MAKING BETTER RECOMMENDATIONS WITH ONLINE PROFILING AGENTS

DANNY OH CHIN HOCK

NATIONAL UNIVERSITY OF SINGAPORE

2003

MAKING BETTER RECOMMENDATIONS WITH ONLINE PROFILING AGENTS

DANNY OH CHIN HOCK

(B.SC., COMPUTER AND INFORMATION SCIENCES)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2003

ACKNOWLEDGEMENT

In the course of living, there will always be a few people that come across to us as special and they are the ones that will leave a lasting impression on us by teaching us through their demeanor the true meaning of what it is to learn and to live. Prof. Tan Chew Lim is certainly one of them. With his simple refined demeanor, he patiently sought to give me clear guidance whenever I needed them. His tremendous faith in me, especially in times where my research efforts appeared to be going nowhere, proved to be instrumental in guiding me out of seemingly blind alleys. And, the freedom he gave me in pursing new and sometimes radical ideas taught me how to think and perceive creatively. I am deeply thankful to Prof. Tan for his support, guidance and contribution to this thesis by giving me space, that most precious of gifts - space to work and space to *be*. It is a joy to work with him.

I would like to express my love and gratitude to my mother and belated father, without whom this thesis would not have come into existence.

Finally, I am also thankful to my spiritual teachers and the greatest guru of all: life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SUMMARY

In recent years, we have witnessed the success of autonomous agents applying machine learning techniques across a wide range of applications. However, agents applying the same machine learning techniques in online applications have not been so successful. Even agent-based hybrid recommender systems that combine information filtering techniques with collaborative filtering techniques have only been applied with considerable success to simple consumer goods such as movies, books, clothing and food. Complex, adaptive autonomous agent systems that can handle complex goods such as real estate, vacation plans, insurance, mutual funds, and mortgage have yet emerged. To a large extent, the reinforcement learning methods developed to aid agents in learning have been more successfully deployed in offline applications. The inherent limitations in these methods have rendered them somewhat ineffective in online applications. Moreover, we feel that existing implementations of interactive learning method for online systems are simply impractical as the state-action space is simply too large for the agent to explore within its lifetime. This is further exacerbated by the short attention time-span of typical online users.

In this thesis, we postulate that a small amount of prior knowledge and human-provided input can dramatically speed up online learning. We demonstrate that our agent HumanE - with its prior knowledge or “experiences” about a complex domain such as real estate - can effectively assist users in identifying requirements, especially unstated ones, quickly and unobtrusively. The experimental results showed that the use of HumanE for complex multidimensional domains such as real estate can result in higher customer satisfaction as it can learn faster via a supplied initial policy and is able to elicit trust from users through its user-friendly interface, quality recommendations and excellent performance. HumanE addresses the problem of poor learning when implementing online implementation of large-scale autonomous agent-based recommender systems for several complex domains through the use of a supplied initial policy which allows it to make more “knowledgeable” exploratory recommendations.

Chapter 1

INTRODUCTION

1.1 Summary

In this chapter, we present the motivations and the contributions of this thesis as well as its organization.

1.2 Motivations

Electronic profiling has become the norm in most e-commerce websites. Whether you are making online purchases or using online services, you certainly would need to go through the tedious task of filling up a questionnaire. Merchants would then use the information provided to create an initial electronic profile. Subsequent specifications of user preferences such as keywords used in product searching, goods purchased or placed in wish-lists are used to refine the user profile without much user intervention.

This technique of learning user behavior through the creation of a user profile has been used rather successfully by certain agent-based recommender systems, namely, information filtering (IF) systems and collaborative filtering (CF) systems. IF involves continuous analysis of product content and attributes and the development of a personal user profile which will then be used to produce useful recommendations.

However, IF agents lack the ability to make serendipitous discoveries of new user preferences. CF functions by identifying users with similar tastes and using their opinions (usually by asking them to rate the product on a predefined scale) to recommend items. But, CF systems suffer from the reliance of user ratings which make recommending new or obscure items very difficult. Ongoing research work such as the GroupLens Research Project [45] has successfully combined the two techniques to form hybrid recommender systems that have proven that they can make better recommendations than using either IR systems or CF systems alone.

Unfortunately, the successes of these systems have been restricted to simple consumer goods such as movies, books, clothing and food. When the IR and/or CF techniques plus other reinforcement learning methods are applied in online applications for complex

consumer products such as real estate, vacation plans, insurance, mutual funds, and mortgages, they fail to enjoy much success.

