tastes and a good solution for one agent might be a bad solution for another. For this reason there is a requirement that although an agent might be qualified to make an informed recommendation, their recommendation might not be suited to the originating user's preferences. The user preferences themselves should be incorporated into the collaboration function. Ideally, collaboration should occur with agents (users) who are both competent to offer a useful recommendation and who have similar tastes to the originating agent.

3.6 Example Case-based Recommendation Systems

We mentioned in the introduction to this chapter that CBR is a commonly used methodology in recommender systems. In this section we describe four case-based recommender systems and the strategies they use to make recommendations.

3.6.1 DieToRecs Travel Assistant

The DieToRecs travel recommender system (Ricci et al., 2002; Fesenmaier et al., 2003) allows a user to compile a *bag* of travel items for a vacation. This bag may contain a number of tourist products, e.g. locations, accommodations, attractions, activities and events. The system helps the user to search and add locations and activities to their travel bag. The user makes queries (which contain travel preferences) for travel products to add to their bag and DieToRecs returns a number of suitable travel products. If there are not enough matching products DieToRecs suggests ways to relax the queries. This is a conversational or mixed-initiative approach — the system and user share control over the reasoning process, at times taking the initiative to contribute to the interaction as required (Allen, 1999).

When enough products have been received, DieToRecs orders them using a two-stage CBR recommendation process (Ricci et al., 2003). Cases in DieToRecs are represented as sessions, with features describing the travel preferences, the travel queries, the products and ratings on the bag as well as the individual products themselves. DieToRecs uses the explicit travel requests of the user in the current session to find similar session-cases from other users. It then uses the ratings received on selected products from these similar sessions to order the current set of products. As the user develops her travel plan on-line, the system iteratively computes similarity to other user sessions, and makes suggestions on how to complete the itinerary. Once the user's trip is agreed, the session is stored as a case for possible use by another on-line traveller.

3.6.2 CASPER Job Recommender

The CASPER (Case-Based Profiling for Electronic Recruitment) research project has as its objective the building of intelligent aides for use in specialised retrieval applications (Bradley et al., 2000). The test bed application for this project is an on-line recruitment site[1]. Job are described as cases with a set of features such as job type, salary, skills and experience. Jobs can also be classified according to their relevance to the user-model. CASPER works by taking a query from a job seeker and ranking a set of jobs based on similarity to the query and on relevance to the user-model. In CASPER the user model contains a set of job-cases with rating scores indicating whether or not the user liked that job offer.

3.6.3 Turas Route Planner

Turas (McGinty and Smyth, 2000b) is a personalised route planner that allows users to plan and design routes through Dublin city. It is personalised for the needs of individual users, rather than more general routes, e.g. shortest path between two points. Cases represent previously graded routes and each user-model stores a case-base of route planning experience for a given user. Turas

[1]`http://www.jobfinder.ie`

reuses relevant sections from multiple cases to generate new routes (McGinty and Smyth, 2000a). It also uses collaborative approaches to generate routes in unfamiliar territories by reusing case knowledge from other users' case-bases that have experience in those areas (McGinty and Smyth, 2001). Users borrow cases in much the same way as Plaza et al.'s collCBR technique described in Section 3.5. Cases are shared between similar users, i.e. those with similar tastes. Similarity between users is calculated by examining their shared problems and solution overlaps, i.e. users are deemed similar if they solve the same problems in a similar way.

3.6.4 PTV Personal Television Guide

Smyth and Cotter's PTV system (1999) employs case-based user profiling and cooperative techniques to generate web-based TV viewing guides that are personalised for the viewing preferences of individual users. TV programs are stored as cases in a program case-base and user-models contain lists of programs with associated ratings. Personalised TV viewing guides are created using two techniques. Programs may be recommended to the user based on their similarity to programs the user liked and their dissimilarity to programs the user disliked in the past. Programs may also be recommended if similar users rated them highly. Similarity between users is calculated by comparing the common programs they rated. Similar users rated the same programs similarly, i.e. they had the same likes and dislikes.

3.7 Conclusions

This chapter provided a description of CBR and the types of case-based recommendation techniques that will be applied in the PTA application. In Section 3.2 we discussed the knowledge contained in a CBR system and showed how case knowledge can be transferred into the other knowledge containers. In Section 3.4 we described how utility can be incorporated into the similarity measure using introspective learning and described work by Bonzano and Cunningham, Stahl and Branting in this area. We propose to use a similar approach to learning personal similarity profiles to improve the accuracy of flight recommendations in the PTA application. This approach is described in detail in Section 5.3. In Section 3.5 we described some work in the area of cooperative CBR. We propose to use cooperative techniques to make recommendations to new users (who have not built up a user-model yet) and mature users whose user-models may not always be competent to make an informed recommendation. We describe our cooperative approach in Section 5.4.

Finally, in Section 3.6 we briefly described four different systems that use case-based recommending approaches. We take inspiration from some aspects of these systems. We use a similar two-stage recommendation technique to the DieToRecs system described in Section 3.6.1. We represent user-models as sets of cases much like the CASPER recommender system described in Section 3.6.2. We also use collaborative recommendation techniques much like the Turas and PTV systems described in Sections 3.6.3 and 3.6.4.

In Chapter 5 we will describe the application of the CBR techniques described in this chapter to the problem of flights recommendation and in Chapter 7 we will describe evaluations that verify their power in this domain. Next, in Chapter 4 we describe our work on CBR representation.

Chapter 4

CBR Representation

Kitano and Shimazu (1996) have proposed that CBR applications have been too narrowly focused on domain specific problems. They suggested that a CBR system should be viewed as a *medium* to be used in conjunction with the mainstream corporate information system. Hayes et al. (1999) describe imposing a standard case-based *view* on the information system of an application in order to retrieve case-knowledge. They anticipated that a standard way of representing CBR information will make this easier to achieve and proposed a case representation language that would facilitate this. Without such a standard means of representing case data it is up to the application developer to shape the case data from the available knowledge base. The manipulation of case data is dependent on the representation format chosen by the developer. This limits transformation of the data into a format suitable for the presentation layer, or its movement to another back-end component or between distributed CBR components. Hayes et al. proposed a standard case representation language called *Case-Based Mark-up Language* (CBML) in 1998. Our work in the field of CBR representation is a continuation of that work. We describe some prior art in the field of CBR-representation and give an in-depth description of the current implementation of CBML, its format and its capabilities. CBML is used in every component of the PTA system and CBML snippets will feature in later chapters.

4.1 Prior Art in CBR Representation

Developing a standard means of CBR representation is not a trivial task. Such a standard should be able to represent similarity measures, cases, their descriptions and adaptation knowledge. It should also be flexible enough to cater for complex case representations, such as hierarchical cases, as well as the commonly used ‘flat’ vector format.

4.1.1 CASUEL

One prominent Case Representation format was created by the INRECA group in the form of CASUEL (Manago et al., 1994). CASUEL was intended to serve as a standard for exchanging information between classification and diagnostic systems that use cases. It is used to represent all the information relative to a particular application domain in a common format. It is a flexible, object-oriented frame-like language for storing and exchanging descriptive models and case libraries in ASCII files. The descriptive model defines the basic terms used to describe the cases. It consists of a class hierarchy, a set of slots which represent the attributes of each class and a set of types which specify the range of possible values of the slots. CASUEL also supports a rule formalism for exchanging case completion rules and case adaptation rules, as well as a mechanism for defining similarity measures.

4.1.2 XML-Based CBR Representation

One of the disadvantages of CASUEL was that it used the Extended Backus Naur Form (EBNF) representation format (Wirth, 1977). The current standard for marking up structured, knowledge-rich data is XML, the eXtensible Mark-up Language. XML is a description language that supports meta-data descriptions for particular domains and these meta-descriptions allow applications to interpret data marked-up according to this format. A representation language based on XML has many advantages: interoperability, ease of reuse, as well as the application-independent exchange of data over existing network protocols. Most importantly it allows the developer access to the entire XML tool-set. This tool-set includes fast, reliable document parsers, e.g. SAX and DOM, validating documents e.g. DTDS and XML-Schema, and document transformers, e.g. XSLT.

However, as Wilson (2001) has pointed out, the benefits that accrue to XML in general will not be fully passed on to the CBR community until a standard means of representing case data in XML is developed. With such a standard, case data can be passed from one CBR system to another, irrespective of the complexity of the case definition.

Several CBR systems using XML-based CBR representation formats have appeared over the past few years. Some of the earliest work on representing CBR objects in XML was done in the area of distributed CBR. Shimazu's (1998) CARET system used XML to mark-up cases of natural language text describing technical support problems and solutions. Watson and Gardingen (1999) developed a HVAC sales support system that used an XML-based case representation language based on Shimazu's language. This facilitated the distribution of case knowledge from a central server to off-site salesmen. Both of these applications used standard HTTP protocols to distribute case data.

4.1.3 CBML 1998-2001

The earliest work in the CBR community on an XML-based case representation language was the introduction of CBML (CBMLv1) by Hayes and Cunningham in (1998). The main motivation behind the development of CBML was the ability to facilitate the storage and distribution of case data over a network, thus allowing the creation of distributed CBR applications. They intended for CBML to provide similar functionality to the CASUEL case representation language; therefore its design, as it evolved, was based in part on CASUEL.

Hayes and Cunningham described two documents; the *case content document* and the *case structure document*. The content document contained the contents of the case and the structure document contained the definition of the case as well its constituent features. It was possible to define symbol, integer and string feature types. The case structure document also contained a simple similarity measure representation containing feature weights and some simple local similarity measures.

However, a number of weaknesses were noted with CBMLv1, mostly related to its reliance on XML DTDs (Document Type Declarations) to validate its data. The DTD specification was concerned primarily with structure and did not allow for a definition of data typing in any form. CBMLv1 required the user to 'shoe-horn' data into a specific case format which was not easy to re-transform into other schema, a key advantage of XML representation. Furthermore, CBMLv1 could only cater for flat case structures.

4.2 CBML 2001-2004

Our work continued the development of CBML to the point where we are currently at CBMLv3[1] (Coyle et al., 2003; Coyle et al., 2004). The XML community recognised the limitations of the early DTD model and developed an alternative which allows for structural and type validation called XML-Schema. The maturity of XML-Schema led to the redevelopment of CBML and now the description of CBML is stored in an XML-Schema document — the **CBML Schema**. The CBML Schema is shown in Appendix C.1. Only documents that follow this schema exactly can be considered valid CBML.

CBMLv3 continued the tradition of separating structure from content. We developed the structural definition by adding new feature types, extending the abilities to restrict case data and allowing the creation of hierarchical cases. We separated the similarity measure description from the structure document and stored it in a new CBML document — the *similarity profile* document. The case content document was simplified and condensed. This was done to reduce the memory footprint of case content documents as well as to allow it to be parsed and transformed more easily from one form to another. New feature types were added to allow the representation of floating-

[1]The CBML web page is located at `http://www.cs.tcd.ie/research_groups/mlg/CBML/`

point numbers and features with tree-like or taxonomy structures. Feature structure inheritance was also implemented to reduce the size of structure documents. These changes led to a change in the syntax of CBML such that the original version is unrecognisable to the current version.

CBML is integral to the CBR components of the Personal Travel Assistant. All CBR objects in this application are stored in CBML format. The remainder of this chapter is dedicated to describing the capabilities and syntax of CBMLv3 (with examples from the PTA domain). Finally we discuss some advantages that we have observed from using CBML. The important CBML documents used in the PTA domain are included in Appendix C and this chapter will give the reader a more clear understanding of their function.

4.3 The Case Content Document

The syntax of a case is described in CBML as follows: feature values are encased by a pair of XML tags with the feature name. This makes them easy to read and easy to translate into other formats since their tags do not contain attributes. Figure 4.1 shows an example case from the PTA domain in CBML format. This case represents a flight that was selected by a user in an interaction with the PTA. The features of the case are the origin, destination, time of day, price, number of hops, stopover locations (`via`), total time, stopover time and whether or not the flight was purchased (`selected`). The values of each feature are enclosed within the feature tags, i.e. the value for feature `totaltime` is `615`.

```
<case name="offer4461">
        <origin>Dublin</origin>
        <destination>Ciampino</destination>
        <timeofday>615</timeofday>
        <price>62.53</price>
        <numberofhops>2</numberofhops>
        <via>London Stansted</via>
        <totaltime>615</totaltime>
        <stopovertime>335</stopovertime>
        <selected>true</selected>
</case>
```

Figure 4.1: An Example CBML Case

CBML cases are stored in a case content document. We have developed a Case content parser that loads these documents into Java class instances (this process is described in more detail in Section 6.3). However, without an underlying structural description it would be impossible for the parser to perform this transformation or for the CBR system to interpret the case. CBML stores this description in a case structure document.

4.4 The Case Structure Document

In order to read a case correctly the parser needs to be sure of the hierarchy and cardinality of its features, their types and possible restrictions of their values. The case structure document defines these. A case structure document is made up of feature structure definitions describing the features that can occur in a case. The feature structure defines the feature's type, its value restrictions, and other attributes. Table 4.1 describes the four attributes of a feature structure definition. If any of these attributes are omitted from a feature structure definition the parser assumes the default values.

Table 4.1: Description of a CBML Feature Structure

Attribute	**Definition**	**Default**
`name`	The name of the feature — used for identification	N/A
`discriminant`	If `true` this feature is used in the CBR Process	`true`
`mandatory`	If `true` this feature is required to appear in every case	`true`
`solution`	If `true` this feature is the solution component of the case. N.B. only one feature can be defined as a solution feature in a case.	`false`

Figure 4.2 shows an example feature structure definition. This feature is called `selected` and is used in the case description of flights in the PTA application. It describes whether or not a user purchased the flight that this case represents. `selected` is defined as the `solution` feature of the case. Because the definition does not have explicit values for the `discriminant` and `mandatory` attributes it takes the default values (`true` and `true` respectively). The enclosed tag (`<boolean/>`) defines the feature's type. There are seven possible feature types in CBML: `symbol`, `integer`, `double`, `boolean`, `string`, `taxonomy` and `complex`. We will describe these feature types later in this section.

```
<feature name="selected" solution="true">
        <boolean/>
</feature>
```

Figure 4.2: An Example Feature Structure Definition

The case parser requires the case structure document to understand and convert a case content document into a case or case-base. If it parses a case from a content document that does not follow the structure definition exactly it will fail. In effect, this means that the case structure document is used to *validate* the case content document. This validation serves to protect the CBR system from an unexpected failure due to bad case or feature content. The Case Structure document is itself validated against the CBML Schema document by the CBML document parser.

4.4.1 Simple and Complex Feature Types

There are two groupings of feature types: simple and complex. A simple feature is an attribute-value pair; each of the following structure types are simple — `symbol`, `integer`, `double`, `boolean`, `string` and `taxonomy`. A complex feature (`complex`) is simply a container for sub-features (both simple and complex). Use of the `complex` feature type allows us to create cases with hierarchical structures, i.e. to nest features within a case. An example of this nesting is shown in Figure 4.3. This figure shows a fragment of an offer-case that is used internally in the PTA application. `hops` contains only informational features that are not used in the CBR process (therefore `discriminant="false"`). `hop1`, `hop2` and `numberofhops` are the child features of `hops`. The internal feature structures for these child features are not shown. Child features retain the properties of their parent, i.e. as `hops` is non-discriminating, its child features are also non-discriminating.

```
<case>
        ...

        <feature name="hops" discriminant="false"/>
                <complex>
                        <feature name="numberofhops">
                                <integer>
                                        ...
                                </integer>
                        </feature>
                        <feature name="hop1"/>
                                <complex>
                                        ...
                                </complex>
                        </feature>
                        <feature name="hop2"/>
                                <complex>
                                        ...
                                </complex>
                        </feature>
                </complex>
        </feature>
        ...
<\case>
```

Figure 4.3: An Example Hierarchical Case Structure

4.4.2 Symbol and Taxonomy Structures

Symbolic features are commonly used in the CBR community and such features are defined in CBML as `symbol` types. `symbol` features can have one of an enumerated list of possible values. This structure definition of a `symbol` feature simply contains this list. All `symbol` features that correspond to this definition must contain one of the values from this list. An example `symbol` feature structure definition is shown in Figure 4.4. This feature structure is called `tickettype` and is used in the request-case structure definition (shown in Appendix C.2). It describes whether the flight request was for a one-way or return flight. It has two possible values: `single` and `return`.

```
<feature name="tickettype">
        <symbol>
                <enumeration value="single"/>
                <enumeration value="return"/>
        </symbol>
</feature>
```

Figure 4.4: An Example Symbolic Feature Structure Definition

The `taxonomy` feature type is based on the `symbol` type. The `taxonomy` feature type allows the definition of a tree of symbolic values. The structure definition of the `taxonomy` feature type defines a tree containing the values and structure of the taxonomy itself. This feature type is useful for tree-type similarity measures that measure the similarity based on the number of branches between two feature values (this type of similarity measure is described in more detail in Section 4.5.2). `taxonomy` features are validated in the same way as `symbol` types; they must contain one of the node names from their tree definition. An example `taxonomy` feature structure is shown in Figure 4.5. This shows part of the feature structure for the `origin` feature and is used in the offer-case structure definition (The full definition contains more than 100 entries and is shown in the offer-case feature structure in Appendix C.4). This structure defines the political geography of the available airports in the system in a tree-like structure. By this structure, the similarity between Ciampino and Dublin (five branches separate them) is defined as being less than the similarity between Ciampino and Da Vinci (two branches separate them).

```
<feature name="origin" >
        <taxonomy name="Airports">
                <node name="Ireland">
                        <node name="Dublin"/>
                </node>
                <node name="Italy">
                        <node name="Rome">
                                <node name="Ciampino"/>
                                <node name="Da Vinci"/>
                        </node>
                        <node name="Milan">
                                <node name="Malpensa"/>
                        </node>
                </node>
        </taxonomy>
</feature>
```

Figure 4.5: An Example Taxonomy Feature Structure Definition

4.4.3 Integer and Double Feature Structures

Another commonly used feature type is the numeric type. CBML describes two numeric feature types: the `integer` type, which describes whole-number valued features and the `double` type, which describes floating-point valued features. Numeric features may be constrained by setting an optional maximum and minimum possible value. Figure 4.6 shows an example `integer` feature structure. This shows the feature structure for the `timeofday` feature used in the offer-case structure. This constrains the values of the feature `timeofday` to whole numbers valued between `0` and `1339`, inclusive. The value of this feature corresponds to the number of minutes after midnight that the flight departed (1339 corresponds to 23:59).

```
<feature name="timeofday">
        <integer>
                <minInclusive value="0" />
                <maxInclusive value="1339" />
        </integer>
</feature>
```

Figure 4.6: An Example Integer Feature Structure Definition

4.4.4 String and Boolean Feature Structures

There are two further feature types in CBML; the `string` and `boolean` types. `string` features may be any String value and so are usually used for purely informational features (i.e. feature structures where `discriminant="false"`). `boolean` features are a trivial type of symbolic feature in that they may only contain the values `true` or `false`. Figure 4.2 showed an example `boolean` feature structure definition.

4.5 Similarity Measure Representation

We mentioned in Section 3.1.1 that retrieval in a CBR system is dependent on the definition of a similarity profile describing the similarity between two cases. CBML defines this measure in the similarity profile document (Coyle et al., 2004). Similarity profile documents are defined by the CBML Schema and are validated by the CBML parsers in the same way as case structure documents.

In Section 3.1.1 we also defined a similarity profile as the amalgamation of the local similarities of the common features between two cases:

$$Sim(Q, C) = \sum_{f \in F} w_f \times \sigma_f$$

where w_f is the feature relevance weight and σ_f is the local similarity measure. In order to provide a representation of the similarity measure it is therefore necessary to represent both a relevance weight and a description of the local similarity measure for every feature.

Feature similarities have attributes containing their name and the relevance weight attached to them in the amalgamation function. The relevance weight is simply an attribute called `weight` that can have any floating-point value. Within each feature-similarity description there is also the description of the local similarity measure. These fall into one of four categories: `exact`, `difference`, `array` or `complex`. These local similarity types are described in the following sections.

4.5.1 Exact Similarity Measures

The simplest local similarity measure is based on exact matching. Similarity is assigned the value `1` if two feature values are equal, otherwise it is assigned the value `0`. In this sense it is a boolean similarity measure. We represent this type of function as an `exact` type similarity measure. Figure 4.7 shows the representation of a local similarity measure for a feature called `Gender` that uses an `exact` similarity measure. `Gender` is a `symbol` feature used in the blood-alcohol domain (Cunningham et al., 2003a). `Gender` is defined as having a relevance weight of `0.25`.

```
<feature name="Gender" weight="0.25">
        <exact/>
</feature>
```

Figure 4.7: An Example Exact Similarity Measure Definition

4.5.2 Difference-Based Similarity Measures

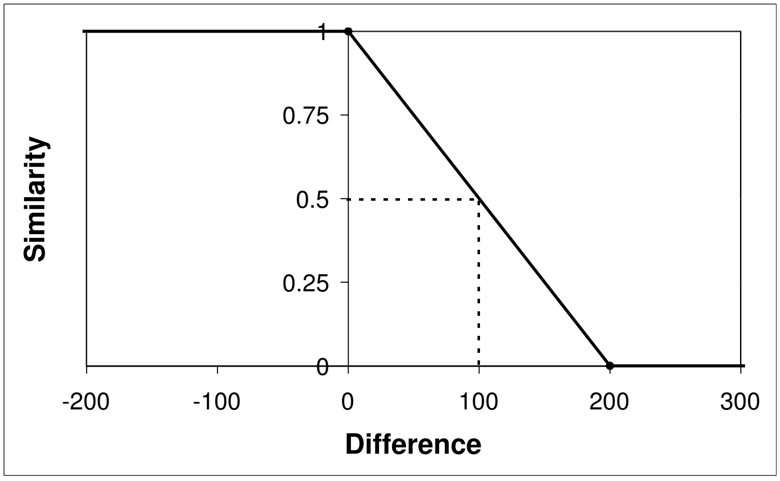
Local similarity measures may also be based on the difference between the feature values. These measures assume that the similarity between two feature values is related to the difference, δ between them. CBML represents these as `difference` measures. `difference` similarity measures are suitable where a difference can be easily defined between feature values; e.g. with numeric features the difference is defined as the mathematical difference of their values. Since `string` features were conceived to be used for purely informational purposes (and since there are a multitude of possible difference definitions), and difference functions for `boolean` types are trivial, we will confine ourselves to the definition of numeric, symbolic and taxonomic difference functions. These difference functions are defined in Table 4.2. With a definition of difference in place it remains to come up with an adequate definition of the relationship between difference and similarity.

The difference-based similarity definition also describes how the difference value (which is a number) should be transformed into a local similarity value. This transformation is described as a graph of difference against similarity. The points in this graph are defined in the similarity definition. By defining a suitable set of points, any piece-wise linear relationship between similarity and difference can be represented. This graph may be symmetrical (the default) or asymmetrical. A

Table 4.2: Definition of Difference Functions

Feature Type	Difference (δ) definition between Value1 and Value2
Numeric	`Value1 - Value2`
Symbolic	Relative difference in the positions of `Value1` and `Value2` in the list of possible values.
Taxonomy	Number of branches between `Value1` and `Value2`'s nodes in the taxonomy

symmetrical graph only deals with absolute difference values. Figure 4.8 shows the representation of an asymmetrical `difference` function similarity measure for the feature `price`. `price` is a numeric (`double`) feature that represents the price of a flight in the PTA application. `price` is defined as having a relevance weight of `0.2`. A graph of similarity versus difference is defined with a number of point definitions. For demonstrative purposes, this graph is also plotted. From the graph, the calculated similarity between two prices with a positive difference of €100 is `0.5` (whereas a difference of negative €100 would have yielded a similarity of `1`).



```
<feature name="price" weight="0.2">
        <graph type="asymmetrical">
                <point difference="-Infinity" similarity="1"/>
                <point difference="0" similarity="1"/>
                <point difference="200" similarity="0"/>
                <point difference="Infinity" similarity="0"/>
        </graph>
</feature>
```

Figure 4.8: An Example Difference-Based Similarity Measure Definition

The CBML similarity profile parser uses the feature type from the case structure to determine which type of difference function to use, i.e. numeric, symbolic or taxonomic. Similarity between two feature values is calculated by measuring the difference and using the difference graph to convert this value into a similarity value.

4.5.3 Array Similarity Measures

The `array` similarity measure is a means to define similarity exactly for each combination of feature value. It is useful for features with a finite number of possible values, but requires the user to calculate the similarities in advance. If a similarity value is not defined in this array, the `exact` similarity measure will be used. Figure 4.9 shows the CBML representation of an `array` similarity measure for the feature `meal`. `meal` is a symbol feature used in the blood-alcohol domain (Cunningham et al., 2003a). `meal` is defined as having a relevance weight of `0.75`. The similarity definition also defines an array of every possible feature value combination with a similarity value for each, e.g. $\sigma_{Meal}\left('none', 'snack'\right) = 0.8$. The array as defined is also shown in this figure.

```
<feature name="meal" weight="0.75">
        <array>
                <primary name="none">
                        <secondary name="snack" value="0.8"/>
                        <secondary name="lunch" value="0.4"/>
                </primary>
                <primary name="snack">
                        <secondary name="none" value="0.8"/>
                        <secondary name="lunch" value="0.8"/>
                        <secondary name="full" value="0.4"/>
                </primary>
                <primary name="lunch">
                        <secondary name="none" value="0.4"/>
                        <secondary name="lunch" value="0.8"/>
                        <secondary name="full" value="0.8"/>
                </primary>
                <primary name="full">
                        <secondary name="snack" value="0.4"/>
                        <secondary name="lunch" value="0.8"/>
                </primary>
        </array>
</feature>
```

	none	snack	lunch	full
none	1	0.8	0.4	0
snack	0.8	1	0.8	0.4
lunch	0.4	0.8	1	0.8
full	0	0.4	0.8	1

Figure 4.9: An Example Array Similarity Measure Definition

4.5.4 Complex Similarity Measures

It is impossible to provide for a representation scheme that could cover every possible type of local similarity measure, e.g. polynomial or exponential. If the number of possible values is finite, it may be appropriate to calculate all possibilities a priori and store them using the `array` similarity definition. If this is impossible or impractical it is possible to define a similarity measure externally from the similarity profile document, e.g. in a Java function or MathML document, and refer to it using the `complex` similarity definition. Figure 4.10 shows the representation of a `complex` similarity measure for the feature `sepal-length` that contains a reference to a predefined local similarity measure (`iris.similarity.SepalLength`). `sepal-length` is a numeric (`double`) feature used in Fisher's iris domain (Fisher, 1936). It defines `sepal-length` as having a relevance weight of `0.25`. It also tells the CBR system to use the predefined `iris.similarity.SepalLength` local similarity measure for this feature.

```
<feature name="sepal-length" weight="0.25">
        <measure name="iris.sim.SepalLength"/>
</feature>
```

Figure 4.10: An Example Complex Similarity Measure Definition

`iris.similarity.SepalLength` actually refers to a Java class that implements a CBML Similarity Measure Interface (shown in Figure 4.11). Any Java class that implements this interface can be referred to in the CBML similarity profile document. It will then be used by the CBR system to calculate the similarity between `feature1` and `feature2`. The interface itself contains a single method that takes in two features and returns a double — the similarity value. This ensures a level of interoperability. In this way any user-defined local similarity measure is capable of being used in a CBML compliant CBR system. However, since `complex` similarity measures depend on external resources that are outside the CBML core specification their use is discouraged.

```
package cbml.cbr;
public interface SimilarityMeasure extends java.io.Serializable {
  double calculateSimilarity(Feature feature1, Feature feature2);
}
```

Figure 4.11: The Local Similarity Measure Java Interface

4.6 Conclusion

We have suggested that a standard integrated CBR view of an e-commerce information system is an important facility. We proposed that using a standard XML-based meta-language, CBML, is a key to enabling CBR applications to integrate with existing information systems. This chapter described the form and syntax of this language with examples. In Chapter 5 we will describe the recommendation and personalisation processes involved in the PTA application. All CBR data in this application are represented in CBML format.

Our experiences with CBML in the Machine Learning Group in Trinity College have been positive. There are currently six researchers in this group using CBML for representing their CBR data in several applications, including explanation-based CBR (Cunningham et al., 2003a; Doyle et al., 2004a; Nugent and Cunningham, 2004), spam filtering (Delany et al., 2004a; Delany et al., 2004b) and feature selection (Loughrey and Cunningham, 2004). We observed a number of advantages from using CBML as our representation format:

Access to the XML Tool-set A primary advantage of using CBML was access to the XML tool-set; especially the wide number of publicly available, reliable and fast XML document parsers and validators. Access to these tools allowed us to concentrate on the CBR process rather than the development of parsing technologies.

Modularisation Using a standard representation format for our CBR data allowed us to keep structure and similarity definitions separate from application code. If the structure of a case (e.g. adjust the properties of a feature) or the definition of a similarity measure needed to change we simply had to alter the relevant CBML document. The structure of the cases in the PTA application changed several times over the course of this research (e.g. when the recommendation process was changed — which we describe in Section 5.1) and the modularisation enabled by CBML made the updates easier than they might otherwise have been.

Portability of Data across Heterogeneous CBR Systems The CBML specifications, parsers and other associated CBR components were formed into a CBR framework called *Fionn* (Doyle et al., 2004b). Fionn contains several heterogeneous CBR components. Because these components use CBML representations and parsers, we have been able to transfer case data between the components seamlessly. In Section 6.4 we give a description of the Fionn framework.