This is because agents operating in complex domains require a substantial amount of knowledge and it is difficult to build such agents as it requires too much insight, understanding and effort from the end-user, since the user has to endow the agent with explicit knowledge (specifying this knowledge in an abstract language) and item-maintain the agent's rules over time (as work habits or interests change, etc.). This approach of making the end-user program the interface agent has proven to be feasible for simple tasks [25] but not so for complex ones.

Other agent developers tried to endow an interface agent with extensive domain-specific background knowledge about the application and the user (called a domain model and user model respectively). This knowledge-based approach is adopted by the majority of people working in AI on intelligent user interfaces [20, 24, 68] for simple tasks. The disadvantage of this approach is that even for simple tasks it requires a huge amount of work from the knowledge engineer. A large amount of application-specific and domain-specific knowledge has to be entered into the agent's knowledge base. Little of this knowledge or the agent's control architecture can be used when building agents for other applications. Another problem is that the knowledge of the agent is fixed once and for all. It cannot be customized to individual user habits and preferences. The possibility of providing an agent with all the knowledge it needs to always comprehend the user's sometimes unpredictable actions is questionable. Furthermore, there is also a problem with trust. It is probably not a good idea to give a user an interface agent that is very sophisticated, qualified and autonomous from the start. Schneiderman [7] has argued convincingly that such an agent would leave the user with a feeling of loss of control and understanding. Since the agent has been programmed by someone else, the user may not have a good model of the agent's limitations, the way it works, etc.

Another reason for the low success rate of agent-mediated systems for complex domains is that many reinforcement learning implementations assume that the agent developed knows nothing about the environment to begin with, and that the agent must gain all of its information by exploration and subsequent exploitation of learned knowledge. When dealing with a real, complex online system such as a large-scale real estate listing and

brokerage application, however, this approach is simply not practical. Typically, the state-space is too large to explore satisfactorily within the lifetime of the agent (much less within the attention time-span of typical online users). Worse still, making "random" exploratory recommendations can frustrate and disappoint the user, potentially causing the user to abandon the system totally.

1.3 Contributions

In our work, we explore an alternative approach to building autonomous interface or profiling agents that relies on Machine Learning techniques for complex products. In this thesis, the complex product is real estate properties. We also aim to resolve the problems confounded by the "knowledge-based approach" to building profiling agents.

Accumulated knowledge in the form of memories and experiences allows humans to go about performing daily tasks. In the real world, we often go to a human real estate agent for assistance in selling or acquiring real estate properties. We naturally expect the agent to be an expert in the real estate domain, and hence able to offer suitable advice and recommendations. Certainly, we do not expect the real estate agent to have no knowledge about the real estate domain. Hence, in order to take our prior knowledge (which are often implicit) and incorporate them into a reinforcement learning framework, we have examined in this work the idea of supplying the agent with an initial policy about the real estate domain in the learning algorithm for HumanE.

The learning approach is inspired by the metaphor of a smart and experienced personal assistant and a similar approach has been reported upon by Kaelbling [74] working on mobile robots. In the real world, we usually tend to hire smart and experienced people for such tasks. Even though, the personal assistant is not very familiar with the habits and preferences of his or her employer and may not even be very helpful, he or she must prove his or her abilities in a relatively short span of time with the help of prior knowledge and experiences accumulated previously.

The goal of our research is to demonstrate that the learning approach can present a satisfactory solution to developing effective and practical profiling agents for use in large, complex and multi-dimensional domains.

We believe that the learning approach has several advantages over past approaches. First, it requires less work from the end-user and application developer to specify initial knowledge. Second, the agent is potentially more competent at the initial stage of use and thus can elicit greater trust from the user. Thirdly, the agent can more easily adapt to the user over time and become customized to individual and organizational preferences and habits.

Furthermore, the agent framework and architecture can be transferred easily to other complex domains. Finally, the approach helps in transferring information, habits and know-how among the different users of a community.

1.4 Organization

The rest of the thesis is organized as follows:

- Chapter 2 – We discuss some related work pertaining to agent-mediated e-commerce systems and the corresponding AI techniques.

- Chapter 3 – We introduce a general model of agent learning and explain its working in the context of the real estate domain.

- Chapter 4 – We discuss in detail the proposed model of agent learning using a running example.

- Chapter 5 – We discuss the experimental findings obtained when we apply the general agent learning model to the real estate domain and explain the advantages of using an initial policy for better performance of web agents.

- Chapter 6 – We conclude the thesis and outline future work.

Chapter 2

RELATED WORKS

2.1 Summary

Intelligent agents help to automate a variety of tasks including those involved in buying and selling products over the Internet. This chapter surveys several of these agent-mediated e-commerce systems. We then discuss the various AI techniques that support agent mediation and conclude with the challenges faced when applying these techniques to complex domains.

2.2 Introduction

Intelligent agents are particularly useful for the information-rich and process-rich environment of e-commerce as they are personalized, continuously running and semi-autonomous. E-commerce encompasses a broad range of issues including security, trust, reputation, law, payment mechanisms, advertising, ontologies, online catalogs, intermediaries, multimedia shopping experiences, and back-office management. Agent technologies can be applied to any of these areas where a personalized, continuously running, semi-autonomous behavior is desirable. However, certain characteristics will determine to what extent agent technologies are appropriate. Generally, the more time and money that can be saved through automation, the easier it is to express preferences, the lesser the risks of making sub-optimal transaction decisions, and the greater the loss for missed opportunities, the more appropriate it is to employ agent technologies in e-commerce.

Intelligent agents will play an increasing variety of roles as mediators in e-commerce [23]. This section explores these roles, their supporting technologies, and how they relate to e-commerce in its three main forms: business-to-business, business-to-consumer, and consumer-to-consumer transactions.

2.3 Roles of agents as mediators in e-commerce

The roles of agents as mediators in e-commerce typically fall into the following three categories:

- Product Broker
    - Comprises of the retrieval of information to help determine what to buy. This includes product evaluation based on consumer-provided criteria to come up with a "consideration set" of products.
    - Examples include PersonaLogic [57], Firefly [27, 72], Apt Decision agent [65], and RentMe [17, 18].
- Merchant Broker
    - Combines the "consideration set" from Product Brokering with merchant-specific information to help determine who to buy from. This includes merchant evaluation based on consumer-selected criteria (e.g. price, warranty, availability, delivery time, reputation, etc.)
    - Examples include BargainFinder [9], Jango [43, 59], and Kasbah [1, 45].
- Negotiator
    - Determines the terms of the transaction. Negotiation varies in duration and complexity depending on the market. In traditional retail markets, prices and other aspects of the transaction are often fixed leaving no room for negotiation. In other markets (e.g. stocks, automobile, fine art, local markets, etc.), the negotiation of price or other aspects of the deal are integral to product and merchant brokering.
    - Examples include OnSale [55], eBay [30], AuctionBot [5], and Tete-a-Tete [38, 71].

The personalized, continuously-running, semi-autonomous nature of agents make them well-suited for mediating those consumer behaviors involving information filtering and retrieval, personalized evaluations, complex co-ordinations, and time-based interactions.

2.4 Agent technologies for e-commerce

Most of today's agent-mediated e-commerce systems are powered by AI technologies. In this section, we review several AI technologies that support the systems described earlier on, discuss user interface challenges, and then focus on issues and technologies concerning the next-generation agent-mediated e-commerce infrastructure.

2.4.1 Recommender systems

The majority of product recommender systems are developed using content-based, collaborative-based or constraint-based filtering methods as their underlying technology.

In content-based filtering [2, 33, 39, 52] the system processes information from various sources and tries to extract useful features and elements about its content. The techniques used in content-based filtering can vary greatly in complexity. Keyword-based search is one of the simplest techniques that involve matching different combinations of keywords (sometimes in Boolean form). A more advanced form of filtering is the one based on extracting semantic information from a document's contents. This can be achieved by using techniques like associative networks of keywords in a sentence or price list, or directed graphs of words that form sentences.

Systems like BargainFinder and Jango try to collect information (e.g. product descriptions, prices, reviews, etc.) from many different web information sources. These sources were intended to be read by humans and their content is rendered accordingly (i.e. in HTML). Different sources have presentation methods, so recommender systems have to adjust their interaction methods depending upon the web site. Since there is no standard way of defining and accessing merchant offerings, most recommender systems employ "wrappers" to transform the information from a specific website into a locally common format. The recent adoption of XML has made it easier for these systems to collect information.