Personalisation and Collaboration In Chapter 5 we describe how we use CBR to generate user-models of the users of our PTA system. These models are made up of case-bases describing interactions with the system and similarity measures that drive the recommendation process. Because these components are stored in CBML format they can be easily distributed and shared between users.

Distributed CBR Since the original motivation of the creation of an XML-based CBR representation language was to facilitate distributed CBR applications, it is fitting that we should examine how the current version has lived up to these ideals. One of the motivations behind the PTA was the development of a mPTA which would allow personalisation to take place at the client side. CBML would certainly facilitate this process as long as the client CBR system was CBML compliant.

Ease of Translation Since CBML is well structured it is easy to translate it into different forms. In the PTA application case data are translated from CBML in the CBR components to HTML for the user interface, and to a database at the back-end of the application. It would be easy to develop translators to convert CBML data into any other well-structured language, e.g. FIPA's Agent Communication Language (FIPA, 1997a).

The current implementation of CBML can represent three of Richter's four knowledge containers (described in Section 3.2) — the case base, the similarity measure and the domain vocabulary. It is therefore clear that the next step in the development of CBML should be the creation of a representation for the fourth container — the solution transform.

Chapter 5

PTA Recommendation

An intelligent flights recommendation component is an important requirement of the Personal Travel Assistant application if it is to overcome the problem of information overload. In order to make good recommendations to a user, it is crucial that the PTA understands and replicates their travel preferences. We make the assumption that these preferences can be captured from observations of behaviour. We store these preferences in a model of user tastes — a user-model. This model is used to personalise the recommendation process for each individual user. It is developed and extended with every user interaction with the system.

Ideally, with enough data to generate a comprehensive user-model, it should be possible to infer a user's exact preferences and always recommend the correct flight. However, the generation of a perfect user-model is impossible due to a number of factors, most notably:

Unforeseen Circumstances User preferences are often dependant on factors that we cannot foresee or model. A good example of this is when users make purchases on behalf of another person, e.g. purchasing a gift. Clearly inferences made on these purchases are unlikely to be important in making future recommendations to that user (unless she is looking to purchase another gift for the same person).

Changing Tastes Changing preferences can occur gradually or be triggered by a specific event, e.g. price might be less of an important feature for a user that has just won the lottery. Gradual changes are easier to model — it is possible to place higher importance on the data contained in more recent sessions.

Irrational or Illogical Behaviour The user may not always behave consistently or with logic. They may make selections based on a whim. It is a well known psychological fact that there are random elements to user selections. If a person is presented with a similar set of items they will rarely select the same item consistently as their preferred one.

We mentioned in Section 2.3 that the PTA stores a session after the conclusion of each successful user interaction. The most important part of the session is the selection of preferred offers from the presented offer-sets. It is from these selections that the PTA makes its inferences on the user's travel preferences. These inferences in turn drive future recommendations. This is the only feedback the PTA receives from the user. Because the selection of preferred flights is a necessary part of the transaction we view this feedback as purely *implicit*. This is a critical point because by using implicit feedback the PTA minimises the cognitive load it imposes on the user.

We have already explained how the PTA receives offer-sets from the broker agent and how it presents them to the user. The recommendation process occurs between these two events whereby the PTA calculates a recommendation score for each offer in the offer-sets and orders the sets based on these scores. We implemented two distinct recommender systems that operate using different paradigms:

Offer-Based Recommender This was our initial recommender implementation. Recommendations are made on the basis that the user would prefer similar offers to those they purchased in previous interactions.

Session-Based Recommender This was our final recommender implementation. Recommendations are made on the basis that the user would prefer similar offers to ones they selected in *similar* previous interactions. This recommender uses knowledge from sessions with similar contexts to recommend flights. The context of a session is the set of circumstances that led to a set of offers being created.

This chapter describes the case-based recommendation strategies we applied in the PTA application. Many of these strategies were developed from techniques described in Chapter 3. We discuss some of the issues we came across and outline some novel solutions to the problems of recommendation in this domain. We also describe some additional components that were added to the session-based recommender that allowed further personalisation and facilitated collaborative recommendation. Later, in Chapter 7 we describe a set of evaluation results that compare the recommendation accuracy of these approaches and discuss their relative utility.

5.1 Offer-Based Recommendation

Our initial case-based recommender system made the assumption that the user would like similar offers to those they purchased in previous interactions. The PTA quantifies the user preferences on flights as rating scores. It then uses these rating scores to drive the recommendation process. This strategy is similar to the approach used in the PTV and CASPER systems (described in Sections 3.6.2 and 3.6.4 respectively).

In CBR terms it is appropriate to view these flights and their ratings as cases (we will call them *offer-cases*). The parameters of the flight represent the problem component of the case and the rating score represents the solution. Figure 5.1 shows an example offer-case in CBML format. This case represents an outgoing offer that the user selected in a previous session. It is a two-hop flight departing Dublin airport at 13:35 (815 minutes after midnight) for Ciampino airport in Rome, stopping-over for 110 minutes in Stansted Airport. The price of this offer was €75.29 and it takes a total of six-and-a-half hours (390 minutes) to complete the journey. This case was has a rating score of 1.0 (this is the maximum possible rating score and indicates that the user selected this offer). The CBML case structure document for offer-cases is shown in Appendix C.4.

```
<case name="offer5151">
        <outoffer>departure</outoffer>
        <numberofhops>2</numberofhops>
        <origin>Dublin</origin>
        <timeofday>815</timeofday>
        <destination>Ciampino</destination>
        <via>London Stansted</via>
        <stopovertime>110</stopovertime>
        <price>75.29</price>
        <totaltime>390</totaltime>
        <rating>1.0</rating>
</case>
```

Figure 5.1: An Example Offer-Case in CBML format

In summary the offer-based recommendation process is based on two phases. The first phase is an *offer-rating* phase. The PTA automatically calculates a rating score for each flight in the offer-set at the end of each session based on the user's purchases. These rating scores are stored in the offer-cases in the user-model. The second phase is the *offer-recommendation* phase. The PTA recommends new offers based on the rating scores that were calculated for similar previous offers. These phases are described in more detail in the remainder of this section. We also outline some limitations of this approach that lead to a development of an alternative recommendation process.

Phase 1: Rating Offers

The offer-rating phase of the recommendation process occurs at the end of a session, when the user selects a preferred offer from the presented offer-set. The PTA sets the `rating` feature of each of these offers according to the user's observed flight preferences. However, one of the requirements for the PTA recommender system is that it should not impose a high cognitive load on the user. Clearly, it would be inappropriate for the PTA to require an explicit rating score for each offer (the offer-sets shown in Figure 2.4 contained 56 offers). Therefore, the PTA makes inferences from the user's selection of a preferred offer, $O_{selected}$ from the offer-set to drive the rating process. Obviously $O_{selected}$ is the user's preferred offer, so the PTA uses it as a reference by which it rates all other offers. The assumption is made that the more similar an offer is to the selected offer the higher a rating score it should get. Figure 5.2 shows a diagram of the offer rating phase.

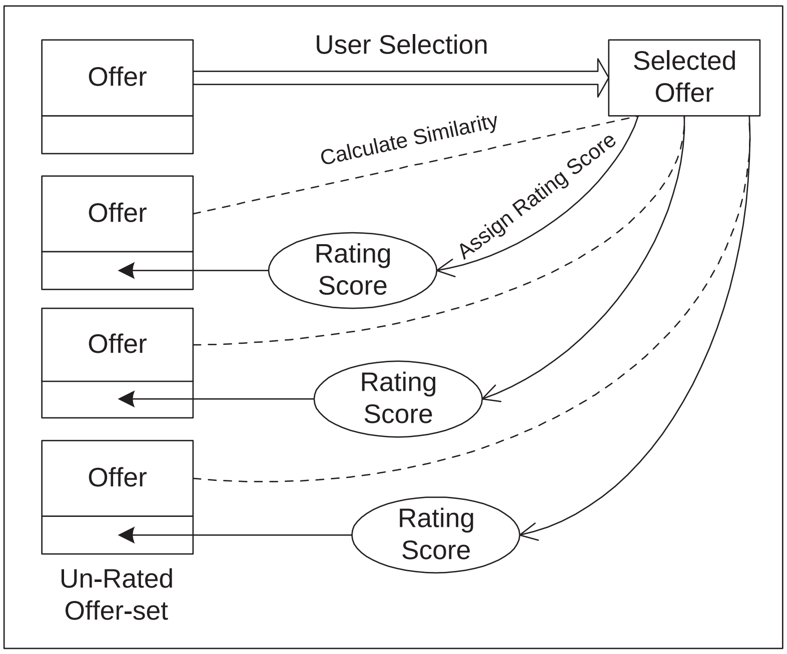


Figure 5.2: Offer-Based Recommendation: Rating Offers

The similarity between each of the offers in the offer set is used as that offer's rating score, $Score_i$, i.e.

$$Score_i = Sim\left(O_{selected}, O_i\right)$$

This means that the $O_{selected}$ is granted a score of 1.0 and all other offers get scores lower than this related to their similarity to $O_{selected}$. The similarity between two cases is calculated in terms of the local similarities of nine features — four `integer`, three `taxonomy`, one `symbol` and one `double` feature (The CBML representation of this similarity profile is shown in Appendix C.5). Features are weighted equally.

Consider an example scenario, where a user is making a request for a flight to London and has selected a flight for purchase. Table 5.1 illustrates how global similarity is calculated and applied to the rating process. Offer O_{4426} (in the leftmost column) was selected by the user and so it gets a rating score of `1` (the maximum). The other offers in the offer-set are rated according to their similarity to O_{4426}. The second half of each column shows the local similarity values (σ) between the features of that offer and O_{4426}, e.g. $\sigma_{price}\left(31, 149\right) = 0.46$. Since features are weighted equally the global similarity is the sum of the local similarities, i.e. $Sim\left(O_{4426}, O_{4458}\right) = 8.21$. The maximum possible similarity is `9` so O_{4458} is given a rating score of `0.91` ($8.21/9 = 0.91$).

Each of the offers in the offer-set are rated in this way and added to the user's offer case-base. This case-base is used to drive the offer-recommendation phase in future recommendation cycles. The rating phase corresponds to the Retention phase of the CBR cycle (described in Section 3.1.3).

Table 5.1: Rating of New Offers in the Offer-Based Recommendation Process

Parameter	O_{4426}	O_{4458}	σ	O_{4441}	σ	O_{4460}	σ
numberofhops	1	1	1.0	1	1.0	2	0
origin	Dublin	Dublin	1.0	Dublin	1.0	Dublin	1.0
timeofday	610	505	0.85	620	0.99	605	0.99
destination	LHR	LGW	0.9	LHR	1.0	STN	0.9
via	null	null	1.0	null	1.0	Shannon	0
stopovertime	0	0	1.0	0	1.0	480	0.47
price	31	149	0.46	31	1.0	65.99	0.84
totaltime	75	80	1.0	80	1.0	605	0.48
carrier	Aer Lingus	Aer Lingus	1.0	Aer Lingus	1.0	Mixed	0.5
Selected	true	false		false		false	
Similarity			8.21		8.99		5.18
Rating Score	1.0	0.91		1.0		0.58	

Phase 2: Recommending New Offers

The offer-recommendation phase begins when the PTA receives an offer-set from the broker agent. The PTA takes each offer from the offer-set and converts it into an offer-case. It presents this case to the similarity retrieval engine as a target case and retrieves the k most similar cases from the user's personal offer case-base (Retrieval). It calculates a recommendation score by averaging the rating-scores of these k cases which is allocated to the current offer-case (Re-use). Each offer in the offer-set is given a recommendation score in this way and the offer-set is ordered based on these recommendation scores. This process is illustrated in more detail in Figure 5.3 where $k = 3$.

Table 5.2: Recommendation of New Offers in the Offer-Based Recommendation Process

Parameter	O_{new}	O_{4458}	σ	O_{4500}	σ	O_{4441}	σ
numberofhops	1	1	1.0	1	1.0	1	1.0
origin	Dublin	Dublin	1.0	Dublin	1.0	Dublin	1.0
timeofday	610	505	0.85	710	0.86	620	0.99
destination	FCO	LGW	0.0	CDG	0.0	LHR	0.0
via	null	null	1.0	null	1.0	null	1.0
stopovertime	0	0	1.0	0	1.0	0	1.0
price	148	149	1.0	200	0.76	31	0.47
totaltime	240	80	0.84	155	0.92	80	0.84
carrier	Aer Lingus	Aer Lingus	1.0	Aer Lingus	1.0	Aer Lingus	1.0
Similarity			7.69		7.54		7.3
Rating Score		0.91		1.0		1.0	
Score	**0.97**						

Table 5.2 shows how the recommendation score for offer O_{new} is calculated. The 3 most similar cases to O_{new} are O_{4458}, O_{4500} and O_{4441}. The table shows the local similarity values for these offers as well as their global similarity values (feature weights are also equally weighted in this similarity profile). The recommendation score for O_{new} is calculated as the average of the rating scores of these three nearest neighbours (`0.97`). Each of the offers in the current offer-set are given

a recommendation score in this way and the offer-set is ordered based on these scores. It should be noted that Table 5.1 described how O_{4458} and O_{4441} got their rating scores.

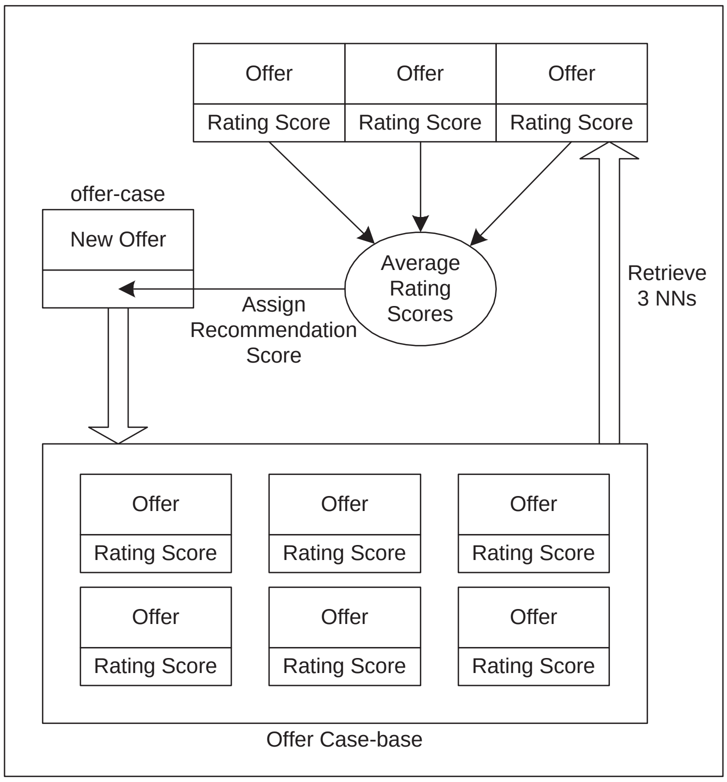


Figure 5.3: Offer-Based Recommendation: Recommending New Offers

The offer-recommendation phase corresponds to the Retrieve and Reuse phases of the CBR cycle (described in Section 3.1.1 and 3.1.2)

Problems with the Offer-Based Approach

Although offer-based recommendation is good at making recommendations (which we document in Section 7.2) there are a number of limitations with the approach. Offer-based recommendation ignores the context in which the request was made. We believe that context is an important factor to take into account in making recommendations. We believe users will have different travel preferences depending on the type of flight they are making, e.g. users may have different preferences when purchasing long distance flights than they would have for shorter flights. Another shortcoming is that offer-based recommendation uses a single similarity profile for every user. This means that the system is making the assumption that each user places the same relative importance on features — this is clearly inadequate. Unfortunately, it is difficult to personalise the offer similarity profile within this recommendation framework.

With these problems in mind, we developed a recommendation strategy that makes contextual recommendations and is capable of personalising the relative importance of features. We call this strategy *session-based* recommendation. Before we describe our session-based recommendation implementation, we need to clarify that session-based recommendation and offer-based recommendation are totally distinct strategies and there is no interdependence between them. From this point onwards we will confine our discussion to the session-based recommender.

5.2 Session-Based Recommendation

Session-based recommendation is more complicated than offer-based recommendation. The PTA uses the selections the user made in similar sessions in the past to make recommendations in new sessions. This approach is similar to the two-stage recommendation approach used by Ricci et al.'s DieToRecs system that was described in Section 3.6.1. We believe that by incorporating context into the recommendation process the PTA will improve its recommendation accuracy. The PTA makes the assumption that the request the user made is a good indicator of the context in which the offer-set is created. It uses information from the session with the most similar context to the current session to drive recommendation.

Session-based recommendation depends on two phases. The first is the *context-matching* stage where the PTA attempts to find the session from the user's history of past sessions with the most similar context to the current session. It does this by finding the session with the most similar request to the current request. The second phase is the *offer-recommendation* phase where the PTA orders the current offer-set based on their similarity to the flights purchases in the session retrieved in the context matching phase. Figure 5.4 shows a diagram of the session-based recommendation process. The two recommendation phases are explained in more detail in the remainder of this section.

The two recommendation phases use two different types of case. The first phase uses *request-cases* and the second phase uses *offer-cases*. Request-cases represent sessions where the problem component is represented by the request features, and the solution is represented by a reference to the selected offer(s) from that session. Figure 5.5 shows an example request-case in CBML format. This case represents a request made by user johnlo for a 9 day trip to Rome from Dublin, leaving on a Friday and returning on a Sunday. The case structure document for request-cases is shown in Appendix C.2 and the similarity profile document is shown in Appendix C.3. The solution feature is `sessionID`. This acts as a reference to the offer(s) that were selected in this session.

The offer-cases are almost identical to the cases described in the offer-based recommender system (see Figure 5.1), except there is no rating feature in these offer-cases. Each user-model contains a case-base of request-cases representing that user's interactions with the PTA.

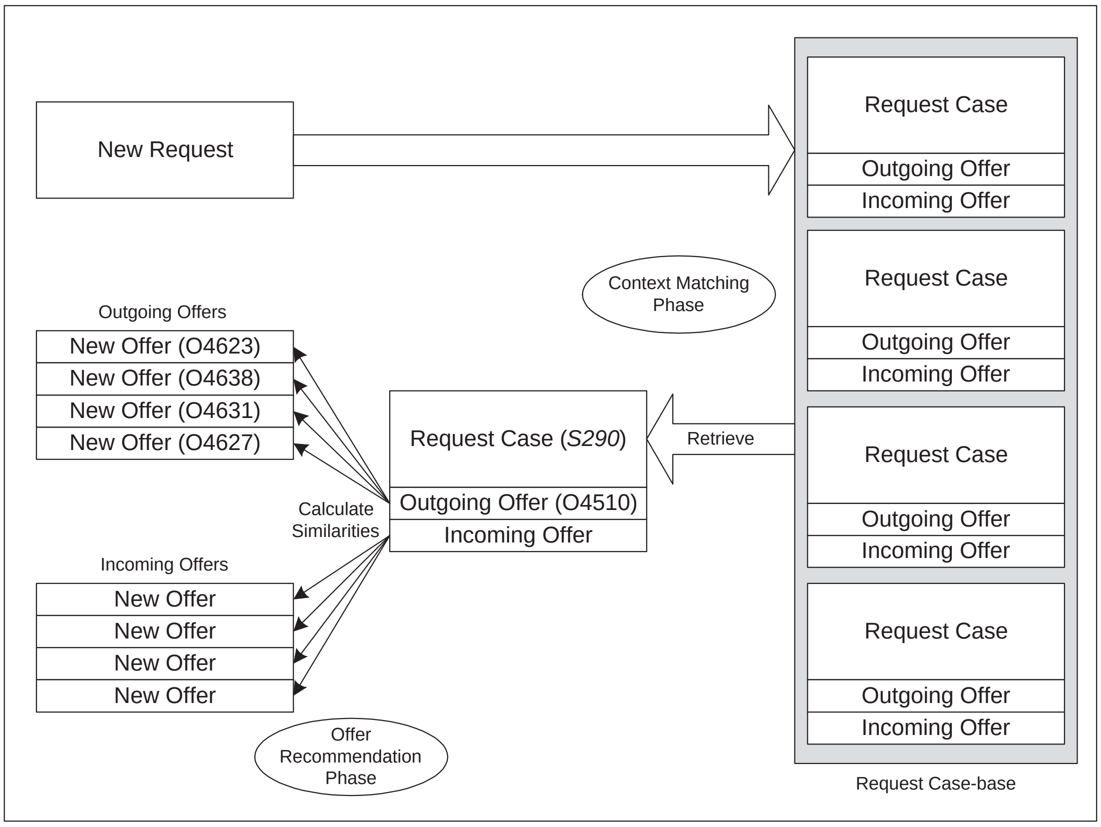


Figure 5.4: Session-Based Recommendation

```
<case name="johnlo-session293-request412">
          <username>johnlo</username>
          <origin>DUB</origin>
          <destination>Rome</destination>
          <distance>187</distance>
          <departuredayofweek>Fri</departuredayofweek>
          <returndayofweek>Sun</returndayofweek>
          <duration>9</duration>
          <tickettype>return</tickettype>
          <sessionID>293</sessionID>
</case>
```

Figure 5.5: An Example Request-Case in CBML Format

Phase 1: Context Matching

At the beginning of each new recommendation cycle the PTA takes the user's request and converts it into a new request-case. This is a target case; there is no solution component attached — the solution is attached after the user selects a preferred offer. The context phase retrieves the most similar request-case from the user's request case-base. The solution feature of a request feature (`sessionID`) acts as a reference to the offer(s) that were selected from that session. These offer-cases are passed on to the offer-recommendation phase. The context-matching phase is illustrated in Table 5.3. User `johnlo` has made a request for a return flight to Rome ($S_{current}$ — also shown in Figure 5.5). The PTA initiates the context-matching phase and calculates the similarity between $S_{current}$ and the other request-cases in the user's request case-base. This table shows the similarity between $S_{current}$ and two other request-cases; S_{290} and S_{300}. S_{290} is the more similar and so the selected offers from that session are passed onto the offer-recommendation phase.

Table 5.3: Matching the Context of a Session

Parameter	$S_{current}$	S_{290}	$\sigma\,(S, S_{290})$	S_{300}	$\sigma\,(S, S_{300})$
sessionID		290		300	
origin	Dublin	Dublin	1.0	Dublin	1.0
destination	Rome	Barcelona	0.0	Bologna	0.4
distance	187	142	0.89	165	0.95
departuredayofweek	Fri	Fri	1.0	Mon	0.0
returndayofweek	Sun	Sun	1.0	Tue	0.33
duration	9	2	0.0	8	0.86
Similarity/Score	N/A		3.89		3.54

Phase 2: Offer Recommendation

The offer-recommendation phase starts when it receives a pair of selected offer-cases from the context-matching phase. These offer-cases, we will call them target cases, represent the offers that the user selected in the most similar session to this one. The PTA calculates the similarity between the target cases and the offer-cases in the current outgoing and returning offer-sets. These offer-sets are then ordered based on their similarity to the target cases.

The offer-recommendation phase is illustrated in Table 5.4 as a continuation of the context-matching example illustrated in Table 5.3. The PTA uses the selected offers from the most similar session (S_{290}) to rank the offers in the current offer-set. In this example, the outgoing selected offer from the retrieved session was O_{4510}. The PTA calculates the similarity between this offer and the current outgoing offer-set. Three offers from this set are shown, as well as the local similarities (there were 24 outgoing offers in total in this offer-set). The bottom two rows show the global similarity values and ordering indices. The three offers will be presented in this order.

Table 5.4: The Offer-Recommendation Phase

Parameter	O_{4510}	O_{4623}	σ	O_{4638}	σ	O_{4631}	σ
numberofhops	1	1	0.5	2	0.0	2	0.0
origin	Dublin	Dublin	1.0	Dublin	1.0	Dublin	1.0
timeofday	435	610	0.76	465	0.96	970	0.26
destination	Barcelona	Rome, FCO	0.0	Rome, CIA	0.0	Rome, CIA	0.0
via	null	null	1.0	London, STN	0.0	Paris, BVA	0.0
stopovertime	0	0	0.5	120	0.43	185	0.40
price	89	148	0.37	43.73	0.6	54.98	0.58
totaltime	215	240	0.49	400	0.41	455	0.38
carrier	Aer Lingus	Aer Lingus	1.0	Ryanair	0.0	Ryanair	0.0
Similarity	N/A		5.62		3.40		2.62
Ordering	N/A	1/30		2/30		24/30	

5.2.1 Further Recommender Components

Other work in the area of personalised recommendation showed further possibilities of increasing recommendation accuracy. We implemented two further components that were added to the session-based recommendation engine:

Personalising Global Similarity Measures This involves personalising the recommendation process itself, by learning personal feature weights. We describe this strategy and the application of these weights in Section 5.3.

Collaborative Recommendations These involve utilising the experiences of other users to make recommendations in scenarios where the current user has little experience. We describe these strategies in Section 5.4

The remainder of this chapter describes these approaches.

5.3 Learning Personal Similarity Profiles

Up to this point, our learning focussed on the case-base knowledge container. This section describes our work in the learning and personalisation of the global similarity measure. The definition of good similarity measures is critical to the success of CBR systems in general and the same is true of our case-based recommender system. Our session-based recommender is dependant on two separate similarity profiles to drive recommendation: the request-profile — used in the context-matching phase; and the offer-profile — used in the offer-recommendation phase. The request-profile determines the context of the travel request and the offer-profile represents the recommendation criteria for individual offers. Traditionally, CBR systems have depended on domain experts to design similarity measures but we see a potential for improving them by incorporating the knowledge contained in user feedback. In Section 3.4 we described work in this area by Bonzano et al. (1997), Branting (2001) and Stahl et al. (2001; 2002; 2003; 2004) which demonstrated improved retrieval and recommendation accuracies.

We have already described the components of the user-interaction that are stored in each user-model. Now we introduce the concept of storing the personalised similarity profiles alongside them. Although similarity profiles are made up a set of feature weights, local similarity measures and an amalgamation function we will personalise only the feature weights.

Unless the user consistently 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 features. This aspect of personalisation is dependent on a feature weight learning algorithm. The remainder of this section describes the implementation of the PTA's feature weight learning algorithm and its application to learning the feature weights of the offer-profile.

5.3.1 Feature Weight Learning

Our feature weight learning algorithm learns weights in a deterministic goal-driven manner. At its most basic level, it attempts to re-weight the features in such a way that maximises the position of the selected offer in the ordered-set. It should be emphasised that this is a retroactive act in the same way that the storage of a case in the retention stage of the CBR process is retroactive. In other words, the set of learned feature weights are learned too late to be applied in the current session, but are saved in the hope that they may be useful in future recommendation cycles. Feature weight learning is only attempted in the event of a recommendation failure. A recommendation failure occurs when the user selects an offer other than the offer that was ordered highest by the recommender.

We infer the relative importance of features by comparing the PTA's recommendation ordering of the offer-set with the actual selections of the users. By comparing the eventual selection of a preferred offer by the user with this ordering we can generate a recommendation accuracy, A_{rec}. This accuracy is calculated from the position of the selected offer in the ordered offer-set (*index*) as follows:

$$A_{rec} = 1 - \frac{index - 1}{N - 1} \qquad \text{if} \quad N > 1 \tag{5.1}$$

$$A_{rec} = 1 \qquad \text{if} \quad N = 1 \tag{5.2}$$

where N is the size of the offer set. It should be noted that a higher ranking indicates that *index* is numerically lower. The highest (or best) possible ranking is when *index* is 1, the lowest when it is N. If the selected offer was ranked third in an offer-set with 10 offers A_{rec} would be 0.78 (or 78%). 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* would be 6 (two higher ranked plus four equal ranked offers) and A_{rec} would be 0.44 (or 44%). In this way, a recommendation accuracy of 1 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 and follows in our policy of reducing cognitive load. Sessions with return flights have two retrieval accuracies — one for the outgoing and one for the return offers. Feature weight learning is attempted on both the outgoing and return offer-set and may lead to two sets of feature weights to be learned in a single session.

Ideally the PTA should be able to improve the recommendation accuracy by altering the feature weights in such a way that the selected case is ordered highest in the offer-set. However, this ordering depends on the offer(s) that were received from the context-matching phase of recommendation. Because the recommender is tied to this target-case there may exist higher ranked cases that can never be ordered lower than the selected case. This occurs because the features of those (hypothetical) cases are all more similar to the target case than the selected case's features (i.e. they *dominate* the features in the selected case). So rather than attempting to order the selected case highest, the PTA attempts to order it as highly as possible.

The first stage of the feature learning process involves calculating the local similarity values (σ) for every case ranked higher than *index*. It is important to note that local similarities are independent of a feature's weight. The feature-weight learner then calculates the local similarity difference ($\Delta\sigma_f$) for every feature in these higher ranked cases. The local similarity difference for a feature of a case is the difference between the local similarity values of that feature between that case (C_i) and the selected offer-case (C_{index}). $\Delta\sigma_f$ allows us to calculate whether the selected offer-case dominates the higher ranked case for feature f (i.e. if $\Delta\sigma_f > 0$). The feature-weight learner uses the local similarity differences to determine whether or not a case is beatable, (i.e. whether any of its features can be dominated by those in the selected case):

$$Beat\,(C_i) = true \quad \text{if} \quad \exists f \in C_i \quad \Delta\sigma_f > 0 \tag{5.3}$$

All Cases where $Beat\,(C_i) = true$ are beatable by the selected case if we re-weight the feature weights. The process of determining whether a case is beatable is illustrated in Table 5.5 (this example continues our Dublin to Rome scenario). In this example, the user selected O_{4638} which had been ranked second by the recommender.