Different systems adopt different approaches to creating wrappers. In BargainFinder, the URLs of online CD stores and the wrapper methods (i.e. searching for a product and getting its price) are hard-coded by the programmers. This method worked well initially but there is a need to maintain the wrapper for each site whenever it changes its access methods or catalog presentation format. Jango helps automate the creation of wrappers for new sites by generalizing from example query responses to online merchant

databases. This technique is not perfect, but boasts a nearly 50% success rate in navigating random websites [53]. Firefly uses a collaborative-based filtering technology [56, 72, 75] to recommend products to consumers. Systems using collaborative techniques use feedback and ratings from different consumers to filter out irrelevant information. These systems do not attempt to analyze or "understand" the features or the descriptions of the products. Rather, they use consumers' rankings to create a "likeability" index for each product. This index is not global, but is statistically computed for each user on the fly by using the profiles of other users with similar interests. Products that are liked by similar-minded people will have priority over products that are disliked.

As in content-based approaches, constraint-based filtering uses features of items to determine their relevance. However, unlike most feature-based techniques which access data in their native formats, constraint-based techniques require that the problem and solution space be formulated in terms of variables, domains, and constraints. Once formulated in this way, however, a number of general purpose (and powerful) constraint satisfaction problem (CSP) techniques can be employed to find a solution [26, 73].

Many problems can be formulated as a CSP such as scheduling, planning, configuration, and machine vision problems. In PersonaLogic, CSP techniques are used during product brokering to evaluate product alternatives. Given a set of constraints on product features, PersonaLogic filters products that do not meet the given "hard" constraints and prioritizes the remaining products using "soft" constraints (which need not be completely satisfied).

Tete-a-Tete uses CSP techniques to assist shoppers during product brokering, merchant brokering, and negotiation. This is achieved by consumers providing product constraints (as in PersonaLogic) as well as merchant constraints such as price, delivery time, warranty, etc. Hard and soft constraints are used to filter and prioritize products and merchants as well as construct a multi-attribute utility that is used to negotiate with the merchants. Tete-a-Tete's argumentative style of negotiation resembles a distributed CSP [76] with merchants providing counter-proposals to each customer's critiques [61].

2.4.2 Profiling-based recommender systems

In this section, we talk about a special class of agents - electronic profiling agents, and their roles in agent-based recommender systems. Additionally, we discuss the limitations of existing implementations of these systems and ask if we are expecting too much from our agents.

2.5 Electronic profiling

Electronic profiling has become the norm in most e-commerce websites. Whether you are making online purchases or using online services, you certainly would need to go through the tedious task of filling up a questionnaire. Merchants would then use the information provided to create an initial electronic profile. Subsequent specifications of user preferences such as keywords used in product searching, goods purchased or placed in wish-lists are used to refine the user profile without much user intervention. This technique of learning user behavior through the creation of a user profile has been used rather successfully by recommender systems such as information retrieval (IR) systems, information filtering (IF) systems and collaborative filtering (CF) systems.

2.5.1 Information retrieval systems

Information retrieval (IR) systems allow users to express queries to select documents that match a topic of interest. IR systems may index a database of documents using the full text of the document or only document abstracts. Sophisticated systems rank query results using a variety of heuristics including the relative frequency with which the query terms occur in each document, the adjacency of query terms, and the position of query terms. IR systems also may employ techniques such as term stemming to match words such as “retrieve,” “retrieval,” and “retrieving” [62]. IR systems are generally optimized for ephemeral interest queries, such as looking up a topic in the library [11]. In the Internet domain, popular IR systems include Google for web pages [35] and Google Groups [36] for discussion list postings.

An IR front-end is useful in a recommender system both as a mechanism for users to identify specific products about which they would like to express an opinion and for narrowing the scope of recommendation. For example, MovieLens [51] allows users to specifically request recommendations for newer movies, for movies released in particular time periods, for particular movie genres such as comedy and documentary, and for

various combinations of movie. However, the knowledge that a user can acquire from such systems depends predominantly on a user's skill to query the system and to assimilate the results. IR techniques are less valuable in the actual recommendation process, since they capture no information about user preferences other than the specific query.