O_{4623} was the only offer that was ranked higher than the selected offer and so we calculate whether it is beatable. The first column shows O_{4623} and the second column shows the selected offer. The local similarity values for each of their features are shown and in the third column we see the local similarity differences ($\Delta\sigma$), whether or not learning of that feature weight would be useful (f_{learn}). Clearly $Beat\,(O_{4623}) = true$ since $\Delta\sigma < 0$ for two features (`timeofday` and `price`).

The next stage in learning involves using the similarity differences in the highest ordered beatable case to alter the feature weights in such a way as to improve the ranking of the selected case

Table 5.5: Determining the Increment Direction when Altering Feature Weights

Parameter	O_{4623}	σ	O_{4638}	σ	$\Delta\sigma$	f_{learn}
Ordering	1/24		2/24			
Selected	false		true			
numberofhops	1	0.5	2	0.0	0.5	true
origin	Dublin	1.0	Dublin	1.0	0	false
timeofday	610	0.76	465	0.96	-0.2	true
destination	Rome, FCO	0.0	Rome, CIA	0.0	0	false
via	null	1.0	London, STN	0.0	1.0	true
stopovertime	0	0.5	120	0.43	0.07	true
price	148	0.37	43.73	0.6	-0.23	true
totaltime	240	0.49	400	0.41	0.08	true
carrier	Aer Lingus	1.0	Ryanair	0.0	1.0	true

with respect to the beatable case. Feature weights are updated using the following equations:

$$w_f = w_f \times increment \quad \text{if} \quad \Delta\sigma_f < 0 \tag{5.4}$$

$$w_f = w_f / increment \quad \text{if} \quad \Delta\sigma_f > 0 \tag{5.5}$$

where *increment* is a constant greater than 1. The increment value is analogous to a learning rate — higher values lead to faster learning at the cost of accuracy. We achieved good results (documented in Section 7.4) with `increment = 1.1`. In the example in Section 5.5 we increase the value of the `price` and `timeofday` features and reduce the `numberofhops`, `via`, `stopovertime`, `totaltime` and `carrier` features. The weights are then normalised and the whole process is repeated. As the algorithm iterates, the highest ranked beatable case will be passed out by the selected case in the ordering. However, it is possible that before this happens the selected offer may be passed out by another beatable case. When this happens there may be a change in the direction of learning (since the new beatable offer will have different σ_f values). As we iterate through the algorithm we check for an improvement in *index* (i.e. a reduction in its value). If this occurs we save the feature weight values as the best overall weights. The algorithm continues iterating until it reaches one of its stopping criteria. At this point the best overall weights are returned as the learned set of feature weights. The stopping criteria are:

Stopping Criterion 1 *The Algorithm breaches an iteration limit*

Stopping Criterion 2 $index = 1$

Stopping Criterion 3 $Beat\,(C_i) = false \quad \forall i < index$

Criterion 1 is to avoid the possibility that the algorithm will stagnate (e.g. two beatable cases may be opposed to each other and lead to flip-flopping) but in reality this criterion is rarely satisfied. Criterion 2 indicates that we have reached the overall optimal solution and that no further learning is necessary. Criterion 3 indicates that no further improvement in *index* is possible. Clearly by

Criteria 2 and 3 many offer-sets will not trigger learning of the feature weights since it would be unnecessary or impossible. Therefore we only store a set of learned feature weights with an offer-set if feature weight learning was triggered and the value of *index* was reduced.

5.3.2 Learning Contextual Feature Weights

Thus far, we have described the application of our feature weight learning algorithms to the offer similarity profile. However, our session-based recommendation algorithm also uses a second similarity profile in the context-matching phase of recommendation — the request similarity profile. We investigated the idea of applying our feature weight learning techniques to improving and personalising this similarity profile.

In the previous section we described how during the recommendation phase the recommender is tied to the session with most similar context, and that there are occasions where it is impossible to learn an optimal set of feature weights. We propose that in these circumstances, the context-matching phase retrieved an incorrect session. We believe that our feature weight learning techniques could be used to correct these failures by improving the request similarity profile. However, since these failures occur less often (by an order of magnitude) we would require a huge amount of data to learn personalised request feature weights. This approach is also computationally expensive and would be highly susceptible to over-fitting. Although we developed a prototype of this strategy, our results were inconclusive and are not presented in this thesis.

5.3.3 Applying Learned Feature Weights

In this section we described how the PTA attempts to improve the similarity profile by learning an optimal set of feature weights and storing them with the session in which it was learned. We now describe two ways that the PTA can use these weights to improve recommendation accuracy.

Session-Weights

Our session-weights approach involves treating the similarity profile as part of the context of the session. It assumes that the similarity profile is representative of a user's selection criteria for a session and that users will use the same selection criteria in similar sessions. When the context-matching recommendation phase returns a most-similar-session and selected offers from that session it also retrieves the set of feature weights that were learned in that session. These weights are applied in the offer-recommendation phase. If no set of feature weights was learned in the previous session we make the assumption that since the original feature weights could not be improved upon we should use them in the current recommendation cycle. We have performed evaluations of this approach in Chapter 7 that show that there are significant benefits to the application of learned feature weights in this way.

User-Weights

Although our session-weights approach makes the assumption that selection criteria are contextually sensitive, it is probable that there are common selection criteria that the user applies in every session. If this is true then we should attempt to incorporate the learned weights of each session into a set of user-weights that are used in every new recommendation cycle. We incorporate these by aggregating the learned session-weights into a set of user-weights. If the user is using a consistent set of selection criteria, these weights should outperform session-weights since some sessions do no trigger feature weight learning and so have no associated feature weights.

A potentially more serious issue with session-specific feature weights is the problem of over-fitting. Previous research has shown that feature weighting algorithms tend to over-fit the data (Kohavi et al., 1997). We believe that over-fitting is unavoidable in this domain due to the lack of data. However, we believe that the aggregation of individual sets of feature weights from numerous learning incidents will lead to a more general set of weights that is less susceptible to noisy data. A further discussion of the over-fitting problem as well as an evaluation of the application of user-weights to offer recommendation is given in Section 7.4.2.

5.3.4 Learning Local Similarity Measures

Thus far our discussion on learning similarity profiles has focused on the learning of feature weights. However, there has also been some research done recently in the field of learning local similarity measures by Stahl and Gabel (2003; 2004). They use evolutionary algorithms to drive changes in the similarity profile. We believe that this technique suffers very badly from over-fitting and due to the lack of data on our part we feel that such an approach is unsuitable for our application. However, we do make subtle changes in elements of our local similarity with respect to our `origin`, `destination` and `via` features. These measures use taxonomy difference functions (described in Section 4.5.2) to capture the relationships between geographical locations. The taxonomy tree is shown in the left frame of Figure 5.6.

Local similarity is related to the number of branches that separate the leaf nodes of the two feature values. We have defined a `complex` local similarity measure for these features that assigns each branch with a cost or attenuation level. As the distance across the tree between two node values is calculated, the branch attenuations are summed. Similarity is inversely related to the sum of these attenuations. All branch attenuations in this similarity measure are initially set to 1.

Due to the fact that users make requests for particular cities (rather than for airports) each offer in an offer-set is guaranteed to have the same city for its origin and the same city for its destination. Because of this, every airport in an offer-set will have the same local similarity value because the distance across the tree to the feature value of the previously selected offer is the same for each value. Therefore it is impossible to learn improved feature weights for those features;

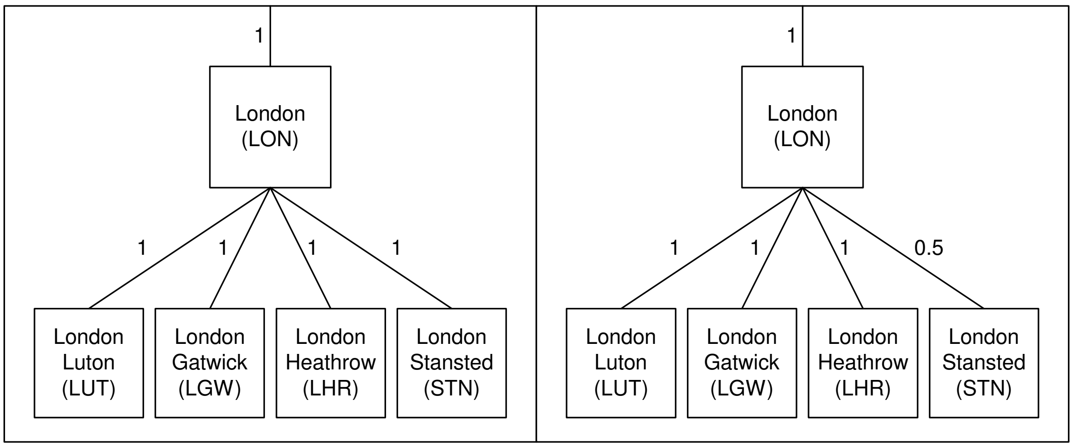


Figure 5.6: Altering the Local Taxonomy Similarity Measure

because $\Delta\sigma_f = 0 \quad \forall f \in \{origin, destination, via\}$.

To allow us to perform learning at this level, the PTA automatically revalues the attenuation values of the branches in the tree connected to the airports of that city, incorporating a measure of utility into the local similarity measure. This is done by reducing the attenuation of the branch connecting the city with the selected airport. This process is illustrated in Figure 5.6. The right frame of the figure shows the new taxonomy tree with its revised attenuation weights. The attenuation of the branch connecting `London` with `London Stansted` has been reduced from 1 to 0.5. This means that `London Stansted` is now less distant and more 'similar' to every other airport in the tree than the other London airports. This updated taxonomy is part of the similarity profile in that user's user-model and demonstrates how we personalise local similarity (albeit at a subtle level).

Although we did not implement any other local similarity measure learning techniques, we have some ideas regarding how this could be done. Utility could be incorporated in the form of sensitivity to difference in feature values. Consider the feature `price`, which describes the price of a flight. The similarity graph for the `price` feature is shown in Figure 5.7. A difference in price of greater than €200 results in a similarity of zero. However, if our user-feedback suggested that users were more sensitive to differences in price, we could move point `[200,0]` towards point `[0,0]` and vice versa. In Figure 5.7 we shift the point to `[100,0]` and generate the dashed graph. This would lead the offer-recommendation phase to focus on offers that were much closer in price to the previously selected offer.

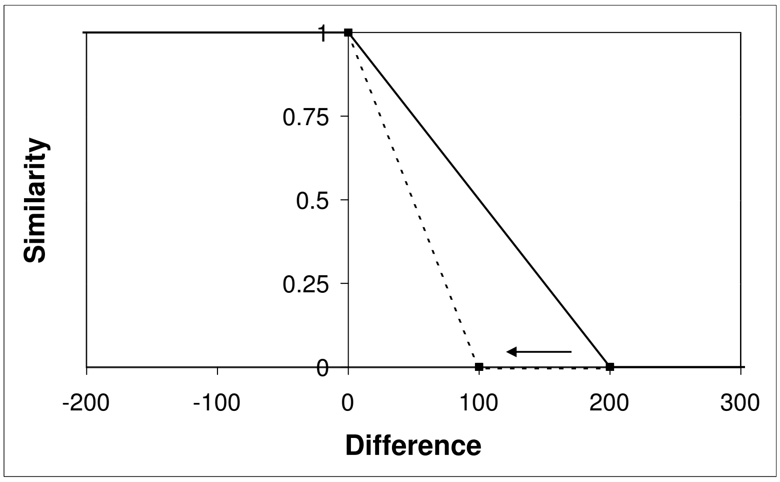


Figure 5.7: Altering the Local Numeric Difference Similarity Measure

5.4 Collaborative Recommendation

In Section 3.5 we described how cooperative and collaborative CBR can be used to improve performance in multi-agent CBR systems. In the PTA application, each user-model can be viewed as an agent. The differences in approach between this view and the view proposed by Plaza et al. (1997) and Prasad et al. (1996) is that although the recommendation process for individual agents are identical, the parameters are different. Different agents may have different solutions for the same problem. This is because users have different travel preferences and there is no single optimal offer for every user in a given offer-set. There are two main issues that need to be resolved before successful collaboration can occur:

Competence/Confidence It is not appropriate to always seek collaboration, often the user's own user-model will have enough information to make a good recommendation. Therefore it is necessary for the recommender to know when it is appropriate to make a recommendation with the user's own user-model and when to seek collaboration with other users. To do this the recommender must be able to gauge the competence of a user-model to make an accurate recommendation in a given session a priori.

Finding Suitable Collaborators If the PTA decides to initiate collaboration it is important to determine the suitable neighbouring user-models to include in the collaborative process. These must have similar preferences to the initiating user-model and also be confident of providing an accurate recommendation.

Confidence is estimated during the context-matching phase of recommendation. The similarity of the most similar previous request to the current session request is used as a measure of the confidence in solving the current problem. This follows from our assertion in Section 3.5 that an agent who has experience of a similar problem to the current one is expected to be competent to solve the current problem. If an agent's own competence score is below a certain threshold then collaboration is triggered. It should be noted that this is a collaborative approach rather than a distributed one in that the originating user-model is responsible for the actual recommendation (i.e. analogous to Plaza et al.'s *collCBR* approach described in Section 3.5).

Once collaboration is triggered it is necessary to find the most suitable neighbours to collaborate with. In Section 5.3.1 we described how we learn feature weights for each offer-set and combine them into a set of user-weights. We make the assumption that users with similar user-weights will be suited to sharing case data. We use the following distance measure (the inverse average Euclidean distance) to estimate the similarity between two users U_1 and U_2:

$$Sim\left(U_1, U_2\right) = 1 - \frac{\sqrt{\sum_{f \in F}\left(w_f - \omega_f\right)^2}}{n} \tag{5.6}$$

where w_f is U_1's feature weight value and ω_f is U_2's feature weight value for feature f. This process is illustrated in Table 5.6 where the similarity is calculated between users `grinness` and `doyledp`. The middle columns show the feature weight values for the users and the rightmost column shows the squared distance between these weights. The bottom row shows the calculated similarity value between `grinness` and `doyledp` (0.862). By contrast `grinness`'s closest user has a similarity of 0.942 and his least similar user has a similarity of 0.782.

Table 5.6: Calculating the Similarity in User-Weights between Users `grinness` and `doyledp`.

Feature Name	grinness	doyledp	$\left(w_f - \omega_f\right)^2$
Origin	1.041	0.933	0.012
Destination	0.777	0.66	0.014
Total Time	1.224	2.272	1.098
Price	2.15	1.628	0.272
Stopover Time	1.467	1.199	0.072
Number Of Hops	0.963	0.771	0.037
Time Of Day	0.415	0.596	0.033
Via	0.427	0.492	0.004
Carrier	0.537	0.45	0.008
$\sum_{f \in F}\left(w_f - \omega_f\right)^2$			1.549
Similarity			**0.862**

We then calculate a confidence score, $Confidence\left(U_i\right)$ for each user $\left(U_i\right)$ in relation to the current session. This score is combined with the user similarity score, $Sim\left(U, U_i\right)$ to yield a collaborator score using the following equation:

$$Collaborator\left(U_i\right) = (1 - \omega) \times Confidence\left(U_i\right) + \omega \times Sim\left(U, U_i\right) \tag{5.7}$$

where ω is a variable relating the two. McGinty and Smyth (2001) use this form of equation to assess the quality of a collaborating agent to solve a target problem. They found 0.75 to be a good value for ω. However they use different equations to assess the similarity between users and to assess the confidence of an agent to solve a problem (they call this *problem coverage*). We performed tests using different values of ω and outline our results in Section 7.5.

The user with the highest collaborator score contributes the selected offer(s) from their closest session to the current user to aid recommendation. It should be noted that the user itself competes in this collaborative step, by calculating its own competence score. If it outweighs the highest collaborator score it performs the recommendation itself. This is done on the assumption that an unconfident recommendation from the user's own case-base is better than an unconfident recommendation from another user's case-base.

In section 7.5 we show that this type of collaborative approach is not as useful as we hoped in providing accurate recommendations in our this application.

5.4.1 Bootstrap Recommendation

The most obvious application of collaborative CBR in recommender systems is when making recommendations to new users. New users are characterised as having no user-model because they are fresh to the system. Since recommendation in the PTA is based on the user-model (both the case-base and the personalised similarity measures), it is impossible to make recommendations to new users using the recommendation strategies we have discussed thus far. However, collaborative recommendation may hold the key to overcoming this problem.

The problem of recommending offers to new users in this way is known as the *bootstrap problem*. The PTA performs bootstrap recommendations using a stripped-down collaborative approach. The full collaborative approach depends on two phases; competence estimation and user-matching. Obviously the user-matching phase is impossible since new users have no user-weights. Therefore we just use the competence estimation stage of collaborative recommendation. Effectively this means that a new user has access to the combined case-base of all users in the system. We call this the *master case-base*. There is no bias between any user's cases and the recommender treats the master case-base as if it was the user's own personal case-base. In Section 7.5.1 we present results that show the advantages of our bootstrapping approach with respect to recommendations made to new users.

5.5 Conclusions

In this chapter we described the case-based reasoning strategies we have applied to the problem of flights recommendation in the PTA application. We outlined how feedback from the observations of user selections from sets of presented flights (offer-sets) is used to generate models describing user flight preferences. We implemented two different recommendation strategies that use these user-models to make personalised flights recommendations; an offer-based recommender and a session-based recommender.

Our offer-based recommender recommends flights based on their similarity to flights the user has rated in the past. These ratings are automatically generated by the PTA at the end of each session based on the user's selections of preferred offers. There were some limitations with this approach, so we developed and implemented a second recommendation strategy — our session-based recommender.

Our session based recommender takes into account the context in which a request for flights is made. It recommends flights using the knowledge gained from sessions with similar contexts from the user's session history. The PTA uses features from the user's flight request as a gauge of the context of that session. We also describe two components that were integrated into the session-based recommender. The first of these uses observed user selections to personalise the recommendation process by attempting to alter the relative importance of features. We described two ways in which the PTA applies these feature weights — as session-weights and user-weights. The second component uses collaborative recommendation techniques. It applies the knowledge contained in similar user's models to improve recommendation in scenarios where the user's own model may be inexperienced or unconfident. These collaborative strategies can be used to improve recommendations for new and mature users.

In Chapter 7 we describe a set of evaluations that test the utility of our recommender strategies.

Chapter 6

System Implementation

This chapter describes some of the interesting aspects of the implementation of the PTA application and the CBML representation format. While these aspects were introduced in earlier chapters, they are described in more detail here. We discuss some details of the implementation of the PTA web-interface, the PTA's ability to access real flight data, CBML and the Fionn framework. While the details in this chapter are not central to the thesis they had an impact on some of the decisions made along the way.

6.1 The PTA User-Interface

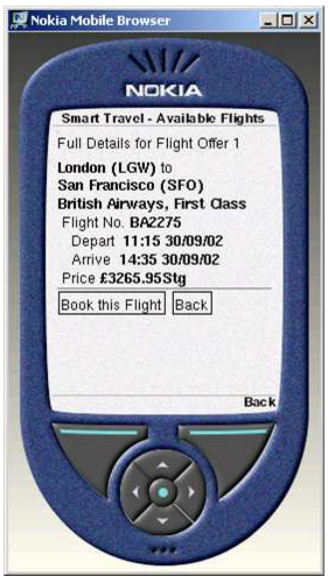
We mentioned in Chapter 2 that users interact with the PTA via a web interface. Since our original target platform was a mobile device, the original implementation of the PTA was in wireless mark-up language (WML — a wireless equivalent of HTML). Figure 6.1 shows a screen shot of the original interface on Nokia's Mobile Browser Simulator. Wireless platforms such as this have a number of shortcomings, including:

Small Screen This was the main problem with wireless devices. The device shown in Figure 6.2 has a screen resolution of only 240 by 320 pixels. To put this in perspective, modern PCs generally offer resolutions of more than 1024 by 768 pixels.

Poor Text Inputting Facilities Wireless devices have limited facilities for inputting data, e.g. small QWERTY style keyboard, phone-keypad or stylus and touch-screen. This makes it tedious for users to input flight queries.

Limited Power Wireless terminals are characterised as having limited computational power and memory, although over time the capabilities of these devices will continue to improve.

In order to overcome the restriction of a lack in screen resolution, intelligent techniques are needed to provide the maximum amount of information to the user on each screen. Even so,


Nokia Mobile Browser

Smart Travel - Available Flights

Full Details for Flight Offer 1

London (LGW) to
San Francisco (SFO)
British Airways, First Class
Flight No. **BA2275**
Depart **11:15 30/09/02**
Arrive **14:35 30/09/02**
Price **£3265.95Stg**

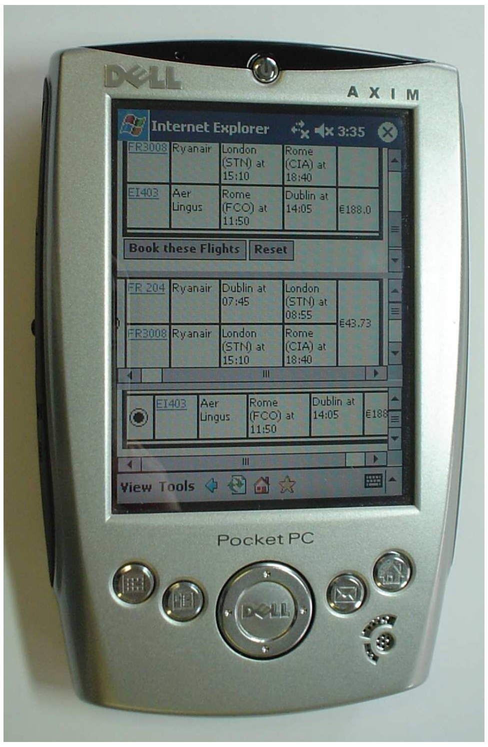
Figure 6.1: The PTA WML Interface

users were frustrated by the difficulty in finding a suitable flight. Even with a flight picked, it was impossible to pass the wireless user onto the vendor web site, because they did not provide a wireless interface. In discussions with users, it emerged that they preferred to access the PTA via a PC-based web browser.

So we decided to move towards a compromise solution. The PTA web site is now implemented in XHTML (an XML-variant of HTML). This is comparable with the current versions of both fixed and wireless web browsers. The screen shots shown in Chapters 2 and 7 show the interface as seen using Mozilla's Firefox Web Browser (v0.92). Appendix A contains a number of screen shots taken using Microsoft's Internet Explorer (v6.0) and Figure 6.2 shows a screen shot of the offers page on a mobile web browser — PocketPC 2003's Internet Explorer browser on a Dell AXIM handheld PDA.

6.2 Travel Service Agents — Web Scraping

Aside from the personalisation aspects of the PTA; the most interesting components of the system were the travel service agents (described in Section 2.2). These had responsibility for providing real-time flights data to the user. We also mentioned that flights providers are wary of granting outsiders access to their flights databases. However, the same flights-providers offer limited access to these databases through their web sites. There has been some work in the last few years in the field of *web-scraping*. Web-scraping involves faking an interaction between a browser and an online flights provider. It involves emulating an interaction with a web site - not just downloading pages, but filling out forms, navigating around the site, and dealing with the HTML received as



FR3008	Ryanair	London (STN) at 15:10	Rome (CIA) at 18:40	
EI403	Aer Lingus	Rome (FCO) at 11:50	Dublin at 14:05	€188.0

FR 204	Ryanair	Dublin at 07:45	London (STN) at 08:55	€43.73
FR3008	Ryanair	London (STN) at 15:10	Rome (CIA) at 18:40	

●	EI403	Aer Lingus	Rome (FCO) at 11:50	Dublin at 14:05	€188

Figure 6.2: The PTA XHTML Wireless Web Interface

a result (Ball, 2003). We use web-scraping libraries developed by the Perl community[1] to scrape flights information from Aer Lingus and Ryanair's web sites.

The Travel Service Agents are implemented in Java and communicate with the broker agents via RMI. However, their web-scraping components are implemented in Perl. We found that the best way of executing Perl scripts using Java was via the Apache web server[2]. Apache has a built-in CGI (Common Gateway Interface) component that allows the execution of Perl scripts. The web-scraping scripts were hosted using Apache. The broker agent called these scripts by sending Apache HTTP requests for them passing the required parameters as HTTP parameters. Apache executed these scripts and returned their results (the flight details) in XML format — specifically in CBML format — as offer-sets. Figure 6.3 shows a diagram illustrating this process.

We wrote scripts to web scrape the Aer Lingus and Ryanair web sites. Apart from the fact that these are the two biggest flyers out of Dublin airport, they both have well designed web sites and are quite easy to scrape for information. They each have a simple form for making flight requests and return their offers in plain HTML. The parameters for these requests are the origin, destination, dates of travel and number of passengers (which are the same as the PTA's own request page). There are two scripts each for the Aer Lingus and Ryanair web sites that accomplish two different

[1]We used the packages `LWP::UserAgent` and `WWW::Mechanize` which are available for download from the CPAN search web site here: `http://search.cpan.org/`

[2]The Apache HTTP Server is an open-source HTTP server available here: `http://httpd.apache.org/`

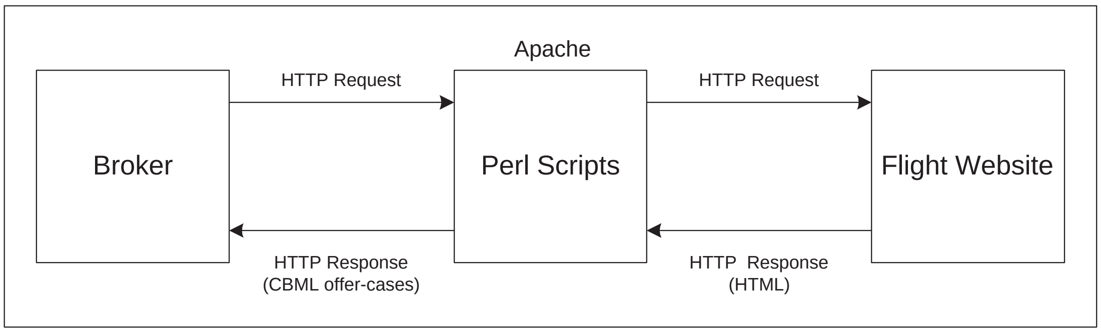


Figure 6.3: An Overview of Web Scraping

tasks. The first polls the web site for general flights information (the *poll script*) and the second satisfies a travel request (the *request script*).

The Poll Script: In Section 2.2 we described how the broker agent has a map describing the possible paths across Europe and how it uses this map to calculate the possible flight paths for a given request. The poll scripts are used to gather the information necessary to generate this map. Hidden within the request form of each web site is information describing the possible combinations of origin and destination for that airline. This information is stored in JavaScript and used by the web site to ensure the validity of a user's request, e.g. an airline may fly to Prague or to New York from Dublin but not from Prague to New York directly. The poll script sends a HTTP request for the web page that contains this JavaScript and translates it into a suitable format for the broker.

The Request Script: The request script allows the satisfaction of a travel request. It takes a broker request and initiates an interaction with the appropriate web site. This interaction is essentially a simulation of what would happen if a user made the same request from a browser, i.e. the front page is loaded, the fields are filled in on the request form and the form is submitted. The script then waits for the response from the web site. This response includes a HTML document containing (among other things) a set of offers. The script then parses this HTML and extracts the offers, converts them into CBML format and returns them to the broker agent. This process is illustrated in Figure 6.4.

The PTA sends an RMI message to the broker agent with the details of the user's request for flights. The broker agent made a set of requests to the Apache Web server which executed a set of request scripts (two Aer Lingus and three Ryanair Scripts). These scripts initiate sessions with the web sites and parse the responses for offers, which they convert into CBML. These are then returned to the broker (via the web server) as CBML (i.e. XML) documents using standard HTTP protocols. The broker agent then uses CBML parsers to convert these documents into case objects which are sent via RMI to the PTA.

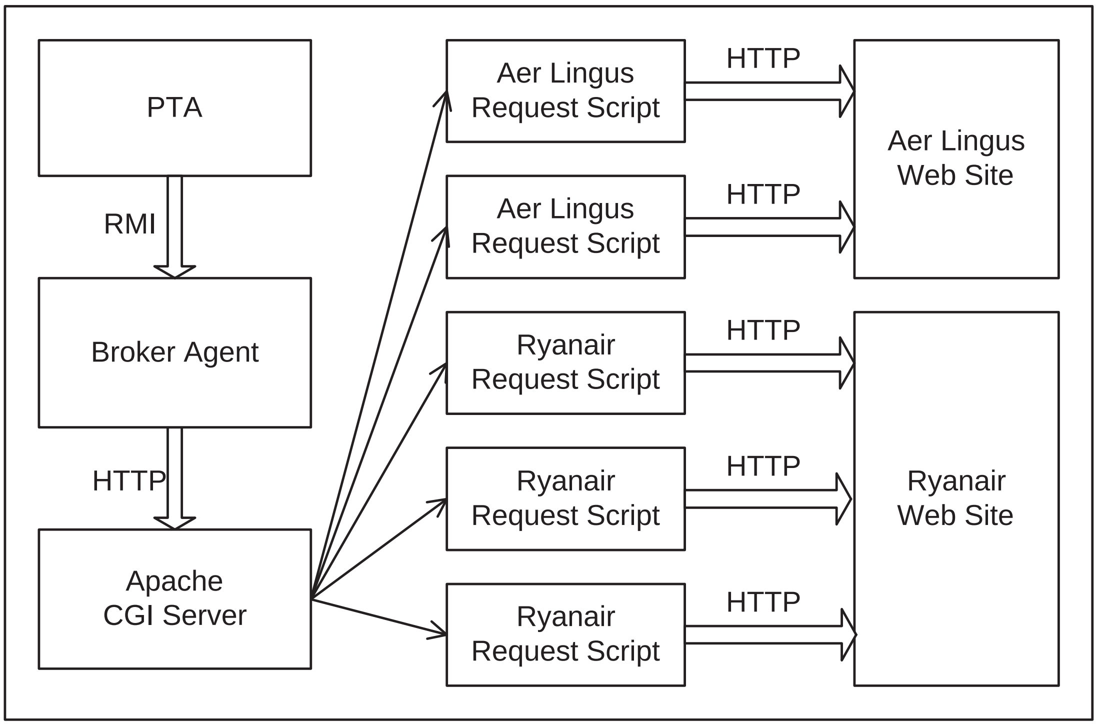


Figure 6.4: Web-Scraping — Information Flow

The web-scrapers provided one further benefit to the PTA system. They provided the information necessary to hand-off the user to the appropriate vendor web site after they have selected a flight. This information was gathered at the same time as the travel offers themselves. Basically, the vendor maintains a session for each user that accesses their web site. Our web scraping scripts are viewed as users by the vendor web site and have session information that identify them. As our scrapers make requests they maintain the necessary information to renew the session at a later time. They pass this information to the broker agent with the offers. The broker ensures that when the user makes a selection they are forwarded to the correct booking page. In effect, the travel service agent holds a session for the user and at the hand-off stage the user takes its place in that session.