2.5.2 Information filtering systems

Information filtering (IF) systems require a profile of user needs or preferences. The simplest systems require the user to create this profile manually or with limited assistance. Examples of these systems include: spam killers that are used to filter out advertising, e-mail filtering software that sorts e-mail into categories based on the sender, and new-product notification services that request notification when a new book or album by a favorite author or artist is released. More advanced IF systems may build a profile by learning the user's preferences. A wide range of agents, including Maes' agents for e-mail and Usenet news filtering [49] and Lieberman's Letizia [48], employ learning techniques to classify, dispose of, or recommend documents based on the user's prior actions. Similarly, Cohen's Ripper system has been used to classify e-mail [21]; alternative approaches use other learning techniques and term frequency [14]. More complex IF systems provide periodic personalized digests of material from sources such as news wires, discussion lists, and web pages [11].

One embodiment of IF techniques is software agents. These programs exhibit a degree of autonomous behavior, and attempt to act intelligently on behalf of the user for whom they are working. Agents maintain user interest profiles by updating them based on feedback on whether the user likes the items selected by the current profile. Research has been conducted in various feedback generation techniques, including probabilistic models, genetic algorithms and neural network based learning algorithms [7]. NewT is a filtering agent for Usenet news based on genetic algorithm learning techniques [49]. It performs full text analysis of articles using vector-space technique. Amalthaea is a multi-agent system for personalized filtering, discovery and monitoring of information sources in the World Wide Web domain [49].

IR and IF systems can be extremely effective at identifying documents that match a topic of interest, and at finding documents that match particular patterns (e.g. discarding email

with the phrase “Get Rich Fast” in the title). Unlike human editors, however, these systems cannot distinguish between high-quality and low-quality documents on the same topic. As the number of documents on each topic continues to grow, even the set of relevant documents will become too large to review. For some domains, therefore, the most effective filters must incorporate human judgments of quality.

Information filtering techniques have a central role in recommender systems. IF involves continuous analysis of product content and attributes and the development of a personal user profile which will then be used to produce useful recommendations. The user profile is particularly valuable when a user encounters new content that has not been rated before. IF techniques also have an important property that they do not depend on having other users in the system, let alone users with similar tastes. IF techniques can be effective but they suffer certain drawbacks, including requiring a source of content information, and the inability to make serendipitous discoveries of new user preferences.

2.5.3 Collaborative filtering systems

Collaborative filtering (CF) systems build a database of user opinions of available items. They use the database to find users whose opinions are similar (i.e. those that are highly correlated) and make predictions of user opinion on an item by combining the opinions of other likeminded individuals. In their purest form, CF systems do not consider the content of the documents at all, relying exclusively on the judgment of humans as to whether the document is valuable. In this way, collaborative filtering attempts to recapture the cross-topic recommendations that are common in communities of people.

Tapestry [46], one of the first computer-based collaborative filtering systems, was designed to support a small, close-knit community of users. Users could filter all incoming information streams, including email and Usenet news articles. When users evaluated a document, they could annotate it with text, with numeric ratings, and with Boolean ratings. Other users could form queries such as “show me the documents that Mary annotated with ‘excellent’ and Jack annotated with ‘Sam should read.’” A similar approach is used in Maltz and Ehrlich’s active collaborative filtering [50], which provides an easy way for users to direct recommendations to their friends and colleagues through a Lotus Notes database.

Collaborative filtering for large communities cannot depend on each person knowing the others. Several systems use statistical techniques to provide personal recommendations of documents by finding a group of other users, known as neighbours that have a history of agreeing with the target user. Once a neighborhood of users is found, particular documents can be evaluated by forming a weighted composite of the neighbors' opinions of that document. Similarly, a user can request recommendations for a set of documents to read and the system can return a set of documents that is popular within the neighborhood. These statistical approaches, known as automated collaborative filtering, typically rely upon ratings as numerical expressions of user preference. Several ratings-based automated collaborative filtering systems have been developed. The GroupLens Research system [47, 56] provides a pseudonymous collaborative filtering solution for Usenet news and movies. Ringo [72] and Video Recommender [41] are email and web systems that generate recommendations on music and movies respectively, suggesting collaborative filtering to be applicable to many different types of media. Recently, a number of systems have begun to use observational ratings; the system infers user preferences from actions rather than requiring the user to explicitly rate an item [70]. A wide range of web sites have begun to use CF recommendations in a diverse set of domains including books, grocery products, art, entertainment, and information.