Our web-scrapers depend on connections to real web sites, and as such they are subject to a number of issues:

Changes in the web site Although web-scraping is a powerful means of getting real time web data it is dependant on the web site to remain static. Small changes in the structure and operation of the web site can cause the web-scraper to fail completely. In December 2003, during the initial implementation stage (and before user-trials), the Ryanair web-scraper suddenly stopped working. This was a result of a minor change in Ryanair's web-site. The problem was resolved once our scripts were updated but during this period the PTA was unable to offer Ryanair flights. Although this issue did not reoccur it demonstrates the susceptibility of the PTA to sudden critical disruptions.

Timeout Web-scraping is liable to fail in the event that the web site we are scraping is unable to respond quickly (or at all) to our request. This is an inherent problem in web-scraping. Timeout has not been an issue for the PTA application because it is rare. The Aer Lingus and Ryanair web sites both respond to requests within a reasonable time (i.e. usually less than ten seconds). If they take longer than thirty seconds to respond the web-scraping scripts make the assumption that there are no offers available and return empty offer-sets.

Legality of Web-scraping The practice of web-scraping without their knowledge or expressed permission of the travel vendors appears to fall into legal-limbo at this point. Preliminary proceedings on this subject in the United States are coming down on the side of web-scrapers and companies are becoming more open to allowing third parties to scrape their web sites (Chen, 2003b; Chen, 2003a).

6.3 CBML

CBML was introduced and documented in Chapter 4. There follows a more in-depth description of the implementation details of the CBML parsers and the Java CBML interfaces:

CBML Parsers We have implemented two parsers for reading CBML documents; the CBML parser and the case content parser. These are implemented in Java using the Xerces-J[3] parsers. They are validating parsers — as they read CBML documents they validate them against the CBML Schema. Any CBML documents that fail this validation process are rejected. The parser converts Case Structure and Similarity Profile documents into Java CBR objects (`CaseStruct` and `SimilarityProfile` objects). The Case content parsers use these `CaseStruct` objects to validate case content documents against their defined structure.

CBR Interfaces The CBML documents are converted into Java objects by the parsers. These objects have interfaces defined for each of them that allow a third-party application designer to manipulate them easily. Interfaces are defined for Cases (`CBMLCase` — shown in Appendix C.6.1), Case Structures (`CaseStruct`), Feature Structures (`FeatureStruct`), Similarity Profiles (`SimilarityProfile`) and Local Similarity Measures (`SimilarityMeasure` — shown in Appendix C.6.2).

[3]The Xerces Parsers are part of Apache XML Project and are available here: `http://xml.apache.org/`

6.4 Fionn — The Machine Learning Framework

One of the outcomes of our development of CBML was the development of *Fionn* — a framework for the development of CBR systems (Doyle et al., 2004b). Figure 6.5 shows a diagram of the Fionn Framework. The core specifications of Fionn are used by all the other components and CBML makes up an important part of this. CBR objects are represented internally as CBML objects. Case-bases are stored in CBML documents; global similarity measures in similarity profile documents and domain definitions in case structure documents. The other main components of the Fionn core are a suite of classifiers (including k-NNs, support vector machines, neural nets, logistic regression, linear regression and naive Bayes) and an evaluation framework which supports a variety of validation schemes and a selection of error functions.

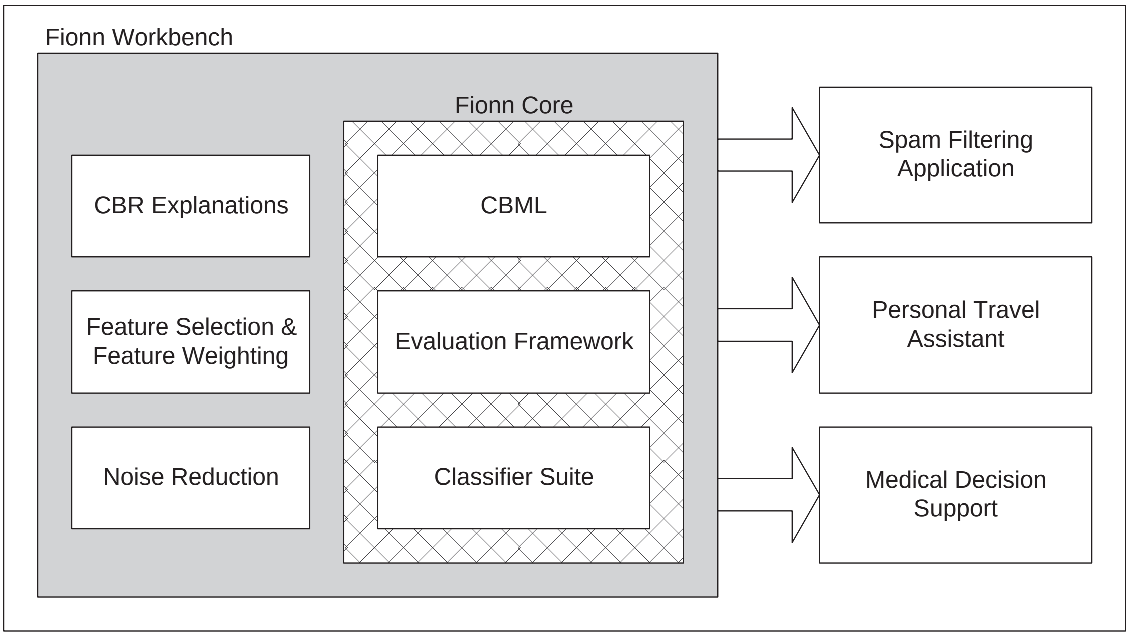


Figure 6.5: The Fionn Framework

Much of the current work being done in the Fionn Framework is in developing the CBR Evaluation, Feature Selection and Weighting, and Noise Reduction components. There are three applications currently in development that use the Fionn Framework:

The Personal Travel Assistant As described in this thesis

Spam Filtering Application Delany and Cunningham are working on a spam filtering application called ECUE (E-mail Classification Using Examples) that dynamically adapts to the changing nature of spam e-mails (2003b; 2004b). Because of the volume of spam e-mail and to its evolving nature their application uses several case-base maintenance techniques that remove noisy and redundant cases (Delany and Cunningham, 2004). Their application uses many features of the Fionn framework including the k-NN classifiers and evaluation framework.

Medical Decision Support Doyle and Cunningham are working on a decision support application that will be used in the medical domain. This application aims to support a doctor in making the decision to admit or discharge a child with bronchiolitis to hospital for observation. The system makes a prediction and backs it up with a compelling explanation based on *a fortiori* arguments (Doyle et al., 2004a). Using the Fionn framework, they were able to concentrate on developing explanations while reusing existing feature selection, noise reduction and evaluation components.

Part of the classifier suite in the Fionn core is an efficient implementation of a k-NN retrieval algorithm. This algorithm integrates elements from an existing efficient retrieval design (Case Retrieval Nets) to improve retrieval speed. It also uses the knowledge stored in CBML documents to maximise retrieval efficiency. The remainder of this section describes Case Retrieval Nets and Fionn's k-NN implementation.

6.4.1 Case Retrieval Nets

The standard k-NN algorithm calculates similarity on a case-by-case basis. This approach is quite inefficient in domains where there is feature-value redundancy and/or missing features in cases, e.g. in the spam filtering domain (Delany et al., 2004b; Delany et al., 2004a). In Section 3.1.1 we introduced the Case Retrieval Net (CRN) (Lenz et al., 1998) as an implementation of a case memory that facilitates efficient retrieval in these circumstances. A CRN is a structured net constructed from the case base to retrieve similar cases to a presented problem-case. CRNs are also able to deal with ambiguous terms, can handle partial cases and are reasonably scalable. They are made up of a number of components:

Case Nodes These represent the cases in the case-base

Information Entities (IEs) These represent feature-value pairs within cases

Relevance Arcs These link case nodes with the IEs that represent them. They have weights relating to the level of importance between the connected IE and the case (i.e. the weight of that IE's feature).

Similarity Arcs These interconnect IEs that refer to the same features. These have weights relating to the similarity between the values of connected IEs. The weight of a similarity arc is equal to the local similarity between the feature-values of the two IEs they connect.

The idea behind the CRN architecture is that if a target case is connected to the net via a set of relevance arcs, and activated, this activation will spread across the net. As activation spreads along arcs in the net, it is attenuated by their weights. Each of the other case nodes will accumulate an activation score representing its similarity to the target case. After activation has spread across the net, the case nodes with the highest activation are the most similar to the target case. The advantages of CRNs are that local similarity calculations are only performed once and that missing feature values are ignored.

6.4.2 Fionn's *k*-NN Implementation

Fionn's *k*-NN implementation uses a combination of case-by-case similarity calculations and CRN optimisations. There is a significant cost associated with initialising a CRN. As such there is a trade-off between time spent initialising the net and faster retrieval times. The CRN-derived speed-ups are counter-productive if there is little or no redundancy in feature values in the case-base. Therefore the speed-ups should not be used unless there are features in the case-base which will benefit from the net structure.

The first step of setting up the *k*-NN is the determination of which features will be indexed. Fionn examines the case structure document and disregards any features that are non-discriminating (i.e. features where `discriminant=false`). It then examines the similarity profile document and disregards features whose weights are zero. The remaining features will be loaded into the *k*-NN.

Fionn then needs to determine which features will benefit from inclusion in the CRN structure. Features that are not included in the CRN structure will have their local similarities calculated on a case-by-case basis. Our *k*-NN implementation has a *net-initiator* function which assesses the features in a domain in order to determine which ones will benefit from inclusion. It reads the case structure document and automatically excludes `double` and `string` feature types as they are unlikely to have much feature-value redundancy. `symbol`, `taxonomy` and `boolean` types will most likely benefit and so they are flagged for inclusion. The `integer` feature type may also be flagged depending on the estimated feature-value redundancy — if the range of an `integer` feature is less than half the size of the case-base it is flagged for inclusion.

Fionn then loads the cases one-by-one into the *k*-NN structure and adds IEs to the net structure as needed. On presentation of a target case, activation is spread to the case nodes through the CRN. The relevance and similarity arcs in the CRN structure are kept in place between retrievals, therefore subsequent retrievals should be quicker. This is because local similarities are effectively cached in the similarity arcs. Next Fionn calculates the similarity contributions of the non-included features. It goes through these features and calculates the local similarities directly between the target case and each case in the case-memory, adding the similarity values to that case's global similarity. On subsequent retrievals, the local similarity values of non-included features will have to be recalculated.

In Section 5.2 we described the context-matching phase of the session-based recommendation process. We also showed a table which showed how the similarity scores were calculated between request S and two requests from the user's request case-base ($S290$ and $S300$). Figure 6.6 shows how Fionn's k-NN actually calculated these similarity values. The case nodes are represented by pentagons. All features were flagged for inclusion in the CRN structure except `distance` and `duration` — because they are `integer` types with a low estimated redundancy. These features are represented by the rectangles above the case nodes and local similarity is calculated between their values directly. The remaining features (`origin`, `departuredayofweek`, `returndayofweek` and `destination`) are shown as IEs (the smoothed rectangles) with their similarity and relevance arcs activated by case S.

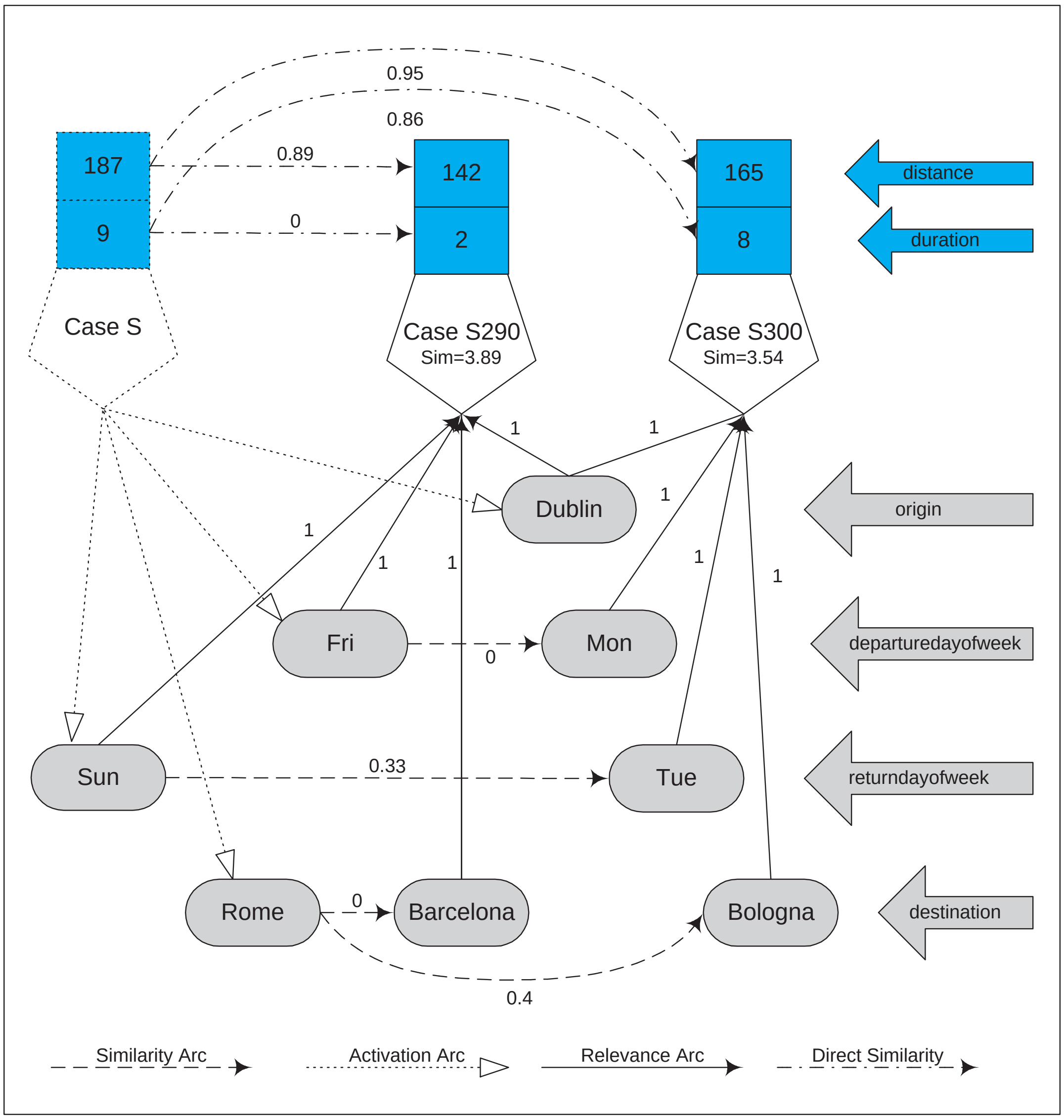


Figure 6.6: Fionn's CRN-Inspired k-NN Implementation

The advantage of the CRN structure can be seen if we count the number of similarity calculations that are performed in this scenario. Similarity calculations are the most computationally expensive part of retrieval. For the features `distance` and `duration` (which are not included in the CRN structure) there were four similarity calculations performed. For the four features that were included in the CRN structure there were also four similarity calculations (half the number of calculations per feature). The magnitude of this disparity increases with the size of the case-base.

6.5 Conclusions

This chapter was something of a diversion from the story of the thesis. However, the issues raised here are due to the specific implementation choices of the PTA application and have a bearing on the readers understanding of the PTA application and the operation of CBML and Fionn.

The PTA user-interface motivated the decisions to reduce user cognitive load to an absolute minimum. Although the PTA was not used by wireless web users in the end, it is clear that the personalisation and intelligent recommendation techniques implemented in this thesis would add value to any such system.

The implementation of the Travel Service Agents posed a particular challenge in the context of the PTA application. We hoped that by allowing users access to real flights information and the capability of actually purchasing flights would lead to more users participating in our user trials. We hoped that these users would provide good data as they bought flights, and this was borne out in the real sessions that were used in the evaluations (which we will describe in Chapter 7). The limitations that using real flight providers brought also had an impact on the operation of the PTA. Because the vendor web sites only took requests in a specific format, and because of time restrictions, we could only allow the user make very specific requests — i.e. a request for a flight on a particular day. Ideally, in a perfect system the user could make a request such as 'I would like a cheap weekend to Paris some time in the next two months'. While this would be technically possible in the current implementation, the request would take several minutes to fulfil and would lead to our request scripts sending the Aer Lingus and Ryanair web sites several hundred HTTP requests. Clearly this is unreasonable.

Sections 6.3 and 6.4 describe implementation details of CBML and the Fionn framework. We also described Fionn's CRN-inspired k-NN retrieval algorithm implementation with an example retrieval event in the context of the PTA application. More information on CBML[4] and the Fionn Framework [5] are available on the web. The CBML Schema document, the CBML and case structure document parsers, the CBR interfaces, Fionn's k-NN implementation and a comprehensive set of Javadocs may also be downloaded from this location. There are also a number of example CBML files and programs that demonstrate how CBML and Fionn work. As work on Fionn proceeds, more resources will be made available here.

[4]The CBML web site is here: `http://www.cs.tcd.ie/research_groups/mlg/CBML/`
[5]The Fionn web site is here: `http://www.cs.tcd.ie/research_groups/mlg/FIONN/`

Chapter 7

Evaluation of the PTA

Recommender systems are commonly evaluated in two distinct ways; in an online manner or offline using machine learning techniques. Online evaluations analyse the live performance of a recommender using users of a running system whereas offline evaluations are performed on a data-set with no further input from real users. Hayes et al. (2002) suggest that it is best to evaluate web-based recommender systems in an online manner. They propose that while offline techniques may provide useful insights, only an online evaluation methodology can truly gauge user satisfaction with a recommendation strategy. They give a number of reasons for this:

Context In most offline evaluations, the context in which a recommendation is made is ignored

Utility of Recommendations Most offline evaluations have no indication of whether recommended resources were consumed

Temporal Constraints Most offline evaluations ignore the order in which recommendation events occurred.

However, they also recognise that while an online evaluation is preferable, it is not always suitable. A successful online evaluation is dependant on a mature fully-engineered system with a large group of regular users. Although the PTA has a number of issues that necessitate the use of an offline evaluation, we implement a *best of both worlds* evaluation framework by incorporating some of those issues that make an online evaluation preferable. Given the selections that have been identified in the user sessions, the evaluations assess how well our recommendation strategies would have predicted them.

The reasons that an online evaluation is inappropriate for the PTA's recommendation engines are mostly to do with the flights domain and the type of user that could be expected to use the PTA. There are two main types of user of an application like this:

Heavy-volume Users These users fly regularly. They tend to travel as part of their work life, e.g. business people. However, because of the connection of their travel with their business they tend not to arrange their travel themselves, leaving this task to an expert based in their corporate workspace. These users often have arrangements with specific (real world) travel agents and for these reasons are unlikely to use our application.

Casual Users These users tend to book flights on their own behalf and travel on them themselves. For this reason they are suitable for an evaluation of our recommendation strategies. However, unlike heavy-volume users they make few flights in a given time, usually less than four trips per year — most trips consisting of two flights.

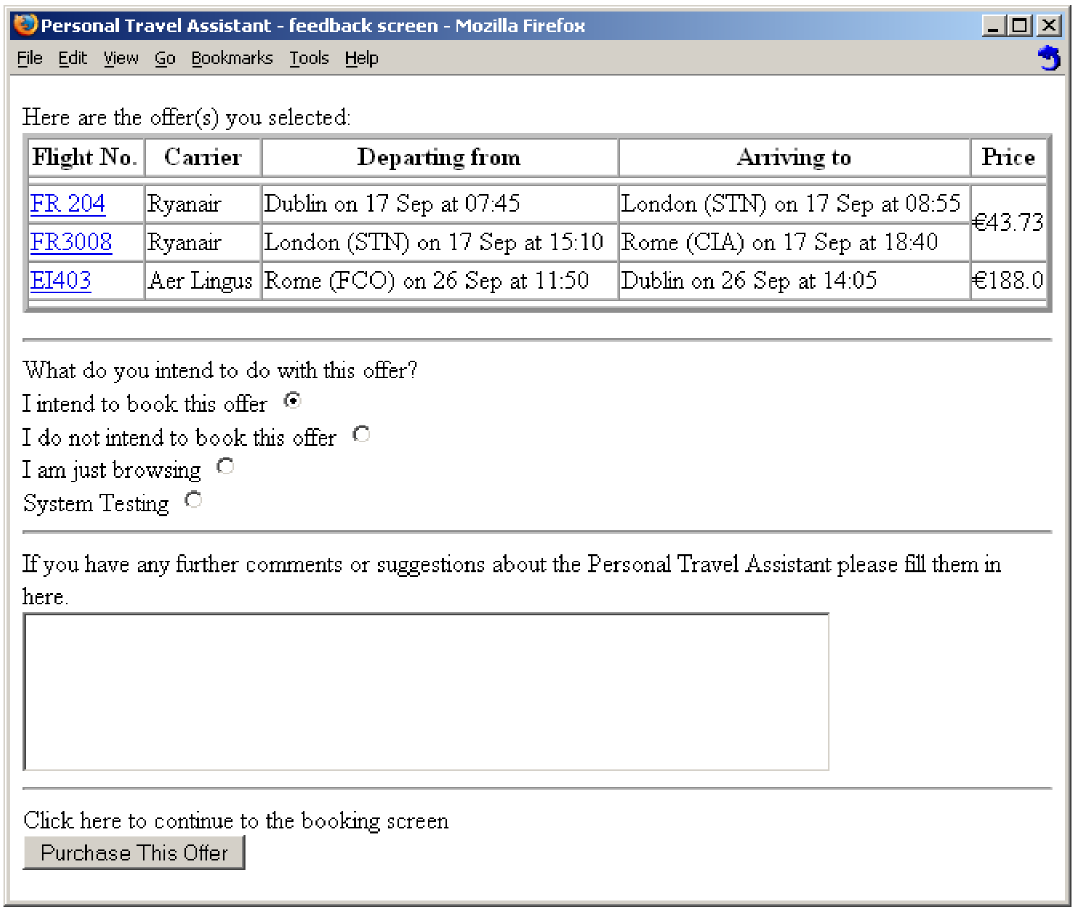
The Personal Travel Assistant was live between December 2003 and June 2004. It offered European flights that were provided by Aer Lingus and Ryanair. During this period twenty-eight users (excluding the author) registered with the system and used it to search for flights. However, because we targeted casual users, there is not a single user with enough data to perform an adequate evaluation.

In order to overcome this problem, we asked these users to undertake a number of scenarios that would generate further user data. Two types of scenarios were described: a request for flights for a weekend holiday away, to a location close to Dublin (e.g. London, Manchester); and a request for flights for a week away to a more distant destination (i.e. one that may require multiple hops, e.g. Rome, Stockholm). The list of cities was selected to maximise the size and diversity of the returned offer-sets — each of the cities were serviced by both airlines with multiple flights per day. These test scenarios are shown in Appendix B.

The test scenarios were broad in that the users had leeway in making a request. In this way, users were encouraged to make requests that suited their tastes, and select offers that they would purchase if the scenario were real. With this in mind, users were given the freedom to choose their own destination from a list, 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. Each user was asked to complete six scenarios; three of each type. These scenarios complement the real sessions that were initiated by users and led to the purchase of actual flights. The evaluations described in this chapter make no distinction between the real and test sessions.

In Chapter 2 we described a user's interaction with the PTA. In order to ensure that we received good data for the evaluations, we forced users to go through one more step in the process. Users were forced to complete a short form asking for information about the selected-offers before they could proceed to the booking page(s). Figure 7.1 shows a screen-shot of this form. The user has selected a two-hop flight to Rome with Ryanair and a single hop return flight with Aer Lingus (this screen-shot shows the continuation of the example interaction we described in Chapter 2) and indicated that they intend to book the flight. This positive feedback ensures that this session will be used in our evaluation, since we only use sessions that resulted in a booked offer (our evaluations ignore incomplete sessions or sessions where the user was 'just browsing'). On selecting the purchase button, the user will be forwarded to two Ryanair booking screens (one for each hop) and one Aer

Lingus booking screen. At this point the user is no longer interacting with the PTA but dealing directly with the flights vendors through their web sites.


Personal Travel Assistant - feedback screen - Mozilla Firefox
File Edit View Go Bookmarks Tools Help

Here are the offer(s) you selected:

Flight No.	Carrier	Departing from	Arriving to	Price
FR 204	Ryanair	Dublin on 17 Sep at 07:45	London (STN) on 17 Sep at 08:55	€43.73
FR3008	Ryanair	London (STN) on 17 Sep at 15:10	Rome (CIA) on 17 Sep at 18:40	
EI403	Aer Lingus	Rome (FCO) on 26 Sep at 11:50	Dublin on 26 Sep at 14:05	€188.0

What do you intend to do with this offer?

[✓ I intend to book this offer]
I do not intend to book this offer
I am just browsing
System Testing

If you have any further comments or suggestions about the Personal Travel Assistant please fill them in here.

Click here to continue to the booking screen

Figure 7.1: The PTA Feedback screen

Over the course of our evaluation the PTA acted as a *dumb* system and did not actually provide recommendations to its users. This was done because there was a dearth of user-sessions. Since recommenders are generally more successful if there were pre-existing user models, we decided to let this dumb PTA system go live as a kind of bootstrap system. We decided that it was better to make no recommendations in a live context than risk making bad recommendations and losing the trust of our users.

7.1 The Evaluations

We implemented a number of recommendation strategies and performed evaluations on each of them to prove their usefulness. These evaluations were performed offline and each of them used the same data; that is to say, the same user-interactions (sessions). Since the same data were used in each evaluation it is fair to compare the results of each evaluation directly.

In order to explain how the recommendation strategies are compared, it is important to give a brief recap on the user's interaction with the PTA. This comprises of three parts: the issuing of a request for a flight by the user; the presentation of an offer-set to the user by the PTA; and the selection of a preferred offer by the user. The recommendation process occurs just before the second part by generating recommendation scores for each of the offers and using these to order the offer-set for presentation. Our evaluations compare this ordering with the user's selection of a preferred offer and generate a recommendation accuracy (A_{rec}) using the equations described in Section 5.3.1. The general form of that equation is reproduced here:

$$A_{rec} \quad = \quad 1 - \frac{index - 1}{N - 1}$$

where *index* is the position of the selected offer in the ordered offer-set and N is the size of the offer set.

Our offline evaluation proceeds as follows: A user-model is generated from all that user's sessions except the session being tested. A recommendation is then made for this session on the basis of that user-model. In this way, each session is tested as if all of the other sessions proceeded it temporally. This is a *leave-one-out* approach whereby the recommendation accuracy for a session is calculated by using the inferences made on all the data except the rating itself. This approach is illustrated in Figure 7.2.

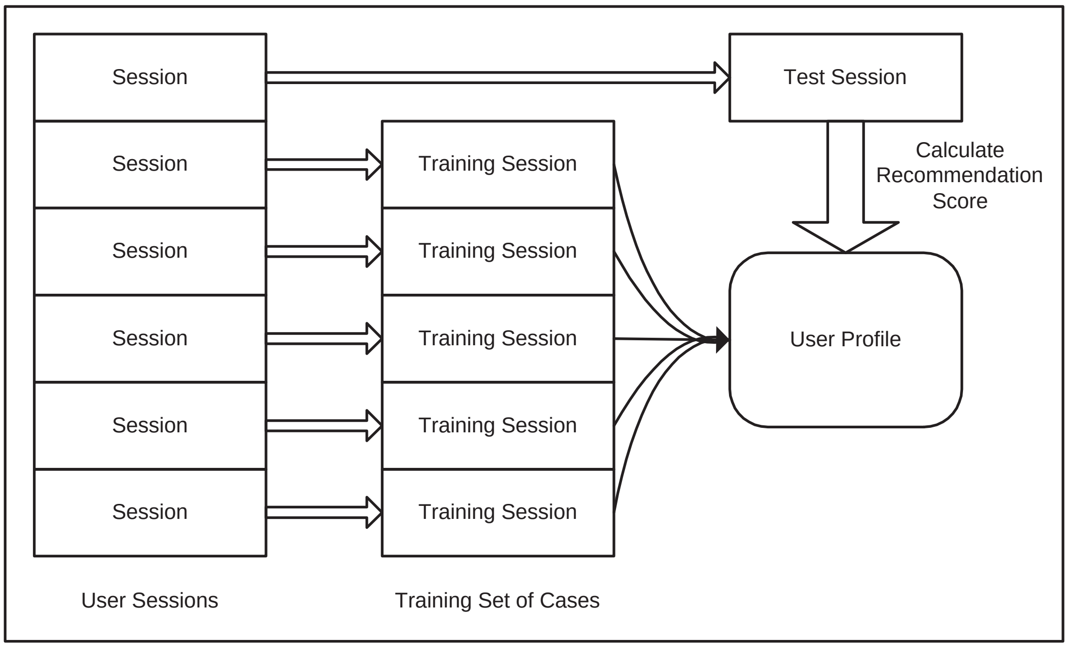


Figure 7.2: Description of our Leave-One-Out Evaluation Framework

This diagram shows how a recommendation strategy is scored using a leave-one-out evaluation framework. The test session is removed from the set of user sessions and a user-model is generated using inferences made in the remaining sessions (the training data). A recommendation ordering is then made on the offers from the test session and a recommendation accuracy is calculated from this ordering. A session recommendation accuracy is generated for each session in this way. A user's recommendation accuracy is the average of that user's individual session recommendation accuracies.

The accuracies of our recommender strategies are shown in a series of graphs showing each user's individual recommendation accuracy as well as an overall recommendation accuracy for that strategy (the average of the user accuracies). Individual user names are not used, rather the users are described as 'user N', where N is simply a number that identifies them. These user identifiers are constant across the evaluations and so recommendation accuracies for individual users can be compared directly.

7.1.1 Statistics Used in the Evaluations

In order to assess the usefulness of our recommendation strategies we need to compare them with a baseline recommendation strategy and with each other. We statistically analyse the user recommendation scores to perform these comparisons. In order to determine the appropriate statistical methods to use to do this it is important to determine the properties of the data we are testing. There are two types of statistical tests we can use: parametric and non-parametric. Parametric tests depend on certain assumptions regarding the underlying data, namely, that the dependent variable (i.e. the recommendation accuracy) is normally distributed and that the observations are independent. Non-parametric tests have less rigorous assumptions concerning the underlying data but are less powerful than parametric ones.

We make the assumption that recommendation scores can be approximated as being normally distributed because they are the result of a large number of small effects acting additively. Dietterich (1998) makes the point that when comparing the results of test folds in *k-fold cross validation* the independence relation is not satisfied since there are overlaps in the training folds. Although our leave-one-out approach has the same properties (i.e. overlapping training data), these overlaps are only on a user-basis. This is important since we are not comparing individual leave-one-out (or fold) accuracies but the mean resulting accuracies. These accuracies have no such overlaps and so satisfy the independence criterion. With these assumptions we are free to use parametric statistical tests to evaluate our recommendation strategies.

We use statistics based on *Student's t-distribution* (Gosset, 1908) to assess the utility of our recommendation strategies. This distribution is useful in estimating the mean of a normally distributed population when the sample size is small. We perform two types of evaluation: comparing a strategy with random recommendation and comparing one strategy against another. The statistics behind these evaluation types are described in the remainder of this section. These tests use a number of variables. For each of our evaluations we provide a table of the variable values (e.g. the average recommendation accuracy of a strategy) that we used to generate our hypotheses. These tables also contain confidence levels of support for each hypothesis.

Strategy A Versus *Random*

The first type of evaluation we perform is the comparison of some of our recommendation strategies with random recommendation. Random recommendation is based on the principle that offer-sets are ordered randomly during recommendation. Random recommendation would be expected to yield a recommendation accuracy of 50%. We perform a *Student t-test* to test the null hypothesis that our recommendation strategies are under-performing random recommendation:

$$t = \frac{(\overline{x} - \mu)\sqrt{n}}{s}$$

where $\overline{x}$ is the overall average recommendation accuracy of Strategy A, μ is the hypothesised mean (50% — the random recommendation accuracy), n is the number of users (19), and s is the standard deviation of the user accuracies of Strategy A. This equation gives us a t value for our null hypothesis. By using this t-value and the number of degrees of freedom we can test whether our null hypothesis is above a certain confidence level. We use the 99% confidence level to determine whether a strategy is better than Random recommendation.

Strategy A Versus *Strategy B*

The second type of evaluation compares two of our recommendation strategies against each other. Because the underlying data are the same in both sets of results we can say that the accuracies are *paired*. In other words, we can compare the accuracies on a user-by-user basis. We use a *Student's paired t-test* to compare our recommendation strategies. This is a parametric test that looks at the *differences* between the recommendation accuracies between the first and second strategies. Let x_i and y_i be the recommendation accuracies for `user i` using strategies A and B respectively. We use the following equation to calculate a value for the paired t:

$$t = \frac{(\overline{x} - \overline{y})\sqrt{n}}{s}$$

where $\overline{x}$ is the overall average accuracy of strategy A and $\overline{y}$ is the overall average accuracy of strategy B, n is the number of users (19) and s is the standard deviation of the differences in accuracy between strategy A and strategy B, i.e.

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - y_i)^2}{n-1}}$$

For the paired t-test the null hypothesis supposes that the difference between the two samples is 0. If there is enough doubt in the null hypothesis we can reject it and say that one strategy is better than the other.

7.2 Analysis of Offer-Based Recommendation

The first test we performed was an evaluation of our original recommender engine; which used the offer-based recommender algorithm. This engine was described in Section 5.1 and in Coyle et al. (2002). Offer-based recommendation proceeds as follows: As sessions are completed, the recommender generates a rating score for each offer in the offer-set based on its similarity to the selected offer. These are stored as cases in the user's offer case-base; where the offer represented the problem component and the rating score represented the solution. When making recommendations on a new offer-set the PTA takes each offer and finds the k most similar offers from the user's offer case-base. The PTA averages the rating scores attached to those k most similar offers to generate a recommendation score for each offer in the test offer-set. When all the target offers have been scored in this way an ordering is calculated across the offer-set. This ordering is used to calculate a session accuracy. These session accuracies are then used to calculate a user accuracy.

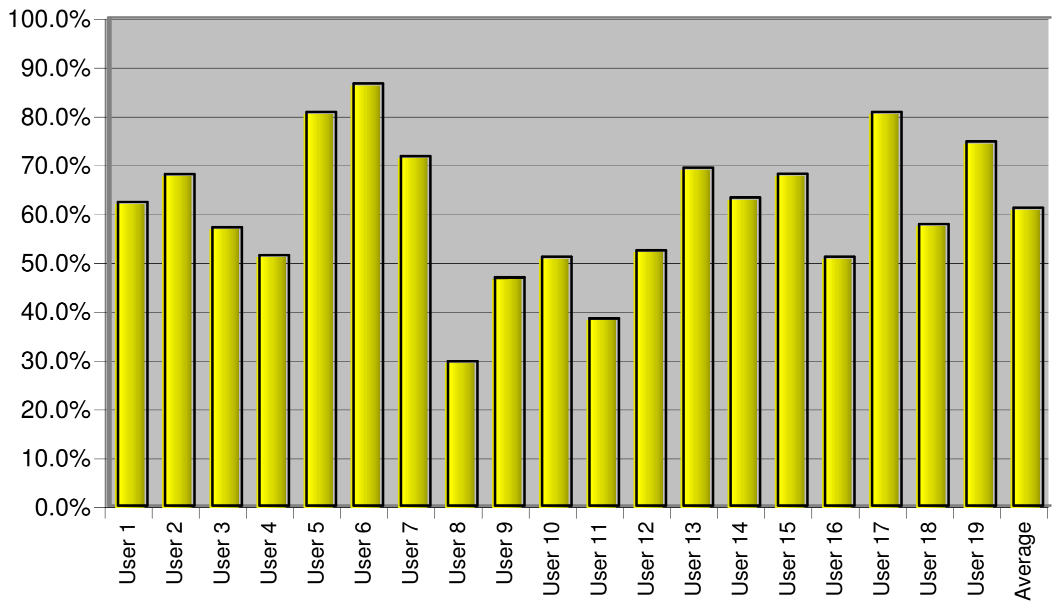


Figure 7.3: Offer-Based Recommendation Accuracy

The evaluation proceeded as follows: We removed the test session from the user's history and generated ratings scores for the offers in the remaining sessions (the training sessions). These offer-cases comprised the user-model for this test session. The offers from the test session are then ordered and the session accuracy is calculated. Each of the sessions are tested in this way and the accuracies are averaged to generate a user accuracy. Figure 7.3 shows the recommendation accuracies for each user of the system using the offer-based recommendation engine. The average recommendation accuracy of all users is shown in the right-most column (61%). We used 7 as the value for k in this evaluation as it combined a high average accuracy with a low variance in user accuracy values (some higher values of k yielded slightly higher overall accuracies).

In order to assess the usefulness of this recommendation strategy these results were compared with random recommendation. Although most of the users have accuracies above 50% there are three who do not. Nevertheless, the average recommendation accuracy of all users is well above 50%. Table 7.1 shows the statistics used in the Student's t distribution calculation. The calculated Student's t ($t = 3.23$) was greater than the 99 Percentile Student's t Distribution for 18 degrees of freedom ($t_{.99} = 2.55$). We can thus reject the null hypothesis and say that offer-based recommendation outperforms random recommendation with a confidence of greater than 99%. This implies that there is value to our offer-recommendation strategy.

Table 7.1: Summary Statistics for Offer-Based Recommendation

Statistic	Value
Number Of Samples (n)	19
Degrees Of Freedom (d)	18
Sample Mean ($\overline{x}$)	61.0%
Sample Standard Deviation (s)	14.9%
Null Hypothesis	$\overline{x} = 50\%$
Student's t (t)	3.23
99 Percentile Student's t Distribution ($t_{.99}$)	2.55

7.3 Analysis of Session-Based Recommendation

The next evaluation we performed tests the accuracy of the session-based recommender, which was described in Section 5.2. The session-based recommender was evaluated in a similar manner to the offer-based strategy. We calculate the recommendation scores for each session individually. First the leave-one-out case-base is created using every request from the user's history except the target session's request. The most similar request-case is retrieved from this case-base and the selected offers from that session are used to drive recommendation in the target session's offer-set. These offers are ordered by comparing their similarities to the target session's selected offer. Recommendation scores are calculated for each session and the overall user recommendation score is the average of that user's session scores. Figure 7.4 shows the recommendation scores achieved with the session-based strategy. The overall recommendation accuracy is shown in the right-most column (62.4%).

In order to assess the usefulness of the session-based recommendation strategy we compared it to random recommendation in much the same way as we did the offer-based strategy. Table 7.2 shows the statistics used in the Student's t-test for comparing the session-based recommender to random recommendation. The calculated Student's t ($t = 3.42$) was greater than the 99 Percentile

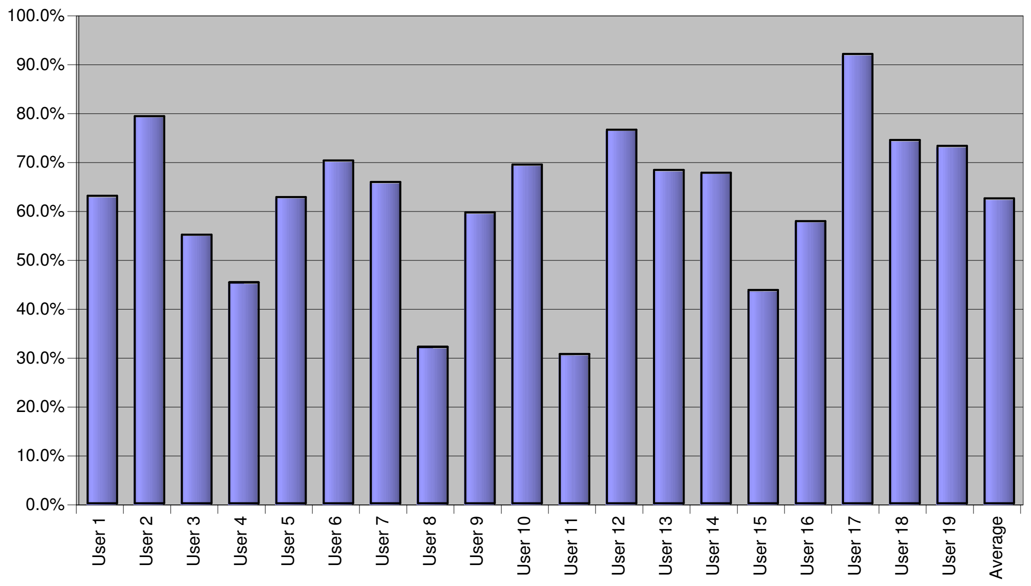


Figure 7.4: Session-Based Recommendation Accuracy

Student's t Distribution for 18 degrees of freedom ($t_{.99} = 2.55$). We can thus reject the null hypothesis and say that session-based recommendation outperforms random recommendation with a confidence of greater than 99%.

Table 7.2: Summary Statistics for Session-Based Recommendation

Statistic	Value
Number Of Samples (n)	19
Degrees Of Freedom (d)	18
Sample Mean ($\overline{x}$)	62.4%
Sample Standard Deviation (s)	15.7%
Null Hypothesis	$\overline{x} = 50\%$
Student's t (t)	3.42
99 Percentile Student's t Distribution ($t_{.99}$)	2.55

Although the recommendation accuracy of session-based recommendation is higher than the offer-based recommendation accuracy (62.4% versus 61.0%) there is not much statistical support to say that one outperforms the other. Despite this, the fact that this recommender significantly outperforms random recommendation suggests that there is a value to using context while making recommendations. In the next sections we show further improvements in recommendation accuracy using techniques that build on this recommender. These improvements involve the incorporation of feature weight learning algorithms and collaborative recommendation components (described in Section 5.4). The session-based results described here and shown in Figure 7.4 will be used as a baseline by which our further recommendation strategies are compared.

7.4 Analysis of Feature Weight Learning

This section describes our evaluation of the application of feature weight learning techniques to the session-based recommendation process. These techniques were described in Section 5.3.1 and Coyle & Cunningham (2003; 2004). We apply these techniques to the offer-recommendation phase of the recommendation process (i.e. to the feature weights of the offer similarity profile). The PTA uses a failure driven approach to improve the similarity profile by examining sessions where the recommendation accuracy was less than 100% and altering the feature weights in order to try to reduce the error. This works by iteratively revaluing the feature weights by incrementing the weights of matching features and decrementing the weights of un-matching features.

Figure 7.5 shows the a posteriori recommendation accuracy, i.e. the average recommendation accuracies that would have been achieved had the learned feature weights for each session been known in advance. They are all close to 100% and show that out feature weight learning algorithm is efficient at learning a set of feature weights that *fit* each user's selections. The fact that most of these accuracies were not actually 100% shows that there are sessions where an optimal set of weights could not be learned. This implies that either user selections are not completely consistent or there is a lacking in the session-based recommendation process. Either of these scenarios are possible, but since the lowest a-posteriori accuracy is so high (95%) we are not concerned.

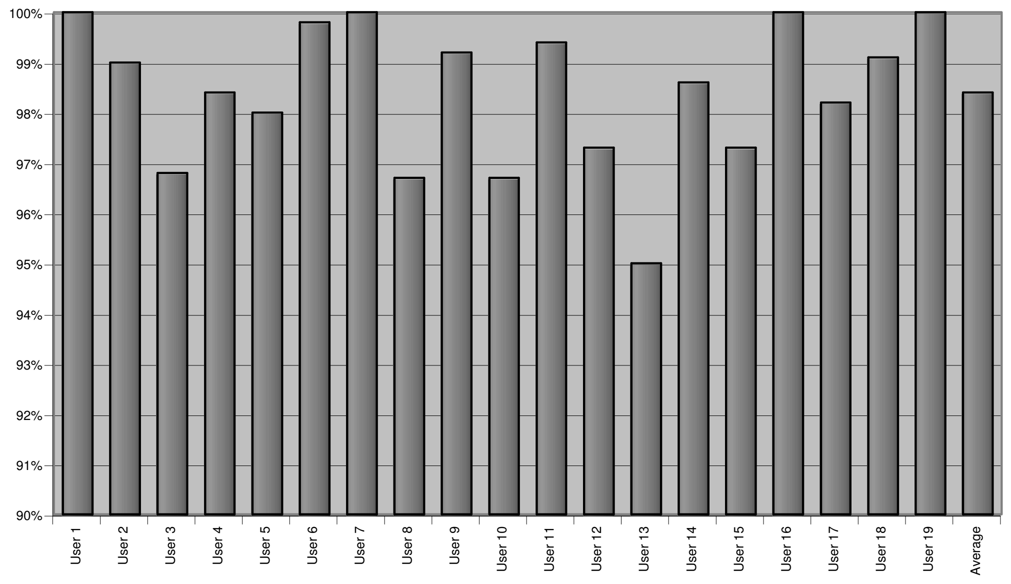


Figure 7.5: A Posteriori Improvement in Accuracy with Learned Feature Weights

We tested different ways of applying these learned weights in the recommendation process. The remainder of this section describes our evaluations of these approaches. It should be noted that both of these evaluations are based on this feature weight learning algorithm.

7.4.1 Analysis of Session-Weights

Our first approach to applying these learned feature weights was to view them as a part of the context of a session, that is to say that similar user-preferences (at the feature weight level) would be observed in sessions with similar contexts. We hoped that by applying the feature weights learned in a similar session we would improve accuracy when recommending offers in a new session.

The evaluation proceeds as follows: we use the same leave-one-out approach as described in Section 7.3 except for one detail. When we retrieve the test session's most similar request-case from the user-model we also retrieve that session's learned feature weights. We use those weights as the basis for the offer similarity measure. These session-weights were learned using a strict leave-one-out process; data from the test session is not used in the feature weight learning process. To this end, feature weights are only learned from data in the leave-one-out case-base.

We plot the accuracies of session-based recommendation with applied session-weights against our baseline accuracy for each user of the system in Figure 7.6. The average session-weights recommendation accuracy of all users is shown in the right-most column (67.0%). The overall improvement of the session-based recommendation strategy against the baseline session-based recommendation strategy is 4.6% (only four users showed declines in accuracy).

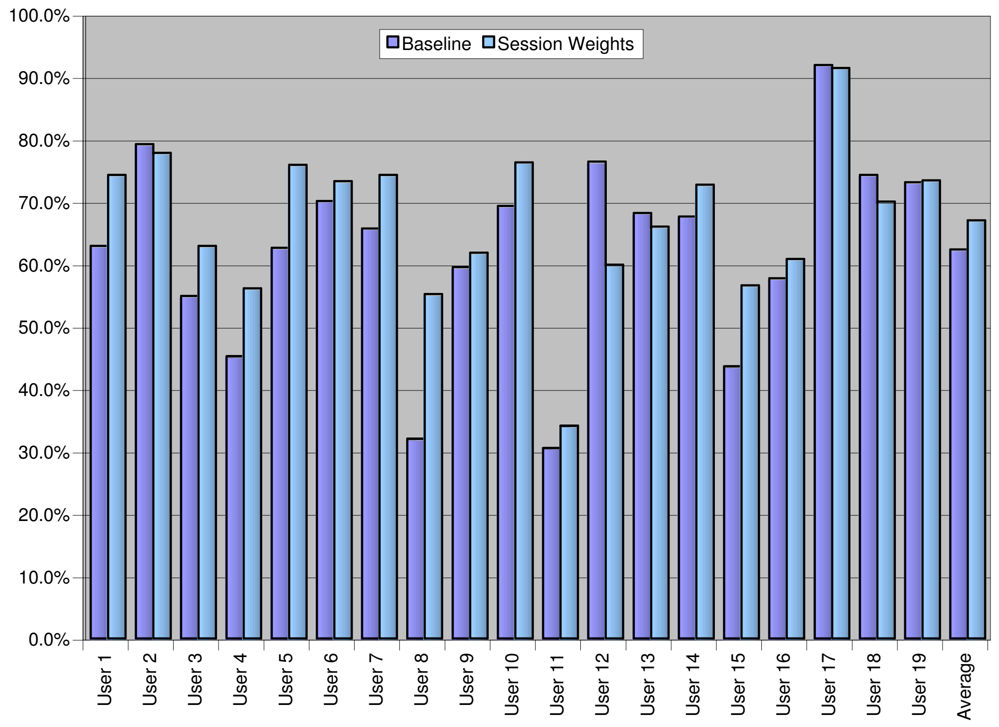


Figure 7.6: Session-Based Recommendation Accuracy with Offer-Weights

We used a paired t-test to determine the confidence level for this improvement over the baseline.

Table 7.3 shows the statistics used in this test. We calculated a Student's t of -2.40. This is less than the 95 Percentile Student's t Distribution for 18 degrees of freedom ($t_{.95} = -2.1$). We can thus reject the null hypothesis and say that the application of learned feature-weights to session-based recommendation improves recommendation accuracy with a confidence of greater than 95%. This implies that there is a significant benefit to learning feature weights and that a user's preferences will be similar to those observed in sessions with a similar context.

Table 7.3: Summary Statistics for Session-Weight Recommendation

Statistic	Value
Number Of Samples (n)	19
Degrees of Freedom (d)	18
Sample Mean of Strategy A ($\overline{x}$)	62.4
Sample Mean of Strategy B ($\overline{y}$)	67.0
Standard Deviation of Differences (s)	8.4%
Null Hypothesis	$\overline{y} < \overline{x}$
Paired Student's t (t)	-2.40
95 Percentile Student's t Distribution ($t_{.95}$)	-2.1

7.4.2 Analysis of User-Weights

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 (Kohavi et al., 1997). 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. One way of minimising over-fitting in this domain is to generalise the session-weights. To do this we have combined the session-weights into a set of *user-weights*. This section describes our evaluation of this strategy.

User-weights are calculated as the *average of the learned weights* from each of that user's sessions. This means that they are an average of all feature weights that were altered by the session-weights learning process; i.e. if the feature `price` was altered upwards in two offer-sets (of twelve) from 1 to 1.4 and 1.2 respectively the user-weight for the feature `price` would be set to 1.3.

A comparison of the recommendation accuracies in the previous evaluation (i.e. the baseline session-based recommendation and session-weight recommendation accuracies) against user-weights is shown in Figure 7.7. The average user-weights recommendation accuracy of all users is shown in the right-most column (73.1%). This is an improvement of 6.1% over the session-weights result and 10.7% over the baseline result.

Our user-weights strategy underperformed our session-weights strategy for four users — although none by more than 5%. Most users showed improvements — one by 21.6%. We see the

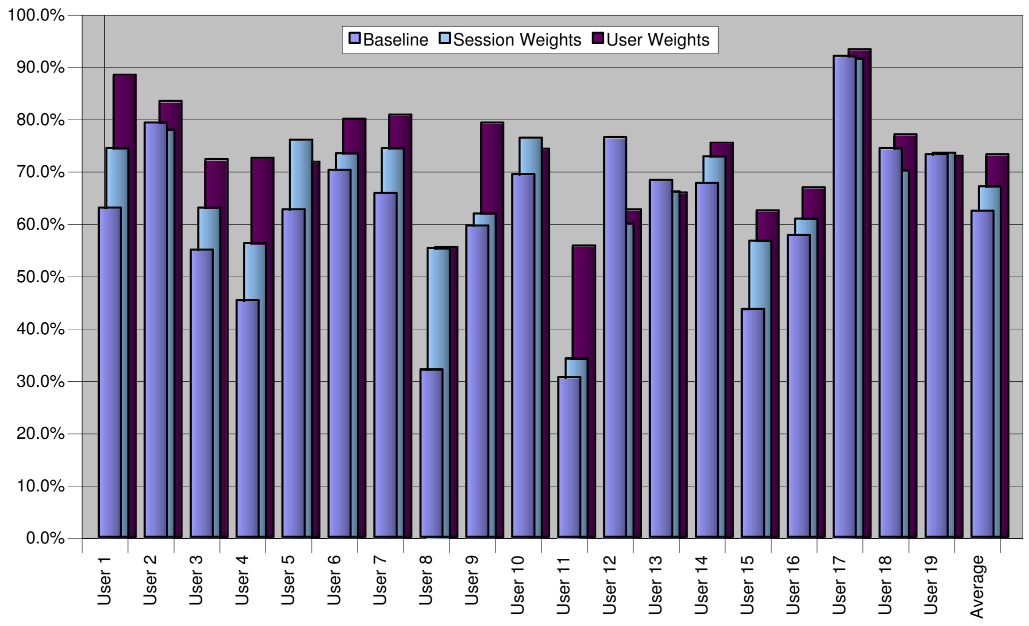


Figure 7.7: Session-Based Recommendation Accuracy with User-Weights

improvement of user-weights recommendation over session-weights recommendation as a validation of our assertion that learning session-weights is susceptible to over-fitting. The amalgamation of session-weights into user-weights reduces over-fitting and leads to improved recommendation accuracy. Table 7.4 shows the statistics used in the paired t-test for comparing the user-weights recommendation accuracy to the session-weights recommendation accuracy. We calculated a Student's t of -3.80. This is less than the 99 Percentile Student's t Distribution for 18 degrees of freedom ($t_{.99} = -2.87$). We can thus reject the null hypothesis and say that user-weights recommendation outperforms session-weights recommendation with a confidence of greater than 99%. Although the statistics are not shown, we can also assert that user-weights recommendation strategy outperformed our baseline session-based recommendation strategy with more than 99.9% confidence.

Table 7.4: Summary Statistics for Session-Weight Recommendation

Statistic	Value
Number Of Samples (n)	19
Degrees of Freedom (d)	18
Sample Mean of Strategy A ($\overline{x}$)	67.0
Sample Mean of Strategy B ($\overline{y}$)	73.1
Standard Deviation of Differences (s)	7.0%
Null Hypothesis	$\overline{y} < \overline{x}$
Paired Student's t (t)	-3.80
99 Percentile Student's t Distribution ($t_{.99}$)	-2.87

7.5 Analysis of Collaborative Recommendation

In Section 5.4 we discussed our collaborative recommendation strategies. In this section we describe evaluations that assess the usefulness of these strategies in the PTA's recommendation process:

Collaboration for Bootstrapping We evaluate our strategy to providing bootstrap recommendations for new users by allowing them access to the combined case-base of every user except their own.

Collaboration for Mature Users We evaluate our integrated collaborative strategy against the user-weights recommendation strategy. This strategy combines recommendations made from a user's own user-model with those made by collaborating with other user-models.

The results of these evaluations suggest that there are some issues that reduce the utility of collaborative strategies in this application domain. We suggest some factors that could explain these results.

7.5.1 Analysis of the Bootstrap Recommender

We mentioned in Section 5.4.1 that the most obvious application of collaborative CBR in the context of our PTA application is in making bootstrap recommendations to new users. Without a user history the PTA cannot make any inferences on their preferences since their user-model is effectively empty. Our bootstrap recommender has a case-base containing every case of every user of the system except the user seeking a recommendation. We call this the *master case-base*. When a new user makes a request they are granted access to the master case-base to drive recommendation. There is no bias between any user's cases and the recommender treats the master case-base as if it was the user's own personal case-base. We compare this recommender to random recommendation as none of the alternative recommendation strategies are capable of making recommendations with empty user-models.

This evaluation proceeds differently to the other evaluations since there are no training data, i.e. the user-models are empty. We calculate a recommendation accuracy for each session in the user's history using the master case-base. Since the other cases in the user's history are not used in the recommendation process each session is treated as if it was the first recommendation they ever received. Thus the average recommendation accuracy for these sessions is a good estimate of the accuracy for a recommendation to this user, were they interacting with the system for the first time. Figure 7.8 shows a graph of the average bootstrap recommendation accuracies for each users. The Average Recommendation Accuracy of all users is shown in the right-most column (62.7%).

We used Student's t test to compare the accuracy of our bootstrapping recommender to random recommendation. Table 7.5 shows the statistics used in the Student's t distribution calculation. The calculated Student's t ($t = 4.96$) was greater than the 99 Percentile Student's t Distribution for

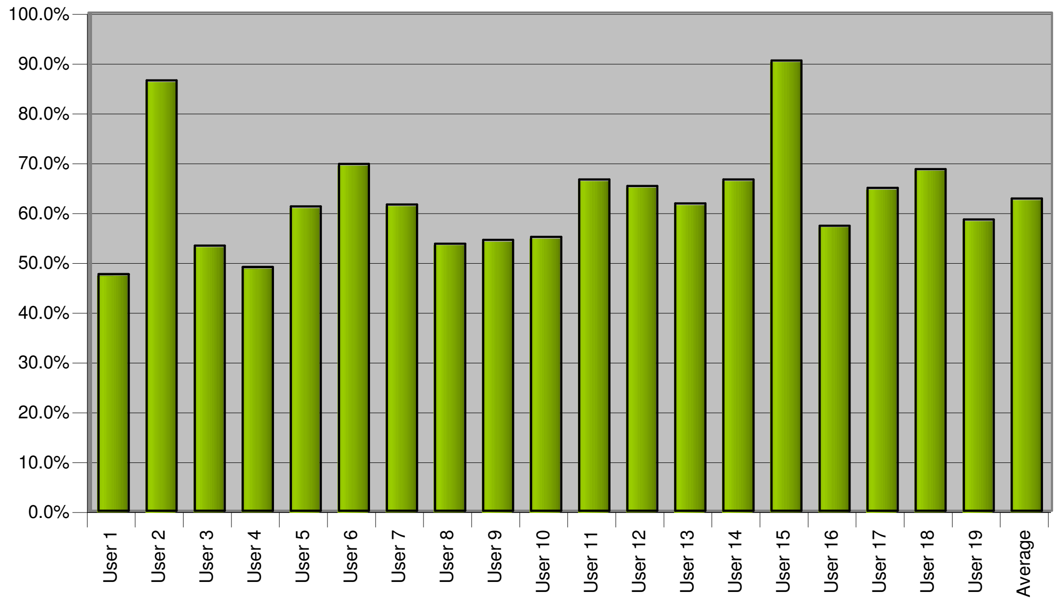


Figure 7.8: Bootstrap Recommendation Accuracy

18 degrees of freedom ($t_{.99} = 2.55$). We can thus reject the null hypothesis and say that bootstrap recommendation outperforms random recommendation with a confidence of greater than 99%. In fact only two users underperformed random recommendation and neither by more than 2.5%. This shows that our collaborative recommendation techniques are useful for making bootstrap recommendations to new users.

Table 7.5: Summary Statistics for Bootstrap Recommendation

Statistic	Value
Number Of Samples (n)	19
Degrees Of Freedom (d)	18
Mean ($\overline{x}$)	62.7%
Standard Deviation (s)	11.1%
Null Hypothesis	$\overline{x} = 50\%$
Student's t (t)	4.96
99 Percentile Student's t Distribution ($t_{.99}$)	2.55

It is interesting to note that the accuracy of our bootstrap recommendation strategy is comparable to the accuracies yielded with our offer-based and session-based recommendation strategies (61.0% and 62.4% respectively). This suggests that it would be better to perform bootstrap recommendation rather than offer-based or session-based recommendation since it comes with the added advantage that it can be used with both new and mature user profiles. However we believe that the bootstrap result is close to the maximum possible accuracy for this technique and that with additional user-data the offer-based and session-based accuracies would improve further.

7.5.2 Analysis of Full Collaborative Recommendation

In Section 5.4 we described our approach to making collaborative recommendations for users with mature user-models. Mature user-models contain data from a number of sessions, and as such would be suitable for generating good recommendations in some scenarios but not necessarily others. The user-model will be used to make recommendations when it is qualified to do so; otherwise collaborative recommendation strategies will be used. Collaborative recommendation involved two steps; the first establishes the confidence of the PTA recommender to use a user-model to make an accurate recommendation and the second involves finding suitable alterative user-models to make a better prediction.

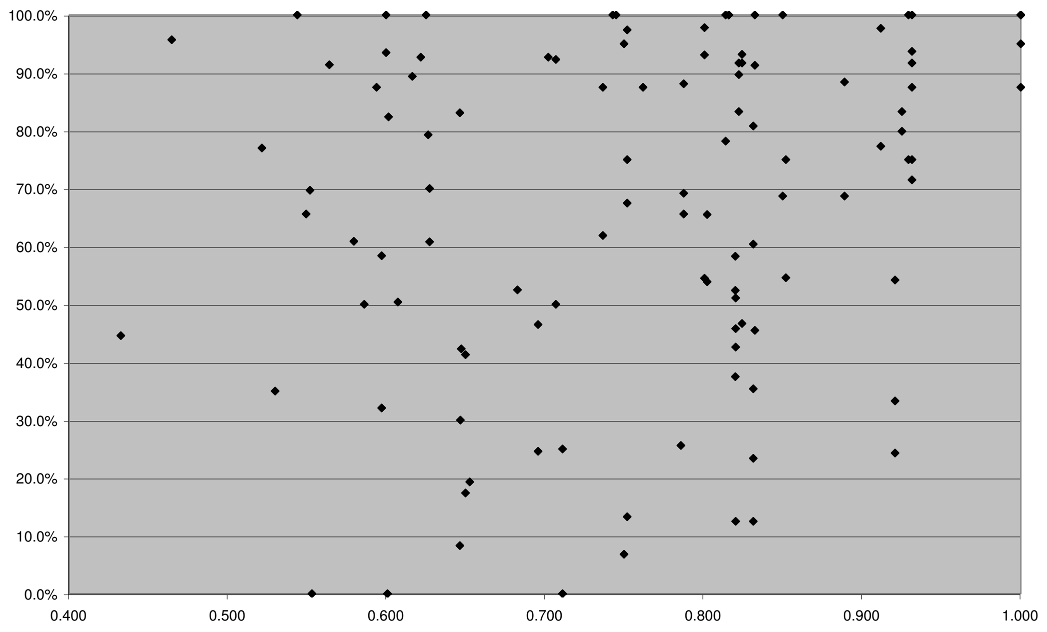


Figure 7.9: Session Competence Versus Recommendation Accuracy

Collaboration is attempted in the context-matching recommendation phase, when the PTA searches the user-model for the most similar previous request that the user made. The collaborator component of the PTA uses the similarity of this request-case to estimate confidence in making an accurate recommendation. Figure 7.9 shows a scatter-plot of competence against accuracy for every session of every user. We would expect this plot to show a linear relationship between competence and accuracy. However, there is no obvious relationship — we calculated a Pearson coefficient[1] of 0.23 between session accuracy and competence. While this value is positive, it is not

[1]The Pearson coefficient is a good measure of the linear relationship between two variables

high enough for us to have much confidence in this measure as a gauge of recommendation success and thus as part of a trigger for collaboration.

We attempted to use other confidence measures to no avail. These included looking at the similarity between the previously selected offer and the current offer-set. None of these measures yielded correlation figures higher than the original confidence measure.

Our collaborative recommendation strategy also depends on two further variables; the competence value above which collaboration will not occur, c and the variable that determines the relative contributions of user competence and user similarity in determining which user-model is used to make a recommendation, ω. We performed a test whereby we varied c and ω and determined the overall recommendation accuracy. The results of this test were not positive. Although there were slight improvements in accuracy at certain combinations of c and ω they were never significant. Figure 7.10 shows the overall optimal result (yielded with $c = 1.5$ and $\omega = 1$). Most other combinations were significantly worse than this result. This result is interesting because the optimal value for ω was one; indicating that competence was not used to produce this result.

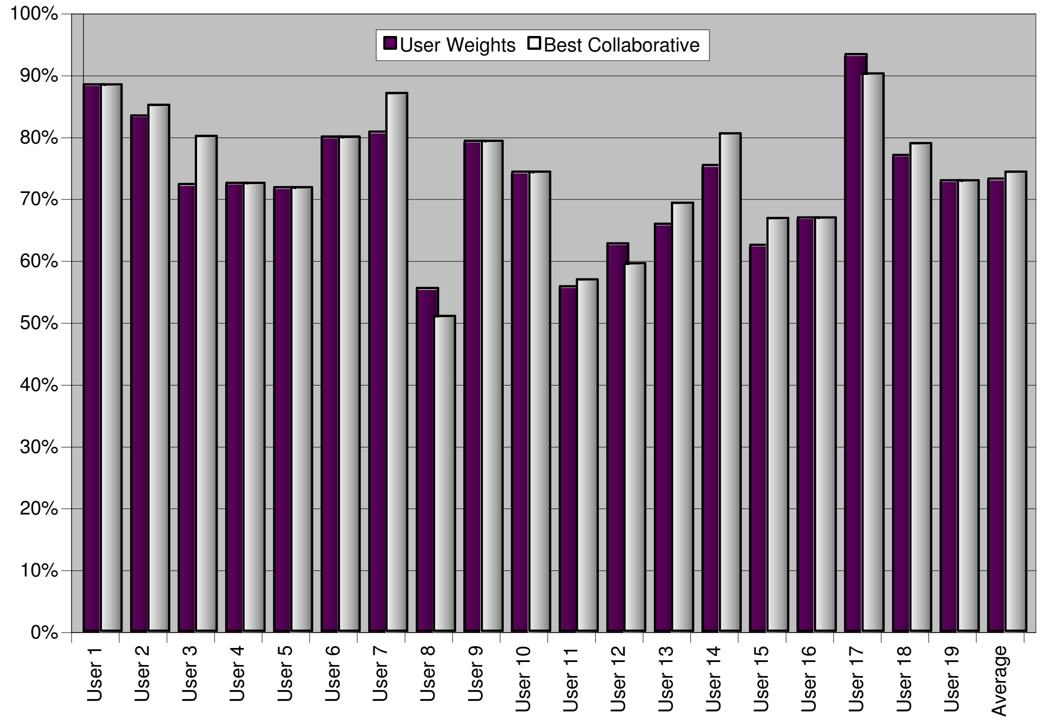


Figure 7.10: Optimal Collaborative Recommendation Accuracy

Although our collaborative strategies are similar to those of Plaza et al. (1997) and McGinty et al. (2001) our collaborative recommender was not successful. Their application domains (classification and recommendation respectively) are dependant on the retrieval component of CBR.

However, our session-based recommendation strategies are heavily dependent on the reuse (or adaptation) phase of CBR, i.e. the offer-recommendation phase. We believe that this difference is what led to a failures to predict recommendation success a priori. If confidence were well correlated with recommendation accuracy our recommendations could have been expected to have been more successful and collaboration would have been successful.

There are also some other factors that may have contributed to failure of the collaborative recommendation evaluation:

Noisy Data Any noise in the data would deteriorate collaborative recommendations. There is a large variance in recommendation accuracies even within individual user histories. We believe that this is contributing to poor collaborative recommendations.

Over-fitting The potential over-fitting problems in feature weight learning would lead to user-weights that do not accurately reflect a user's travel preferences. This in turn would lead to inaccurate user-to-user similarity calculations.

Although these factors may have made collaborative recommendations more difficult, they are not unusual or unexpected issues. Our other recommendation strategies managed to overcome them and we would have expected the PTA's collaborative recommender to also overcome them.

7.6 Conclusions

In this chapter we have evaluated the recommendation strategies that were described in Chapter 5. We began by describing the data, the users of the system, the form of the evaluations and the statistics that were used to evaluate these strategies. The evaluations in this chapter tell a story in some ways. We describe our original recommender and evaluate each step until the final collaborative session-based recommender. These evaluations build on each other and at each step we describe and discuss the change in recommendation accuracy. Figure 7.11 shows the relative accuracies of the each of our recommendation strategies.

Our offer-based and session-based recommenders yielded accuracies of slightly greater than 60% and were significantly better than random recommendations. The session-based recommender used the context of the session to improve recommendation. Although its accuracy was not significantly better than the offer-based recommender, we were able to implement further recommendation strategies that did lead to significant improvements. We implemented a feature weight learning algorithm that attempted to capture more contextual recommendation knowledge. It was our intention that these learned feature weights would be stored with the session itself and would further improve accuracy. We achieved this improvement but were suspicious that over-fitting was occurring in the process. For this reason we used an amalgamation function to counteract this problem by smoothing the local feature weights into an overall set of feature weights — the user-

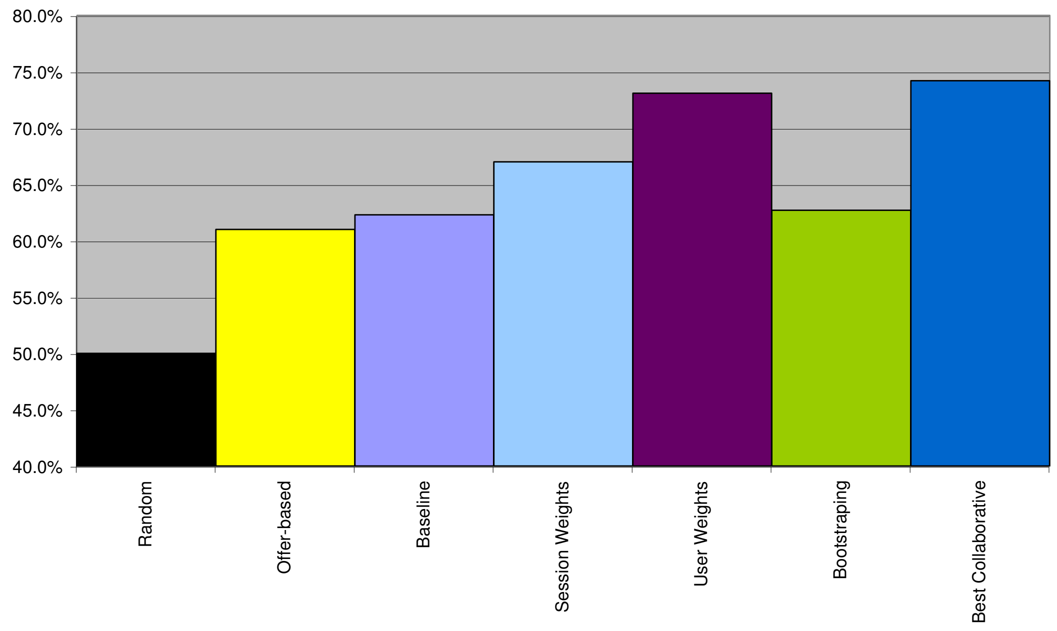


Figure 7.11: Accuracy of the Different Recommendation Strategies

weights. The application of these user-weights to recommendation yielded further improvements in recommendation accuracy — to more than 70%.

Our final component investigated the possibility of using collaborative recommendations to generate further improvements. It was our intention that the PTA could use these strategies in scenarios where it deemed the user-model to be unqualified to making good recommendations. Collaborative recommendation failed because the PTA was unable to successfully gauge when collaboration would be appropriate. When making collaborative recommendations for new users, for whom collaboration is necessary, the PTA successfully made good recommendations. However, when making recommendations for mature users, the PTA failed to improve recommendation accuracy. This failure suggests that the only role for collaboration in this application is in bootstrapping new users. It does not appear to add value as a means of improving accuracy for mature users.

Chapter 8

Conclusions and Future Work

8.1 Overview

This thesis described the Personal Travel Assistant (PTA), an application for assisting a user to search for and find flights. Since these searches usually yield large sets of offers the PTA uses recommendation strategies to ease the burden on the user. The recommendation process is made more difficult by our decision to perform all learning using implicitly gathered feedback alone. This feedback comes from observations of user behaviour and is used to generate models describing their flight-preferences. The PTA uses case-base reasoning techniques to achieve successful recommendations, including context-based recommendation, feature weighting and cooperative recommendation techniques.

The approaches described in this thesis were all developed with the PTA application in mind. Despite this, we feel that our implicit user-modelling, context-based recommendation and similarity learning techniques could successfully be applied to other recommendation domains where it is difficult or inappropriate to gather explicit user feedback. The key to their success is their ability to use the selection of a preferred item from a list of presented items to personalise and improve the recommendation process.

8.2 Implicit User-Modelling

In this thesis we described a number of strategies that we applied to the problem of recommending flights in the PTA application. Each of these strategies depend on a user-model that represents each user's travel preferences. User-modelling has a trade-off between developing a model that accurately represents the user's preferences, while reducing the amount of effort the user has to put into the model's creation and maintenance. The PTA uses only implicit feedback to drive the modelling process. This feedback comes from observations of user selections from sets of presented flights. Using only implicit feedback, we were able to improve recommendation accuracy from 50%

(which we would expect to get if there were no user-model) to more than 70%.

8.3 Context-Based Recommendation Algorithm

In Chapter 5 we described a context-based recommendation algorithm that uses the context in which a set of offers were sought to drive recommendation of that offer-set. This approach assumes that user flight preferences are related to context. The PTA uses features from the user's flight request to gauge the context of a session, e.g. the distance of the flight.

Our context-based recommendation strategy uses two phases to drive recommendation: the first attempted to find the session with the most similar context to the current one from the users history; the second uses knowledge from that session to drive recommendation in the current session. This strategy showed higher accuracy than another non-contextual recommendation strategy we implemented. It also facilitated the integration of more successful feature-weighting and co-operative recommendation components which would have been impossible in our non-contextual recommendation scheme.

8.4 Incorporating Utility into Similarity

Our context-based recommendation strategy uses a similarity measure to order the offer-set. We implemented an efficient introspective feature weight learning technique and used it to incorporate utility (in the form of implicit user-feedback) into this similarity measure. We viewed the feature weights in this similarity measure as being reflective of user preferences — that the higher the weight on a feature, the higher the importance that user placed on that feature. We learned feature weights on a session basis and applied them in two different ways: as local, contextual preferences; and as global user preferences.

The first approach supposed that different travel preferences would be apparent in different contexts. Analysis showed a significant improvement in recommendation accuracy using this approach, which demonstrated that a user's preferences in a given session are similar to those observed in sessions with a similar context. However, feature weighting is characteristically seen as being subject to over-fitting. In order to assess the extent of this problem we amalgamated the individual session weights into a set of overall global user-weights. Our analysis of the application of user-weights in the recommendation process demonstrated a further improvement in recommendation accuracy, to 73.1% when applying personalised user-weights (as opposed to 62.4% without personalised user-weights).

8.5 Cooperative Techniques

Cooperative techniques are useful in recommender systems when a user-model might not have enough information to make an informed recommendation. Cooperative recommenders use knowledge contained in similar users' user-model to make better recommendations. This is especially useful when making recommendations to new users (who have empty or sparse user-models).

We applied the standard cooperative CBR techniques as described by Plaza et al. (1997) and McGinty and Smyth (2001) to making recommendations in the PTA application. These approaches depend on estimating the ability of a user-model to solve a problem a priori and borrowing case data from competent users with similar tastes. Our analysis suggests that these techniques are not suitable for this application domain. The main reason for this failure was that we were unable to reliably estimate the ability of a user-model to make a good recommendations a priori, i.e. the user-model's *competence*.

We believe that this failure to predict accuracy was because the standard approach is applied in CBR systems which derive their power from the retrieval phase of CBR. However, our recommender system is heavily dependent on the reuse (or adaptation) phase of CBR. We believe if competence had been well correlated with recommendation accuracy our cooperative recommendations would have been more accurate. However, despite these failures, we did implement a bootstrap recommender that allowed new users access to the combined case-knowledge of the user base. Analysis of this approach showed significant improvements in recommendations for new users.

8.6 CBR Representation

In Chapter 4 we presented CBML, an XML-based CBR representation language. CBML is ubiquitous in the recommendation components of the PTA application. It separates the domain knowledge, case data and similarity knowledge from the application code and allows them to be stored in ASCII documents. This makes it easy for a developer to alter the CBR processes in an application by changing the CBML documents without needing to rewrite any application code. The portability of case and similarity measure data also makes it easy to implement cooperative and distributed CBR strategies. Because CBML is XML-based, we also have access to the full range of tools created by the XML community, including document parsers, validators, translators, etc.

As a demonstration of the usefulness of CBML we also described work being done by other researchers in a number of other research areas that use CBML to represent their CBR data, including explanation-based CBR, spam filtering and feature selection.

8.7 Future Work

The Personal Travel Assistant

The PTA application was flawed in that it was tied to the protocols we developed for dealing with real-world travel vendors. The problem of information overload would have been more obvious if users were able to make more complicated requests, e.g. 'I would like to fly to Italy for a weekend some time in June.' This scenario would lead to a very large set of offers and would have made the problem of recommending flights much more interesting (and difficult). Even so, this problem only deals with flights; if FIPA's original specification for a travel marketplace containing the full range of travel products were available, the number of possible recommendations would become truly massive. If this marketplace came into existence, a full range of recommendation strategies would be needed to assist a user through the search space.

We would hope that in a mature recommendation system, the strategies described in this work could operate in combination with a larger group of other recommendation and information-visualisation strategies such as those based on diversity boosting (as described in Section 3.3), comparison-based recommendation (McGinty and Smyth, 2002) and critiquing (McSherry, 2004; Reilly, 2004)).

Learning Similarity

In Section 5.3.4 a technique for learning local similarity measures was introduced. This technique was used in a limited manner in the PTA application (it was restricted to learning a single feature type — the `taxonomy` feature type). However, due to the lack of available user-data and the propensity of these techniques to suffer from over-fitting they were not formally evaluated in this work. It would be interesting to run evaluations of these techniques on a more densely populated data-set to assess their effectiveness.

CBML

The current implementation of CBML can represent three of Richter's four knowledge containers — the case base, the similarity measure and the domain vocabulary. It is clear that if CBML is to develop further, the next step should be the development of a representation format for the fourth container — adaptation knowledge.

Bibliography

Aamodt, A. and Plaza, E.: 1994, *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches.*, *AI Commun.* **7(1)**, 39–59

Aha, D. W.: 1997, *Editorial.*, *Artificial Intelligence Review* **11(1-5)**, 7–10

Allen, J. F.: 1999, *Mixed-Iniative interaction*, *IEEE Intelligent Systems* **14(6)**, 377–383

Armengol, E., Plaza, E., and Ontañón, S.: 2004, *Explaining Similarity in CBR*, in P. Cunningham and D. McSherry (eds.), *ECCBR 2004 Workshop Proceedings*, pp 155–164

Ball, C.: 2003, *Screen-scraping with WWW::Mechanize*, Available online at `http://www.perl.com/pub/a/2003/01/22/mechanize.html`

Bergmann, R., Richter, M. M., Schmitt, S., Stahl, A., and Vollrath, I.: 2001, *Utility-Oriented Matching: A New Research Direction for Case-Based Reasoning*, in *Proceedings of the 9th German Workshop on Case-Based Reasoning, GWCBR'01, Baden-Baden, Germany*, pp 264–274

Bezos, J.: 2000, Quote taken from an interview with CIO Magazine, Aug 2000

Bonzano, A.: 1998, *ISAC: a Case-Based Reasoning System for Aircraft Conflict Resolution*, *Ph.D. thesis*, Trinity College Dublin.

Bonzano, A., Cunningham, P., and Smyth, B.: 1997, *Using Introspective Learning to Improve Retrieval in CBR: A Case Study in Air Traffic Control.*, in D. B. Leake and E. Plaza (eds.), *Case-Based Reasoning Research and Development, Second International Conference, ICCBR-97, Providence, Rhode Island, USA, July 25-27, 1997, Proceedings*, pp 291–302, Springer

Bradley, K., Rafter, R., and Smyth, B.: 2000, *Case-Based User Profiling for Content Personalisation.*, in P. Brusilovsky, O. Stock, and C. Strapparava (eds.), *Adaptive Hypermedia and Adaptive Web-Based Systems, International Conference, AH 2000, Trento, Italy, August 28-30, 2000, Proceedings*, pp 62–72, Springer

Bradley, K. and Smyth, B.: 2001, *Improving Recommendation Diversity*, in D. O'Donoghue (ed.), *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science (AICS'2001)*

Branting, K.: 2001, *Acquiring Customer Preferences from Return-Set Selections.*, in D. W. Aha and I. Watson (eds.), *Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, July 30 - August 2,* *2001, Proceedings*, pp 59–73, Springer

Burke, R. D.: 1999, *The Wasabi Personal Shopper: A Case-Based Recommender System.*, in *AAAI/IAAI*, pp 844–849

Chen, I.-C.: 2003a, *American Airlines, FareChase settle suit*, From the Dallas Business Journal, available online at `http://dallas.bizjournals.com/dallas/stories/2003/06/09/daily55.html`

Chen, I.-C.: 2003b, *American Airlines wins injunction against FareChase*, From the Dallas Business Journal, available online at `http://dallas.bizjournals.com/dallas/stories/2003/03/10/daily50.html`

Coyle, L. and Cunningham, P.: 2003, *Exploiting Re-ranking Information in a Case-Based Personal Travel Assistant*, in D. Aha (ed.), *Procs. of the Workshop in Mixed-Initiative Case-Based Reasoning, Workshop Programme at the Fifth International Conference on Case-Based Reasoning*, pp 11–20

Coyle, L. and Cunningham, P.: 2004, *Improving Recommendation Ranking by Learning Personal Feature Weights*, in P. A. G. Calero and P. Funk (eds.), *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 Madrid, Spain, August 30th through Sep 2nd, 2004, Proceedings*, pp 560–572, Springer

Coyle, L., Cunningham, P., and Hayes, C.: 2002, *A Case-Based Personal Travel Assistant for Elaborating User Requirements and Assessing Offers*, in S. Craw and A. D. Preece (eds.), *Advances in Case-Based Reasoning, 6th European Conference, ECCBR 2002 Aberdeen, Scotland, UK, September 4-7, 2002, Proceedings*, pp 505–518, Springer

Coyle, L., Doyle, D., and Cunningham, P.: 2004, *Representing Similarity for CBR in XML*, in P. A. G. Calero and P. Funk (eds.), *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 Madrid, Spain, August 30th through Sep 2nd, 2004, Proceedings*, pp 119–127, Springer

Coyle, L., Hayes, C., and Cunningham, P.: 2003, *Representing Cases for CBR in XML*, *Expert Update* **6(2)**, 7–13

Craw, S., Wiratunga, N., and Rowe, R.: 1998, *Case-Based Design for Tablet Formulation.*, in B. Smyth and P. Cunningham (eds.), *Advances in Case-Based Reasoning, 4th European Workshop, EWCBR-98, Dublin, Ireland, September 1998, Proceedings*, pp 358–369, Springer

Cunningham, P., Doyle, D., and Loughrey, J.: 2003a, *An Evaluation of the Usefulness of Case-Based Explanation*, in K. D. Ashley and D. G. Bridge (eds.), *Case-Based Reasoning Research and Development, 5th International Conference on Case-Based Reasoning, ICCBR 2003, Trondheim, Norway, June 23-26, 2003, Proceedings*, pp 122–130, Springer

Cunningham, P., Nowlan, N., Delany, S. J., and Haahr, M.: 2003b, *A Case-based Approach to Spam Filtering that can Track Concept Drift*, in *In The ICCBR'03 Workshop on Long-Lived CBR Systems, Trondheim, Norway, June 2003. Available: http://www.cs.tcd.ie/publications/tech-*

reports/reports.03/TCD-CS-2003-16.pdf

Delany, S. J. and Cunningham, P.: 2004, *An Analysis of Case-Base Editing in a Spam Filtering System*, in P. A. G. Calero and P. Funk (eds.), *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 Madrid, Spain, August 30th through Sep 2nd, 2004, Proceedings*, pp 128–141, Springer

Delany, S. J., Cunningham, P., and Coyle, L.: 2004a, *An Assessment of Case-Based Reasoning for Spam Filtering*, in L. McGinty and B. Crean (eds.), *Proceedings of the Fifteenth Irish Conference on Artificial Intelligence and Cognitive Science (AICS'2004)*, pp 9–18

Delany, S. J., Cunningham, P., Tsymbal, A., and Coyle, L.: 2004b, *A Case-Based Technique for Tracking Concept Drift in Spam Filtering*, To Appear in Procs. of the twenty-fourth Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (AI-2004)

Dietterich, T. G.: 1998, *Approximate Statistical Test For Comparing Supervised Classification Learning Algorithms*, *Neural Computation* **10(7)**, 1895–1923

Doyle, D., Cunningham, P., Bridge, D., and Rahman, Y.: 2004a, *Explanation Orientated Retrieval*, in P. A. G. Calero and P. Funk (eds.), *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 Madrid, Spain, August 30th through Sep 2nd, 2004, Proceedings*, pp 157–168, Springer

Doyle, D., Loughrey, J., Nugent, C., Coyle, L., and Cunningham, P.: 2004b, *FIONN: A Framework for Developing CBR Systems*, To Appear in Expert Update

Fesenmaier, D. R., Ricci, F., Schaumlechner, E., Wóber, K., and Zanella, C.: 2003, *DIETORECS: Travel advisory for multiple decision styles*, in *Proceedings of Enter conference, Helsinki, Finland, January 29 - 31, 2003.*, pp 232–241

FIPA: 1997a, *FIPA 97 Specification Part 2: Agent Communication Language*, Version 2.0

FIPA: 1997b, *FIPA 97 Specification Part 4: Personal Travel Assistance*, Version 1.0

Fisher, R. A.: 1936, *The Use of Multiple Measurements in Taxonomic Problems*, in *The Annals of Eugenics 7*, pp 109–122

Fox, S. and Leake, D. B.: 1995, *Using Introspective Reasoning to Refine Indexing.*, in *IJCAI*, pp 391–399

Gabel, T. and Stahl, A.: 2004, *Exploiting Background Knowledge when Learning Similarity Measures*, in P. Funk and P. A. G. Calero (eds.), *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 Madrid, Spain, August/September 2004, Proceedings*, pp 169–183, Springer

Gosset, W. S.: 1908, *On the Probable Error of a Mean.*, *Biometrika* 6(1)

Hayes, C. and Cunningham, P.: 1998, *Distributed CBR using XML*, in W. Wilke and J. Schumacher (eds.), *Proceedings of the KI-98 Workshop on Intelligent Systems and Electronic Commerce*

Hayes, C. and Cunningham, P.: 1999, *Shaping a CBR View with XML*, in K.-D. Althoff, R.

Bergmann, and L. K. Branting (eds.), *Case-Based Reasoning Research and Development: Third International Conference on Case-Based Reasoning, ICCBR-99, Seeon Monastery, Germany, July 1999. Proceedings*, pp 468–479, Springer-Verlag Heidelberg

Hayes, C., Massa, P., Avesani, P., and Cunningham, P.: 2002, *An On-line Evaluation Framework for Recommender Systems*, in *In Workshop on Recommendation and Personalization Systems, AH 2002, Malaga, Spain, 2002.*, Springer Verlag.

Jarmulak, J., Craw, S., and Crowe, R.: 2000, *Genetic Algorithms to Optimise CBR Retrieval.*, in E. Blanzieri and L. Portinale (eds.), *Advances in Case-Based Reasoning, 5th European Workshop, EWCBR 2000, Trento, Italy, September 6-9, 2000, Proceedings*, pp 136–147, Springer

Kitano, H. and Shimazu, H.: 1996, *The Experience Sharing Architecture: A Case Study in Corporate-Wide Case-Based Software Quality Control*, pp 235–268, AAAI Press

Kohavi, R., Langley, P., and Yun, Y.: 1997, *The Utility of Feature Weighting in Nearest-Neighbor Algorithms*, in W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (eds.), *9th European Conference on Machine Learning ECML-97, Prague, Czech Republic*, pp 213–220

Kolodner, J. L. (ed.): 1993, *Case-Based Reasoning*, Morgan Kaufmann

Leake, D. B., Kinley, A., and Wilson, D. C.: 1995, *Learning to Improve Case Adaption by Introspective Reasoning and CBR.*, in M. M. Veloso and A. Aamodt (eds.), *Case-Based Reasoning Research and Development, First International Conference, ICCBR-95, Sesimbra, Portugal, October 23-26, 1995, Proceedings*, pp 229–240, Springer

Lenz, M., Auriol, E., and Manago, M.: 1998, *Diagnosis and Decision Support*, pp 51–90, Springer

Loughrey, J. and Cunningham, P.: 2004, *Overfitting in Wrapper-Based Feature Subset Selection: The Harder You Try the Worse it Gets*, To Appear in AI-2004

Manago, M., Bergmann, R., Conruyt, N., Traphöner, R., Pasley, J., Renard, J. L., Maurer, F., Wess, S., Althoff, K.-D., and Dumont, S.: 1994, *CASUEL: A Common Case Representation Language*

McGinty, L. and Smyth, B.: 2000a, *Personalized Route Planning: A Case-Based Approach.*, in E. Blanzieri and L. Portinale (eds.), *Advances in Case-Based Reasoning, 5th European Workshop, EWCBR 2000, Trento, Italy, September 6-9, 2000, Proceedings*, pp 431–442, Springer

McGinty, L. and Smyth, B.: 2000b, *TURAS: A Personalised Route Planning System.*, in *PRICAI*, p. 791

McGinty, L. and Smyth, B.: 2001, *Collaborative Case-Based Reasoning: Applications in Personalised Route Planning.*, in D. W. Aha and I. Watson (eds.), *Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, July 30 - August 2, 2001, Proceedings*, pp 362–376, Springer

McGinty, L. and Smyth, B.: 2002, *Comparison-Based Recommendation.*, in S. Craw and A. D. Preece (eds.), *Advances in Case-Based Reasoning, 6th European Conference, ECCBR 2002 Aberdeen, Scotland, UK, September 4-7, 2002, Proceedings*, pp 575–589, Springer

McSherry, D.: 2004, *Incremental Relaxation of Unsuccessful Queries*, in P. A. G. Calero and P. Funk (eds.), *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 Madrid, Spain, August 30th through Sep 2nd, 2004, Proceedings*, pp 331–345, Springer

Nugent, C. and Cunningham, P.: 2004, *A Case-Based Explanation System for 'Black-Box' Systems*, in P. Cunningham and D. McSherry (eds.), *ECCBR 2004 Workshop Proceedings*, pp 155–164

Núñez-Suárez, J., O'Sullivan, D., Brouchoud, H., Cros, P., and Moore, C.: 2000a, *Personal Travel Market: A real life application of the FIPA Standards.*, in *WTC/ISS'2000: XVII World Telecommunications Congress 7-12 May 2000, Birmingham UK*

Núñez-Suárez, J., O'Sullivan, D., Brouchoud, H., Cros, P., and Moore, C.: 2000b, *Personal Travel Market: I'll book your trip for you sir!*, in *AT2AI'2000: 2nd International Symposium at the 15th EMCSR European Meeting. From Agent Theory to Agent Implementation 25-28 April 2000, Vienna, Austria*

Núñez-Suárez, J., O'Sullivan, D., Brouchoud, H., Cros, P., and Moore, C.: 2000c, *Validating the FIPA Standards Through a Real-World Application*, in *SAC'2000: 2000 ACM Symposium on Applied Computing 19-21 March 2000, Villa Olmo, Como, Italy*

Oehlmann, R., Edwards, P., and Sleeman, D. H.: 1995, *Changing the Viewpoint: Re-Indexing by Introspective Question*, in *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, pp 381–386, Lawrence-Erlbaum and Associates

Ontañón, S. and Plaza, E.: 2001, *Learning When to Collaborate among Learning Agents*, in *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA.*, pp 394–405, Springer-Verlag

Ontañón, S. and Plaza, E.: 2003, *Justification-based Multiagent Learning*, in *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA.*, pp 576–583, AAAI Press

O'Sullivan, D., Núñez-Suárez, J., Brouchoud, H., Cros, P., Moore, C., and Byrne, C.: 2000, *Experiences in the use of FIPA Agent Technologies for the Development of a Personal Travel Application.*, in *Agents*, pp 357–364

Peuret, F.: 1999, Case-based travel agent, *Master's thesis*, Trinity College Dublin

Plaza, E., Arcos, J. L., and Martín, F. J.: 1997, *Cooperative Case-Based Reasoning*, in *Selected papers from the Workshop on Distributed Artificial Intelligence Meets Machine Learning, Learning in Multi-Agent Environments*, pp 180–201, Springer-Verlag

Plaza, E. and Ontañón, S.: 2001, *Ensemble Case-Based Reasoning: Collaboration Policies for Multiagent Cooperative CBR.*, in D. W. Aha and I. Watson (eds.), *Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, July 30 - August 2, 2001, Proceedings*, pp 437–451, Springer

Prasad, M. V. N., Lesser, V. R., and Lander, S. E.: 1996, *Retrieval and Reasoning in Distributed Case Bases*, *Journal of Visual Communication and Image Representation, Special Issue on*

Digital Libraries **7(1)**, 74–87

Reilly, J.: 2004, *Dynamic Critiquing*, in P. A. G. Calero and P. Funk (eds.), *Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 Madrid, Spain, August 30th through Sep 2nd, 2004, Proceedings*, pp 763–777, Springer

Ricci, F., Arslan, B., Mirzadeh, N., and Venturini, A.: 2002, *ITR: A Case-Based Travel Advisory System.*, in S. Craw and A. D. Preece (eds.), *Advances in Case-Based Reasoning, 6th European Conference, ECCBR 2002 Aberdeen, Scotland, UK, September 4-7, 2002, Proceedings*, pp 613–627, Springer

Ricci, F., Venturini, A., Cavada, D., Mirzadeh, N., Blaas, D., and Nones, M.: 2003, *Product Recommendation with Interactive Query Management and Twofold Similarity.*, in K. D. Ashley and D. G. Bridge (eds.), *Case-Based Reasoning Research and Development, 5th International Conference on Case-Based Reasoning, ICCBR 2003, Trondheim, Norway, June 23-26, 2003, Proceedings*, pp 479–493, Springer

Richter, M. M.: 1995, *The Knowledge Contained in Similarity Measures*, Invited talk at ICCBR95, http://www.cbr-web.org/documents/Richtericcbr95remarks.html

Richter, M. M.: 1998, *Introduction*, in M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, and S. Wess (eds.), *Case-Based Reasoning Technology, From Foundations to Applications*, pp 1–16, Springer

Riesbeck, C. and Schank, R. C. (eds.): 1989, *Inside Case-Based Reasoning.*, Lawrence Erlbaum

Schafer, J. B., Konstan, J. A., and Riedl, J.: 2001, *E-Commerce Recommendation Applications.*, *Data Min. Knowl. Discov.* **5(1/2)**, 115–153

Shimazu, H.: 1998, *A Textual Case-Based Reasoning System Using XML on the World-Wide Web*, in B. Smyth and P. Cunningham (eds.), *Advances in Case-Based Reasoning: Proceedings of the Fourth European Workshop on Case-Based Reasoning*, Vol. 1488 of *Lecture Notes in Computer Science*, pp 274–285, Springer

Shimazu, H.: 2002, *ExpertClerk: A Conversational Case-Based Reasoning Tool for Developing Salesclerk Agents in E-Commerce Webshops.*, *Artif. Intell. Rev.* **18(3-4)**, 223–244

Shneiderman, B.: 1986, *Designing the user interface: strategies for effective human-computer interaction*, Addison-Wesley Longman Publishing Co., Inc.

Smyth, B. and Cotter, P.: 1999, *Surfing the Digital Wave: Generating Personalised TV Listings using Collaborative, Case-Based Recommendation*, in K.-D. Althoff, R. Bergmann, and L. K. Branting (eds.), *Case-Based Reasoning Research and Development: Third International Conference on Case-Based Reasoning, ICCBR-99, Seeon Monastery, Germany, July 1999. Proceedings*, pp 561–571, Springer-Verlag Heidelberg

Smyth, B. and Keane, M. T.: 1993, *Retrieving Adaptable Cases: The Role of Adaptation Knowledge in Case Retrieval.*, in S. Wess, K.-D. Althoff, and M. M. Richter (eds.), *Topics in Case-Based Reasoning, First European Workshop, EWCBR-93, Kaiserslautern, Germany, November 1-5, 1993, Selected Papers*, pp 209–220, Springer

Smyth, B. and Keane, M. T.: 1995, *Remembering To Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems.*, in *IJCAI*, pp 377–383

Smyth, B. and McClave, P.: 2001, *Similarity vs. Diversity.*, in D. W. Aha and I. Watson (eds.), *Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, July 30 - August 2, 2001, Proceedings*, pp 347–361, Springer

Smyth, B. and McKenna, E.: 1998, *Modelling the Competence of Case-Bases.*, in B. Smyth and P. Cunningham (eds.), *Advances in Case-Based Reasoning, 4th European Workshop, EWCBR-98, Dublin, Ireland, September 1998, Proceedings*, pp 208–220, Springer

Stahl, A.: 2001, *Learning Feature Weights from Case Order Feedback.*, in D. W. Aha and I. Watson (eds.), *Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, July 30 - August 2, 2001, Proceedings*, pp 502–516, Springer

Stahl, A.: 2002, *Defining Similarity Measures: Top-Down vs. Bottom-Up.*, in S. Craw and A. D. Preece (eds.), *Advances in Case-Based Reasoning, 6th European Conference, ECCBR 2002 Aberdeen, Scotland, UK, September 4-7, 2002, Proceedings*, pp 406–420, Springer

Stahl, A. and Gabel, T.: 2003, *Using Evolution Programs to Learn Local Similarity Measures.*, in K. D. Ashley and D. G. Bridge (eds.), *Case-Based Reasoning Research and Development, 5th International Conference on Case-Based Reasoning, ICCBR 2003, Trondheim, Norway, June 23-26, 2003, Proceedings*, pp 537–551, Springer

Torrens, M., Faltings, B., and Pu, P.: 2002, *SmartClients: Constraints Satisfaction as a Paradigm for Scalable Intelligent Information Systems*, *Constraints* **7**, 49–69

Ware, C.: 2000, *Information Visualization: Perception for Design*, Morgan Kaufmann Publishers

Waszkiewicz, P., Cunningham, P., and Byrne, C.: 1999, *Case-based User Profiling in a Personal Travel Assistant*, in J. Kay (ed.), *User Modeling: Proceedings of the Seventh International Conference, UM99*, pp 323–325, Springer

Watson, I.: 1997, *Applying Case-based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann

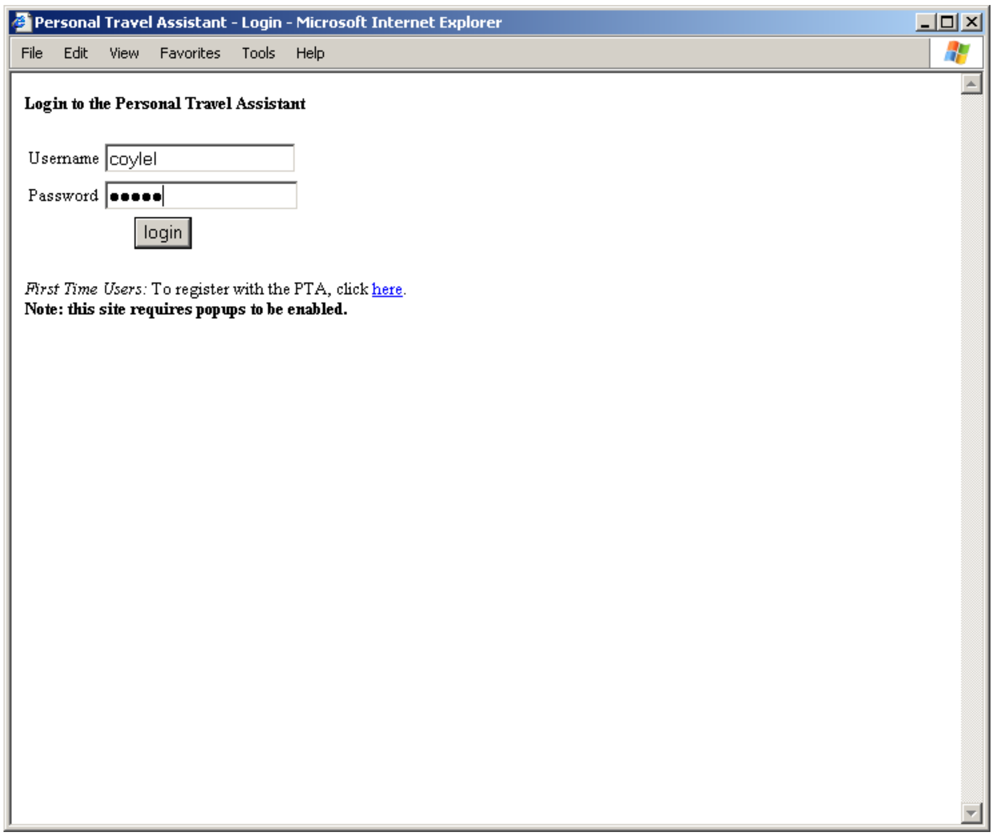
Watson, I. and Gardingen, D.: 1999, *A Distributed Case-Based Reasoning Application for Engineering Sales Support.*, in T. Dean (ed.), *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pp 600–605, Morgan Kaufmann

Wettschereck, D., Aha, D. W., and Mohri, T.: 1997, *A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms.*, *Artif. Intell. Rev.* **11(1-5)**, 273–314

Wilson, D. C.: 2001, *Case-Base Maintenance: The Husbandry of Experience*, *Ph.D. thesis*, Indiana University, Wirth, N.: 1977, *What Can We Do about the Unnecessary Diversity of Notation for Syntactic Definitions?*, *Commun. ACM* **20(11)**, 822–823

Appendix A

An Example Interaction with the PTA


Personal Travel Assistant - Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help

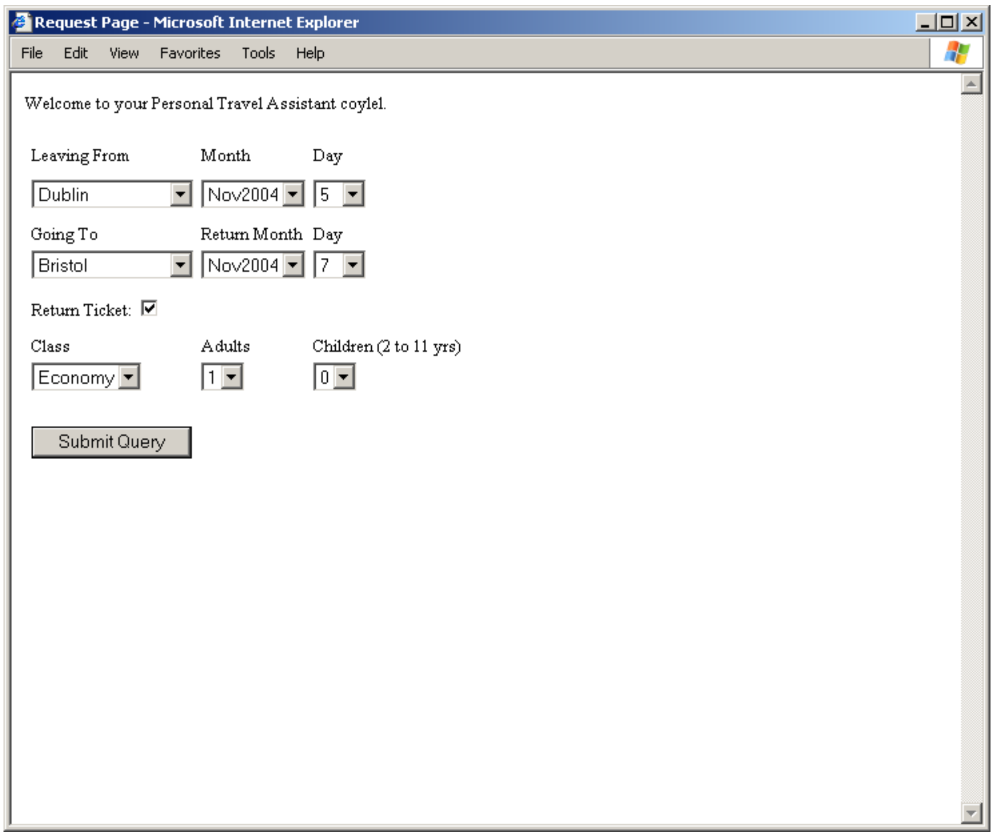
Login to the Personal Travel Assistant

Username coylel
Password ●●●●●

login

First Time Users: To register with the PTA, click here.
Note: this site requires popups to be enabled.

Figure A.1: User (`coylel`) Logs into the PTA


Request Page - Microsoft Internet Explorer

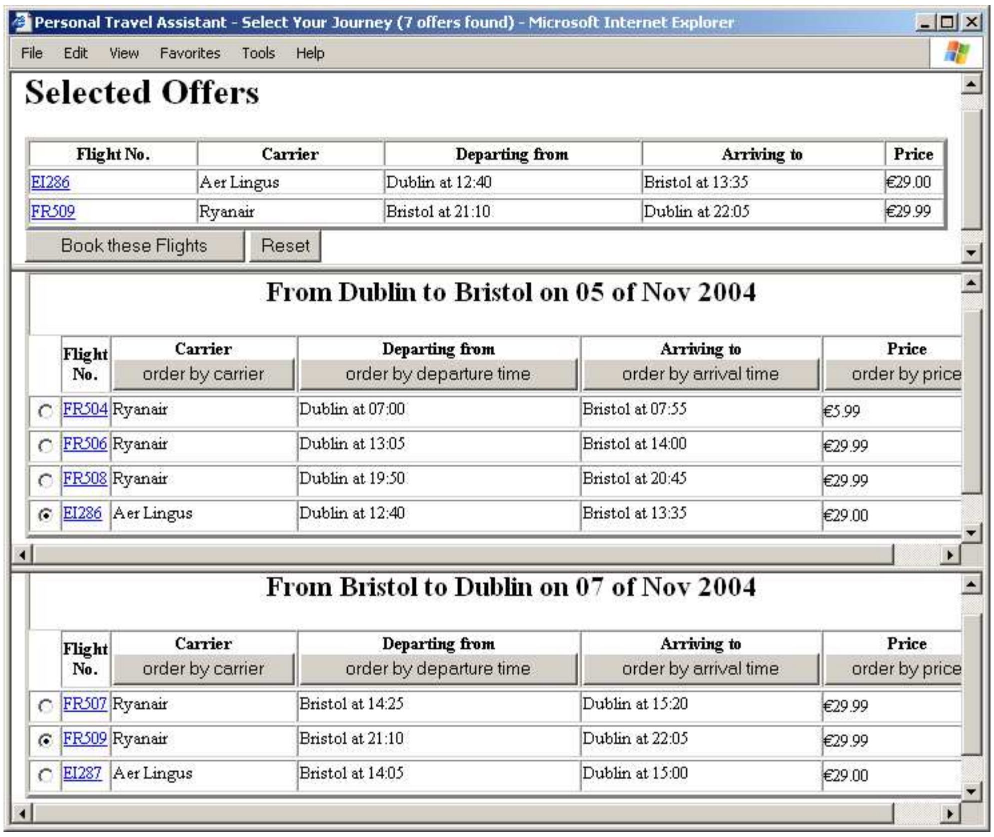
Welcome to your Personal Travel Assistant coylel.

Leaving From	Month	Day
Dublin	Nov2004	5
Going To	Return Month	Day
Bristol	Nov2004	7

Return Ticket: ✓

Class	Adults	Children (2 to 11 yrs)
Economy	1	0

Figure A.2: `coylel` makes a request for a return flight to Bristol departing on the 5th of November and returning on the 7th of November.


Personal Travel Assistant - Select Your Journey (7 offers found) - Microsoft Internet Explorer

Selected Offers

Flight No.	Carrier	Departing from	Arriving to	Price
EI286	Aer Lingus	Dublin at 12:40	Bristol at 13:35	€29.00
FR509	Ryanair	Bristol at 21:10	Dublin at 22:05	€29.99

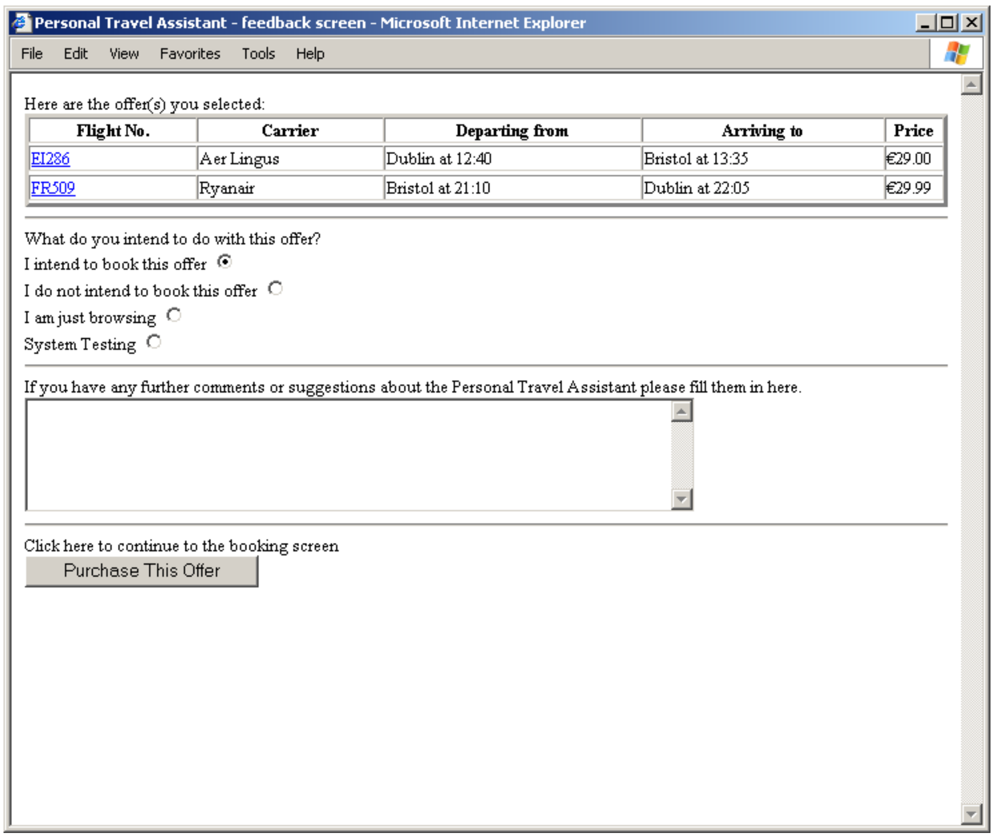
From Dublin to Bristol on 05 of Nov 2004

	Flight No.	Carrier	Departing from	Arriving to	Price
		order by carrier	order by departure time	order by arrival time	order by price
○	FR504	Ryanair	Dublin at 07:00	Bristol at 07:55	€5.99
○	FR506	Ryanair	Dublin at 13:05	Bristol at 14:00	€29.99
○	FR508	Ryanair	Dublin at 19:50	Bristol at 20:45	€29.99
[✓]	EI286	Aer Lingus	Dublin at 12:40	Bristol at 13:35	€29.00

From Bristol to Dublin on 07 of Nov 2004

	Flight No.	Carrier	Departing from	Arriving to	Price
		order by carrier	order by departure time	order by arrival time	order by price
○	FR507	Ryanair	Bristol at 14:25	Dublin at 15:20	€29.99
[✓]	FR509	Ryanair	Bristol at 21:10	Dublin at 22:05	€29.99
○	EI287	Aer Lingus	Bristol at 14:05	Dublin at 15:00	€29.00

Figure A.3: `coylel` is presented with 7 offers; four outgoing and three returning. For the outgoing leg he selects an Aer Lingus flight leaving Dublin at 12:40 and arriving in Bristol at 13:35 that costs €29. For the return leg he selects a Ryanair flight leaving Bristol at 21:10 and arriving in Dublin at 22:05 that costs €29.99.


Personal Travel Assistant - feedback screen - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Here are the offer(s) you selected:

Flight No.	Carrier	Departing from	Arriving to	Price
EI286	Aer Lingus	Dublin at 12:40	Bristol at 13:35	€29.00
FR509	Ryanair	Bristol at 21:10	Dublin at 22:05	€29.99

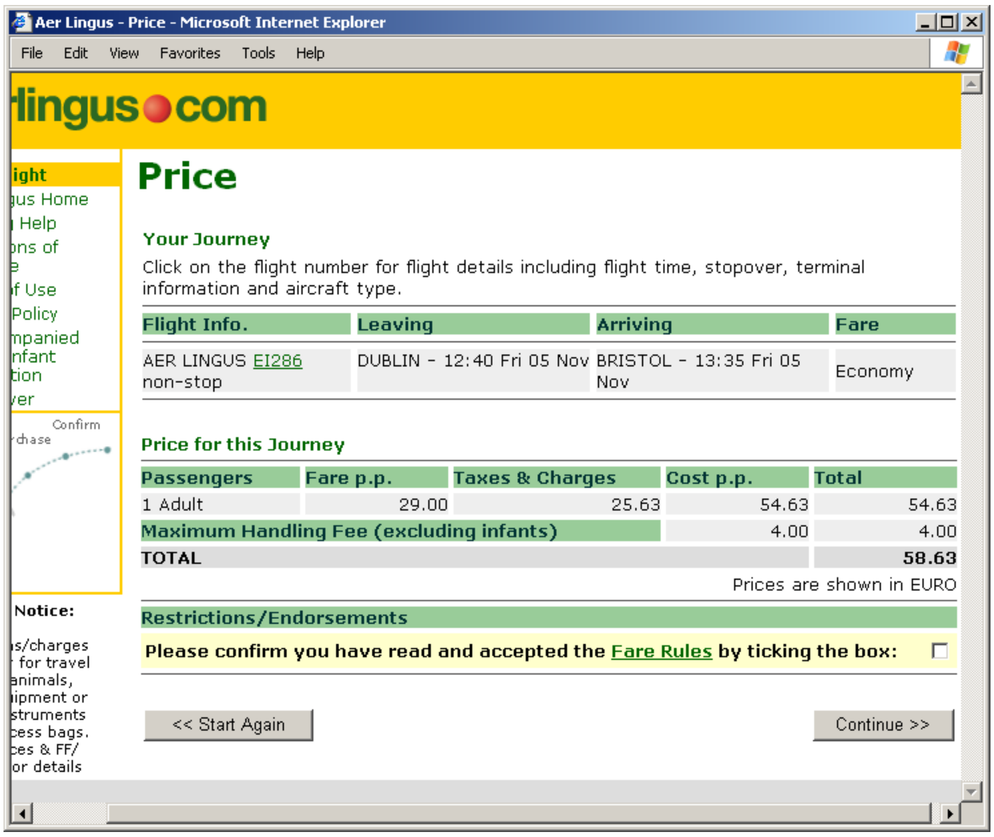
What do you intend to do with this offer?

[✓ I intend to book this offer]
I do not intend to book this offer
I am just browsing
System Testing

If you have any further comments or suggestions about the Personal Travel Assistant please fill them in here.

Click here to continue to the booking screen

Figure A.4: `coylel` clicks on the 'I intend to book this offer' radio button and proceeds to the hand-off pages. At this point his interaction with the PTA is over and he is dealing directly with Aer Lingus and Ryanair. The two hand-off windows pop-up and are shown in the following figures.


Aer Lingus - Price - Microsoft Internet Explorer
File Edit View Favorites Tools Help
lingus com
ight
jus Home
Help
ons of
e
f Use
Policy
mpanied
nfant
tion
ver
Confirm
chase

Price

Your Journey

Click on the flight number for flight details including flight time, stopover, terminal information and aircraft type.

Flight Info.	Leaving	Arriving	Fare
AER LINGUS EI286 non-stop	DUBLIN - 12:40 Fri 05 Nov	BRISTOL - 13:35 Fri 05 Nov	Economy

Price for this Journey

Passengers	Fare p.p.	Taxes & Charges	Cost p.p.	Total
1 Adult	29.00	25.63	54.63	54.63
Maximum Handling Fee (excluding infants)			4.00	4.00
TOTAL				58.63

Prices are shown in EURO

Restrictions/Endorsements

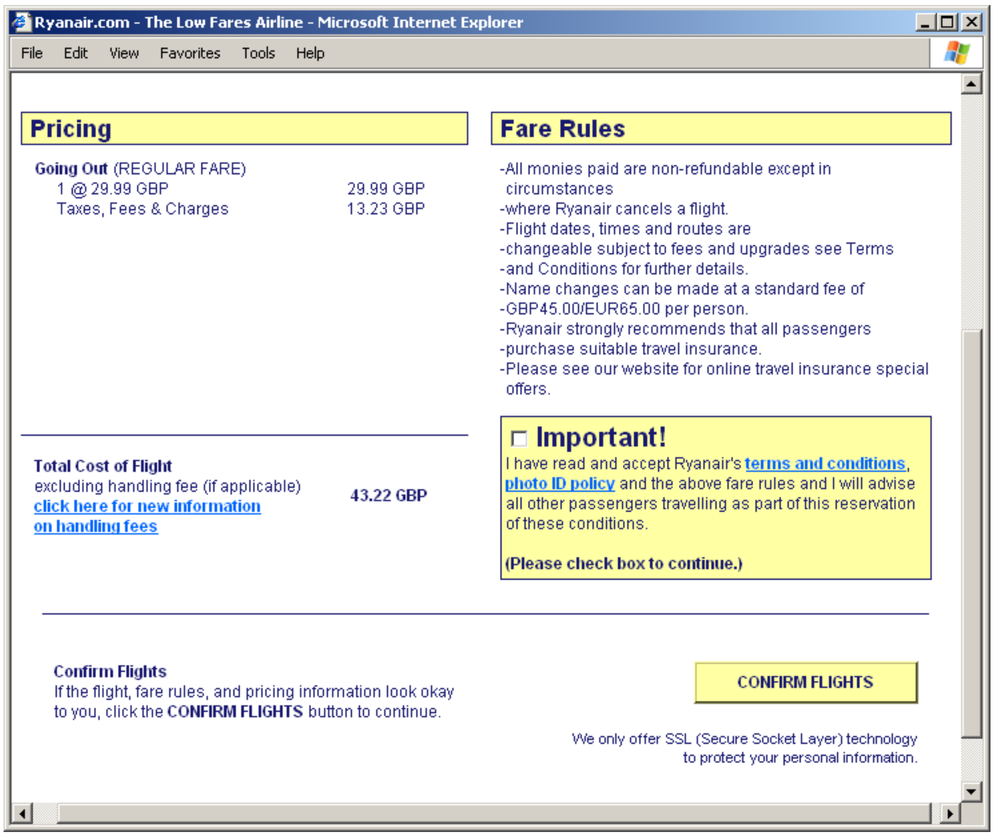
Please confirm you have read and accepted the Fare Rules by ticking the box: ☐

<< Start Again Continue >>

Notice:

s/charges for travel animals, ipment or struments cess bags. ces & FF/ or details

Figure A.5: The Aer Lingus pop-up window opens with the Aer Lingus booking page for the offer the user selected


Ryanair.com - The Low Fares Airline - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Pricing

Going Out (REGULAR FARE)

1 @ 29.99 GBP	29.99 GBP
Taxes, Fees & Charges	13.23 GBP

Fare Rules

-All monies paid are non-refundable except in circumstances
-where Ryanair cancels a flight.
-Flight dates, times and routes are
-changeable subject to fees and upgrades see Terms
-and Conditions for further details.
-Name changes can be made at a standard fee of
-GBP45.00/EUR65.00 per person.
-Ryanair strongly recommends that all passengers
-purchase suitable travel insurance.
-Please see our website for online travel insurance special offers.

Total Cost of Flight
excluding handling fee (if applicable) 43.22 GBP
click here for new information on handling fees

☐ Important!
I have read and accept Ryanair's terms and conditions, photo ID policy and the above fare rules and I will advise all other passengers travelling as part of this reservation of these conditions.

(Please check box to continue.)

Confirm Flights
If the flight, fare rules, and pricing information look okay to you, click the CONFIRM FLIGHTS button to continue.

CONFIRM FLIGHTS

We only offer SSL (Secure Socket Layer) technology to protect your personal information.

Figure A.6: The Ryanair pop-up window opens with the Ryanair booking page for the offer the user selected

Appendix B

The Test Scenarios

In the generation of the artificial scenarios for the evaluation (described in more detail in Chapter 7) the following directives were given to each of the users:

Weekend Holiday Please pick three weekends in the next three months and book trips to any of the following cities. You must select a different city every time and you must stay in the city on the Saturday night but apart from that you are free to select the departure day and return day (e.g. Friday to Monday; Saturday to Sunday):

- Barcelona
- Birmingham
- Bristol
- Brussels
- Edinburgh
- Glasgow
- London
- Manchester
- Paris

One Week Holiday : Please pick any time in the next three months and book holidays to three of the following cities. These trips must be more than 4 days and no more than 14 days in duration:

- Berlin
- Bologna
- Frankfurt
- Milan

- Rome
- Venice
- Stockholm

Appendix C

CBML Resources

C.1 The CBML Schema Document

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="case">
    <xs:complexType>
      <xs:choice>
        <xs:element name="structure" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="feature"
                maxOccurs="unbounded" type="structure_feature"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="similarity" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="feature" minOccurs="0"
                maxOccurs="unbounded" type="similarity_feature"/>
            </xs:sequence>
            <xs:attribute name="username" type="xs:ID" use="required"/>
            <xs:attribute name="extends" type="xs:IDREF" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:attribute name="domain" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="similarity_feature">
    <xs:choice>
      <xs:element name="feature" minOccurs="1" maxOccurs="unbounded"
       type="similarity_feature"/>
      <xs:element name="array" minOccurs="1" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="primary" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="secondary" minOccurs="0"
                  maxOccurs=”unbounded”>
                  <xs:complexType>
                    <xs:attribute name=”name” type=”xs:string”
                      use=”required”/>
                    <xs:attribute name=”similarity” type=”xs:double”
                      use=”required”/>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
              <xs:attribute name=”name” type=”xs:string” use=”required”/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name=”graph”  minOccurs=”1” maxOccurs=”1”>
      <xs:complexType>
        <xs:sequence>
          <xs:element name=”point” minOccurs=”0” maxOccurs=”unbounded”>
            <xs:complexType>
              <xs:attribute name=”difference” type=”xs:double”
                use=”required”/>
              <xs:attribute name=”similarity” type=”xs:double”
                use=”required”/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name=”type” type=”graphType” use=”required”/>
      </xs:complexType>
    </xs:element>
    <xs:element name=”measure” minOccurs=”1” maxOccurs=”1”>
      <xs:complexType>
        <xs:attribute name=”name” type=”xs:string” use=”required”/>
      </xs:complexType>
    </xs:element>
    <xs:element name=”exact” minOccurs=”1” maxOccurs=”1”/>
  </xs:choice>
  <xs:attribute name=”name” type=”xs:string” use=”required”/>
  <xs:attribute name=”weight” type=”positiveDouble” use=”required”/>
</xs:complexType>
<xs:simpleType name=”positiveDouble”>
  <xs:restriction base=”xs:double”>
    <xs:minInclusive value=”0”/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name=”normalisedDouble”>
  <xs:restriction base=”xs:double”>
    <xs:minInclusive value=”0”/>
    <xs:maxInclusive value=”1”/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name=”graphType”>
  <xs:restriction base=”xs:string”>
    <xs:enumeration value=”symmetrical”/>
    <xs:enumeration value=”asymmetrical”/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name=”structure_feature”>
  <xs:choice>
    <xs:element name=”complex”>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="feature" minOccurs="0" maxOccurs="unbounded"
            type="structure_feature"/>
        </xs:sequence>
        <xs:attributeGroup ref="references"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="symbol">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="enumeration" minOccurs="0"
            maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attributeGroup ref="references"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="string">
      <xs:complexType>
        <xs:attributeGroup ref="references"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="taxonomy">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="node" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attributeGroup ref="references"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="integer">
      <xs:complexType>
        <xs:all>
          <xs:element name="maxInclusive" minOccurs="0">
            <xs:complexType>
              <xs:attribute name="value" type="xs:int" use="required"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="minInclusive" minOccurs="0">
            <xs:complexType>
              <xs:attribute name="value" type="xs:int" use="required"/>
            </xs:complexType>
          </xs:element>
        </xs:all>
        <xs:attributeGroup ref="references"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="double">
      <xs:complexType>
        <xs:all>
          <xs:element name="maxInclusive" minOccurs="0">
            <xs:complexType>
              <xs:attribute name="value" type="xs:double" use="required"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="minInclusive" minOccurs="0">
            <xs:complexType>
              <xs:attribute name="value" type="xs:double" use="required"/>
            </xs:complexType>
          </xs:element>
```

```
          </xs:all>
          <xs:attributeGroup ref="references"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="boolean">
        <xs:complexType>
          <xs:attributeGroup ref="references"/>
        </xs:complexType>
      </xs:element>
    </xs:choice>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="discriminant" use="optional" default="true"
      type="xs:boolean"/>
    <xs:attribute name="solution" use="optional" default="false"
      type="xs:boolean"/>
    <xs:attribute name="mandatory" use="optional" default="true"
      type="xs:boolean"/>
  </xs:complexType>
  <xs:attributeGroup name="references">
    <xs:attribute name="name" type="xs:ID" use="optional"/>
    <xs:attribute name="ref" type="xs:IDREF" use="optional"/>
  </xs:attributeGroup>
  <xs:element name="enumeration">
    <xs:complexType>
      <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="node">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="node" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

C.2 The Request-Case Structure Document

```
<?xml version="1.0" encoding="UTF-8"?>
<case domain="pta.offer"
  xsi:noNamespaceSchemaLocation=
  "http://www.cs.tcd.ie/research_groups/mlg/CBML/Schema/cbmlv3.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <structure>
    <feature name="origin">
      <taxonomy name="Airports">
        <node name="ANY">
          <node name="Europe">
            <node name="Austria">
              <node name="SZG" />
              <node name="GRZ" />
              <node name="KLU" />
              <node name="LNZ" />
              <node name="VIE" />
            </node>
            <node name="Belgium">
              <node name="Brussels">
                <node name="BRU" />
                <node name="CRL" />
    </node>
    <node name="OST" />
  </node>
  <node name="Croatia">
    <node name="DBV" />
  </node>
  <node name="Czech Republic">
    <node name="PRG" />
  </node>
  <node name="Denmark">
    <node name="CPH" />
    <node name="AAR" />
    <node name="EBJ" />
  </node>
  <node name="Finland">
    <node name="TMP" />
    <node name="POR" />
  </node>
  <node name="France">
    <node name="EBU" />
    <node name="BES" />
    <node name="BIQ" />
    <node name="CCF" />
    <node name="CFE" />
    <node name="DNR" />
    <node name="EGC" />
    <node name="FNI" />
    <node name="LIG" />
    <node name="LRH" />
    <node name="LYS" />
    <node name="MRS" />
    <node name="MPL" />
    <node name="NCE" />
    <node name="Paris">
      <node name="CDG" />
      <node name="BVA" />
    </node>
    <node name="PIS" />
    <node name="PGF" />
    <node name="PUF" />
    <node name="RDZ" />
    <node name="RHE" />
    <node name="SXB" />
    <node name="TLS" />
    <node name="TUF" />
  </node>
  <node name="Germany">
    <node name="FKB" />
    <node name="Berlin">
      <node name="TXL" />
      <node name="SXF" />
    </node>
    <node name="Dusseldorf">
      <node name="DUS" />
      <node name="NRN" />
    </node>
    <node name="ERF" />
    <node name="Frankfurt">
      <node name="FRA" />
      <node name="HHN" />
  </node>
  <node name="FDH" />
  <node name="AOC" />
  <node name="LBC" />
  <node name="MUC" />
</node>
<node name="Greece">
  <node name="ATH" />
</node>
<node name="Holland">
  <node name="AMS" />
  <node name="GRQ" />
  <node name="MST" />
  <node name="EIN" />
</node>
<node name="Ireland">
  <node name="DUB" />
  <node name="KIR" />
  <node name="NOC" />
  <node name="ORK" />
  <node name="SNN" />
</node>
<node name="Italy">
  <node name="Bologna">
    <node name="BLQ" />
    <node name="FRL" />
  </node>
  <node name="AHO" />
  <node name="AOI" />
  <node name="BRI" />
  <node name="GOA" />
  <node name="Milan">
    <node name="LIN" />
    <node name="BGY" />
    <node name="MXP" />
  </node>
  <node name="OLB" />
  <node name="PMO" />
  <node name="NAP" />
  <node name="PSA" />
  <node name="PSR" />
  <node name="TRN" />
  <node name="TRS" />
  <node name="Rome">
    <node name="CIA" />
    <node name="FCO" />
  </node>
  <node name="VBS" />
  <node name="Venice">
    <node name="VCE" />
    <node name="TSF" />
  </node>
</node>
<node name="Norway">

  <node name="HAU" />
  <node name="TRF" />
</node>
<node name="Poland">
  <node name="WAW" />
```

```
</node>
<node name="Portugal">
  <node name="FAO" />
  <node name="LIS" />
</node>
<node name="Spain">
  <node name="AGP" />
  <node name="ALC" />
  <node name="BCN" />
  <node name="BIO" />
  <node name="GRO" />
  <node name="IBZ" />
  <node name="XRY" />
  <node name="MAD" />
  <node name="MJV" />
  <node name="PMI" />
  <node name="REU" />
  <node name="VLC" />
  <node name="VLL" />
  <node name="TFS" />
</node>
<node name="Sweden">
  <node name="MMX" />
  <node name="GSE" />
  <node name="Stockholm">
    <node name="NYO" />
    <node name="VST" />
  </node>
</node>
<node name="Switzerland">
  <node name="GVA" />
  <node name="ZRH" />
</node>
<node name="United Kingdom">
  <node name="ABZ" />
  <node name="BFS" />
  <node name="BHX" />
  <node name="BLK" />
  <node name="BOH" />
  <node name="BRS" />
  <node name="CWL" />
  <node name="EDI" />
  <node name="EMA" />
  <node name="INV" />
  <node name="JER" />
  <node name="Glasgow">
    <node name="PIK" />
    <node name="GLA" />
  </node>
  <node name="LBA" />
  <node name="LDY" />
  <node name="LPL" />
  <node name="London">
    <node name="LHR" />
    <node name="LGW" />
    <node name="LTN" />
    <node name="STN" />
  </node>
  <node name="MAN" />
  <node name="MME" />
```

```
          <node name="NCL" />
          <node name="NQY" />
        </node>
      </node>
    </node>
  </taxonomy>
</feature>
<feature name="destination">
  <taxonomy ref="Airports" />
</feature>
<feature name="distance">
  <integer>
    <minInclusive value="0" />
    <maxInclusive value="500" />
  </integer>
</feature>
<feature name="duration" mandatory="false">
  <integer>
    <minInclusive value="0" />
    <maxInclusive value="366" />
  </integer>
</feature>
<feature name="departuredayofweek">
  <symbol>
    <enumeration value="Mon" />
    <enumeration value="Tue" />
    <enumeration value="Wed" />
    <enumeration value="Thu" />
    <enumeration value="Fri" />
    <enumeration value="Sat" />
    <enumeration value="Sun" />
  </symbol>
</feature>
<feature name="returndayofweek">
  <symbol>
    <enumeration value="Mon" />
    <enumeration value="Tue" />
    <enumeration value="Wed" />
    <enumeration value="Thu" />
    <enumeration value="Fri" />
    <enumeration value="Sat" />
    <enumeration value="Sun" />
  </symbol>
</feature>
<feature name="tickettype">
  <symbol>
    <enumeration value="single" />
    <enumeration value="return" />
  </symbol>
</feature>
<feature name="dayofyear" discriminant="false">
  <integer>
    <minInclusive value="1" />
    <maxInclusive value="366" />
  </integer>
</feature>
<feature name="quality" mandatory="false" discriminant="false">
  <string />
</feature>
<feature name="username" discriminant="false" mandatory="true">
```

```
      <string />
    </feature>
    <feature name="userIP" mandatory="false" discriminant="false">
      <string />
    </feature>
    <feature name="urgency" mandatory="false" discriminant="false">
      <integer>
        <minInclusive value="0" />
      </integer>
    </feature>
    <feature name="bookingtimeofday" mandatory="false" discriminant="false">
      <integer>
        <minInclusive value="0" />
        <maxInclusive value="1440" />
      </integer>
    </feature>
    <feature name="bookingdayofweek" mandatory="false" discriminant="false">
      <string />
    </feature>
    <feature name="sessionID" discriminant="false">
      <integer />
    </feature>
  </structure>
</case>
```

C.3 The Request-Case Similarity Document

```
<?xml version="1.0" encoding="UTF-8"?>
<case domain="pta"
   xsi:noNamespaceSchemaLocation=
   "http://www.cs.tcd.ie/research_groups/mlg/CBML/Schema/cbmlv3.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <similarity username="default">
    <feature name="origin" weight="1.0">
      <measure name="similarity.LocationSimilarityMeasure" />
    </feature>
    <feature name="destination" weight="1.0">
      <measure name="similarity.LocationSimilarityMeasure" />
    </feature>
    <feature name="distance" weight="1.0">
      <graph type="symmetrical">
        <point difference="0" similarity="1" />
        <point difference="400" similarity="0" />
      </graph>
    </feature>
    <feature name="departuredayofweek" weight="1.0">
      <graph type="symmetrical">
        <point difference="0" similarity="1" />
        <point difference="3" similarity="0" />
      </graph>
    </feature>
    <feature name="returndayofweek" weight="1.0">
      <graph type="symmetrical">
        <point difference="0" similarity="1" />
        <point difference="3" similarity="0" />
      </graph>
    </feature>
    <feature name="tickettype" weight="0">
      <exact />
    </feature>
    <feature name="dayofyear" weight="0">
      <graph type="symmetrical">
        <point difference="0" similarity="1" />
        <point difference="7" similarity="1" />
        <point difference="28" similarity="0.5" />
        <point difference="90" similarity="0" />
      </graph>
    </feature>
    <feature name="duration" weight="1.0">
      <graph type="symmetrical">
        <point difference="0" similarity="1" />
        <point difference="7" similarity="0" />
      </graph>
    </feature>
    <feature name="urgency" weight="0">
      <graph type="symmetrical">
        <point difference="0" similarity="1" />
        <point difference="7" similarity="1" />
        <point difference="28" similarity="0.5" />
        <point difference="90" similarity="0" />
      </graph>
    </feature>
    <feature name="bookingtimeofday" weight="0">
      <graph type="symmetrical">
        <point difference="0" similarity="1" />
        <point difference="60" similarity="1" />
        <point difference="60" similarity="0.8" />
        <point difference="180" similarity="0.6" />
        <point difference="360" similarity="0" />
      </graph>
    </feature>
    <feature name="bookingdayofweek" weight="0">
      <exact />
    </feature>
  </similarity>
</case>
```

C.4 The Offer-Case Structure Document

```
<?xml version="1.0" encoding="UTF-8"?>
<case domain="pta.offer"
  xsi:noNamespaceSchemaLocation=
  "http://www.cs.tcd.ie/research_groups/mlg/CBML/Schema/cbmlv3.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <structure>
    <feature name="origin">
      <taxonomy name="Airports">
        <node name="null" />
        <node name="ANY">
          <node name="Europe">
            <node name="Austria">
              <node name="SZG" />
              <node name="GRZ" />
              <node name="KLU" />
              <node name="LNZ" />
              <node name="VIE" />
            </node>
            <node name="Belgium">
              <node name="Brussels">
                <node name="BRU" />
                <node name="CRL" />
              </node>
              <node name="OST" />
            </node>
            <node name="Croatia">
              <node name="DBV" />
            </node>
            <node name="Czech Republic">
              <node name="PRG" />
            </node>
            <node name="Denmark">
              <node name="CPH" />
              <node name="AAR" />
              <node name="EBJ" />
            </node>
            <node name="Finland">
              <node name="TMP" />
              <node name="POR" />
            </node>
            <node name="France">
              <node name="EBU" />
              <node name="BES" />
              <node name="BIQ" />
              <node name="CCF" />
              <node name="CFE" />
              <node name="DNR" />
              <node name="EGC" />
              <node name="FNI" />
              <node name="LIG" />
              <node name="LRH" />
              <node name="LYS" />
              <node name="MRS" />
              <node name="MPL" />
              <node name="NCE" />
              <node name="Paris">
                <node name="CDG" />
                <node name="BVA" />
              </node>
              <node name="PIS" />
              <node name="PGF" />
              <node name="PUF" />
              <node name="RDZ" />
              <node name="RHE" />
              <node name="SXB" />
              <node name="TLS" />
              <node name="TUF" />
            </node>
            <node name="Germany">
              <node name="FKB" />
              <node name="Berlin">
                <node name="TXL" />
                <node name="SXF" />
              </node>
              <node name="Dusseldorf">
                <node name="DUS" />
                <node name="NRN" />
              </node>
              <node name="ERF" />
              <node name="Frankfurt">
                <node name="FRA" />
                <node name="HHN" />
              </node>
              <node name="FDH" />
              <node name="AOC" />
              <node name="LBC" />
              <node name="MUC" />
            </node>
            <node name="Greece">
              <node name="ATH" />
            </node>
            <node name="Holland">
              <node name="AMS" />
              <node name="GRQ" />
              <node name="MST" />
              <node name="EIN" />
            </node>
            <node name="Ireland">
              <node name="DUB" />
              <node name="KIR" />
              <node name="NOC" />
              <node name="ORK" />
              <node name="SNN" />
            </node>
            <node name="Italy">
              <node name="Bologna">
                <node name="BLQ" />
                <node name="FRL" />
              </node>
              <node name="AHO" />
              <node name="AOI" />
              <node name="BRI" />
              <node name="GOA" />
              <node name="Milan">
                <node name="LIN" />
                <node name="BGY" />
                <node name="MXP" />
              </node>
              <node name="OLB" />
              <node name="PMO" />
              <node name="NAP" />
              <node name="PSA" />
              <node name="PSR" />
              <node name="TRN" />
              <node name="TRS" />
              <node name="Rome">
                <node name="CIA" />
                <node name="FCO" />
              </node>
              <node name="VBS" />
              <node name="Venice">
                <node name="VCE" />
                <node name="TSF" />
              </node>
            </node>
            <node name="Norway">
              <node name="HAU" />
              <node name="TRF" />
            </node>
            <node name="Poland">
              <node name="WAW" />
```

```
</node>
<node name="Portugal">
  <node name="FAO" />
  <node name="LIS" />
</node>
<node name="Spain">
  <node name="AGP" />
  <node name="ALC" />
  <node name="BCN" />
  <node name="BIO" />
  <node name="GRO" />
  <node name="IBZ" />
  <node name="XRY" />
  <node name="MAD" />
  <node name="MJV" />
  <node name="PMI" />
  <node name="REU" />
  <node name="VLC" />
  <node name="VLL" />
  <node name="TFS" />
</node>
<node name="Sweden">
  <node name="MMX" />
  <node name="GSE" />
  <node name="Stockholm">
    <node name="NYO" />
    <node name="VST" />
  </node>
</node>
<node name="Switzerland">
  <node name="GVA" />
  <node name="ZRH" />
</node>
<node name="United Kingdom">
  <node name="ABZ" />
  <node name="BFS" />
  <node name="BHX" />
  <node name="BLK" />
  <node name="BOH" />
  <node name="BRS" />
  <node name="CWL" />
  <node name="EDI" />
  <node name="EMA" />
  <node name="INV" />
  <node name="JER" />
  <node name="Glasgow">
    <node name="PIK" />
    <node name="GLA" />
  </node>
  <node name="LBA" />
  <node name="LDY" />
  <node name="LPL" />
  <node name="London">
    <node name="LHR" />
    <node name="LGW" />
    <node name="LTN" />
    <node name="STN" />
  </node>
  <node name="MAN" />
  <node name="MME" />
```

```
              <node name="NCL" />
              <node name="NQY" />
            </node>
          </node>
        </node>
      </taxonomy>
    </feature>
    <feature name="destination">
      <taxonomy ref="Airports" />
    </feature>
    <feature name="timeofday">
      <integer>
        <minInclusive value="0" />
        <maxInclusive value="1440" />
      </integer>
    </feature>
    <feature name="price">
      <double>
        <minInclusive value="0" />
      </double>
    </feature>
    <feature name="numberofhops">
      <integer>
        <minInclusive value="1" />
        <maxInclusive value="2" />
      </integer>
    </feature>
    <feature name="via" mandatory="false" discriminant="true">
      <taxonomy ref="Airports" />
    </feature>
    <feature name="totaltime">
      <integer>
        <minInclusive value="0" />
      </integer>
    </feature>
    <feature name="stopovertime">
      <integer>
        <minInclusive value="0" />
      </integer>
    </feature>
    <feature name="outoffer" mandatory="true" discriminant="false">
      <symbol>
        <enumeration value="departure" />
        <enumeration value="return" />
      </symbol>
    </feature>
    <feature name="carrier" mandatory="true">
      <symbol>
        <enumeration value="Ryanair" />
        <enumeration value="Mixed" />
        <enumeration value="Aer Lingus" />
      </symbol>
    </feature>
    <feature name="selected" mandatory="true" discriminant="false"
      solution="true">
      <boolean />
    </feature>
  </structure>
</case>
```

C.5 The Offer-Case Similarity Document

```
<?xml version="1.0" encoding="UTF-8"?>
<case domain="pta"
  xsi:noNamespaceSchemaLocation=
  "http://www.cs.tcd.ie/research_groups/mlg/CBML/Schema/cbmlv3.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <similarity username="default">
    <feature name="origin" weight="1">
      <measure name="similarity.LocationSimilarityMeasure"/>
    </feature>
    <feature name="destination" weight="1">
      <measure name="similarity.LocationSimilarityMeasure"/>
    </feature>
    <feature name="totaltime" weight="1">
      <graph type="asymmetrical">
        <point difference="-Infinity" similarity="0" />
        <point difference="-1010" similarity="0" />
        <point difference="1010" similarity="1" />
        <point difference="Infinity" similarity="1" />
      </graph>
    </feature>
    <feature name="price" weight="1">
      <graph type="asymmetrical">
        <point difference="-Infinity" similarity="0" />
        <point difference="-220" similarity="0" />
        <point difference="220" similarity="1" />
        <point difference="Infinity" similarity="1" />
      </graph>
    </feature>
    <feature name="stopovertime" weight="1">
      <graph type="asymmetrical">
        <point difference="-Infinity" similarity="0" />
        <point difference="-905" similarity="0" />
        <point difference="905" similarity="1" />
        <point difference="Infinity" similarity="1" />
      </graph>
    </feature>
    <feature name="numberofhops" weight="1">
      <graph type="asymmetrical">
        <point difference="-Infinity" similarity="0" />
        <point difference="-1" similarity="0" />
        <point difference="1" similarity="1" />
        <point difference="Infinity" similarity="1" />
      </graph>
    </feature>
    <feature name="timeofday" weight="1">
      <graph type="symmetrical">
        <point difference="0" similarity="1" />
        <point difference="720" similarity="0" />
      </graph>
    </feature>
    <feature name="via" weight="1">
      <measure name="similarity.LocationSimilarityMeasure"/>
    </feature>
    <feature name="outoffer" weight="0">
      <exact />
    </feature>
    <feature name="carrier" weight="1">
      <graph type="symmetrical">
        <point difference="0" similarity="1" />
        <point difference="1" similarity="0.5" />
        <point difference="2" similarity="0" />
      </graph>
    </feature>
   </similarity>
  </case>
```

C.6 CBML Java Interfaces

C.6.1 The Case Interface

```
package cbml.cbr;
import java.io.Serializable;
/**
 * This is a CBML Case Interface. The CBML parsers read Case Content
 * Documents and convert the case bases into CBML cases. These cases
 * may then be cast by the user into an implementation of this interface.
 * @author Lorcan Coyle
 * @see cbml.cbr.Feature
 * @see cbml.cbr.CaseStruct
 * @version 3.0
 */
public interface CBMLCase extends Serializable {
  /**
   * Adds the specified feature to this case.
   * @param newFeature the feature to be added to this case.
   * @return <code>true</code> if the feature was added succesfully,
   * <code>false</code> otherwise
   */
  boolean addFeature(Feature newFeature);
  /**
   * Returns a clone of this Case.
   * @return a clone of this instance.
   */
  Object clone();
  /**
   * Returns the feature at the specified path in this case.
   * @return the feature at the specified path in this case or
   * <code>null</code> if there is no feature at the specified path.
   * @param featurePath path whose associated feature is to be returned.
   */
  Feature getFeature(String featurePath);
  /**
  * Returns the name of this case.
  * @return the name of this case.
  */
  String getName();
  /**
   * Returns the solution <code>Feature</code> associated with this case.
   * @return the solution <code>Feature</code> associated with this case
   */
  Feature getSolution();
  /**
   * Removes the feature at the specified featurePath. Returns the removed
   * feature.
   * @param featurePath the path of the feature to be removed.
   * @return the removed feature  or <code>null</code> if there is no
   * feature at the specified path.
   */
  Feature removeFeature(String featurePath);
  /**
   * Sets the name for this case.
   * @param newName the new name of this case.
   */
  void setName(String newName);
  /**
   * Sets the solution <code>Feature</code> of this case to the specified
   * value.
   * @param solution the solution <code>Feature</code> to be associated
   * with this case.
   */
  void setSolution(Feature solution);

}
```

C.6.2 The Local Similarity Measure Interface

```
package cbml.cbr;
/**
 * This is an interface for a similarity measure. It must be implemented by
 * all CBML similarity measures.
 * @author Lorcan Coyle
 * @see cbml.cbr.SimilarityProfile
 */
public interface SimilarityMeasure extends java.io.Serializable {
  /**
   * Returns the similarity between the two specified features. It should
   * be noted that some similarity measures may be asymmetrical so the order
   * of the features being passed into this function should be ordered. The
   * returned similarity value is the similarity of the second feature
   * (<code>feature2</code>) in relation to the first feature
   * (<code>feature1</code>).
   * @return  the similarity between the two specified features. This value
   * should be normalised (i.e. similarity => [0..1])
   * @param feature1 the candidate or base feature.
   * @param feature2 the test feature.
   */
  double calculateSimilarity(Feature feature1, Feature feature2);

  /**
   * Creates and returns a copy of this Similarity Measure.
   * @return a clone of this instance.
   */
  Object clone() throws CloneNotSupportedException;
}
```