Collaborative filtering techniques can be an important part of a recommender system. One key advantage of CF is that it does not consider the content of the items being recommended. Rather than map users to items through "content attributes" or "demographics," CF treats each item and user individually. Accordingly, it becomes possible to discover new items of interest simply because other people liked them; it is also easier to provide good recommendations even when the attributes of greatest interest to users are unknown or hidden. For example, many movie viewers may not want to see a particular actor or genre so much as "a movie that makes me feel good" or "a smart, funny movie." At the same time, CF's dependence on human ratings can be a significant drawback. For a CF system to work well, several users must evaluate each item; even then, new items cannot be recommended until some users have taken the time to evaluate them. These limitations, often referred to as the first-rater and sparsity problems, cause trouble for users seeking obscure movies (since nobody may have rated them) or advice

on movies about to be released (since nobody has had a chance to evaluate them), and not make use of its ratings.

The early-rater problem arises because a collaborative filtering system provides little or no value when a user is the first one in his neighborhood to enter a rating for an item. Current collaborative filtering systems depend on the altruism of a set of users who are willing to rate many items without receiving many recommendations. Economists have speculated that even if rating required no effort at all, many users would choose to delay considering items to wait for their neighbors to provide them with recommendations [6]. Without altruists, it might be necessary to institute payment mechanisms to encourage early ratings.

Another limitation, the sparsity problem, arises because the goal of collaborative filtering systems is to help people focus on reading documents (or consuming items) of interest. In high-quantity, low-quality environments, such as Usenet news, users may cover only a tiny percentage of documents available (Usenet studies have shown a rating rate of about 1% in some areas; we can estimate that few people will have read and formed an opinion on even 1/10 of 1% of the over two million books available through the largest bookstores). On the one hand, this sparsity is the motivation behind filtering: most people do not want to read most available information. On the other hand, sparsity poses a computational challenge as it becomes harder to find neighbors and harder to recommend documents since few people have rated most of them.

2.5.4 Hybrid profiling-based recommender systems

CF functions by identifying users with similar tastes and using their opinions (usually by asking them to rate the product on a predefined scale) to recommend items. But, CF systems suffer from the reliance of user ratings which make recommending new or obscure items very difficult. Ongoing research work such as the GroupLens Research Project [37] has successfully combined the two techniques to form hybrid recommender systems that have proven that they can make better recommendations than using either IR systems or CF systems alone.

Several other systems have also tried to combine information filtering and collaborative filtering techniques in an effort to overcome the limitations of each. Fab [8] maintains

user profiles of interest in web pages using information filtering techniques, but uses collaborative filtering techniques to identify profiles with similar tastes. It then can recommend documents across user profiles. [9] trained the Ripper machine learning system with a combination of content data and training data in an effort to produce better recommendations. Researchers working in collaborative filtering have proposed techniques for using IF profiles as a fall-back, e.g. by requesting predictions for a director or actor when there is no information on the specific movie, or by having dual systems and using the IF profile when the CF system cannot produce a high-quality recommendation. In earlier work, [63] showed that a simple but consistent rating agent, such as one that assesses the quality of spelling in a Usenet news article, could be a valuable participant in a collaborative filtering community. In that work, they showed how these filterbots - ratings robots that participate as members of a collaborative filtering system - helped users who agreed with them by providing more ratings upon which recommendations could be made. For users who did not agree with the filterbot, the CF framework would notice a low preference correlation and not make use of its ratings.

2.6 User interface approaches

Most websites today still use the metaphor of an “electronic catalog” which resembles an enhanced price list with search capabilities as the user interface. Even though these lists are searchable, it is still difficult for consumers to find a product that suit their needs when they have to literally browse through pages and pages of product information. This potentially tedious browsing experience obviously offers less engaging shopping experiences than their physical-store counterparts. Hence, it is reasonable to assume that greater customer satisfaction can be generated by matching the system’s user interface with the consumer’s manner of shopping.

To overcome this problem, some websites try to mimic the familiar physical storefront by constructing virtual shopping malls using VRML (Virtual Reality Markup Language) in the hope of providing a more familiar shopping experience. Although this approach is promising [3], these shopping environments have not yet lived up to their expectations due to the awkwardness of navigating 3D worlds with 2D interfaces and other technical limitations (e.g. bandwidth).

Another approach is the introduction of sales agent avatars - semi-animated graphical characters that interact in natural language with the consumer and feature a long-term consistent “personality” that remembers each customer, his or her shopping habits, etc. Anthropomorphized avatars (e.g. from Extempo [32]) attempt to mimic real-world sales agents to provide a more engaging online shopping experience and assist customers in finding the products that best meet their needs. Through immediate positive feedback and personalized attention, anthropomorphized sales agents can help build engaging, trusted relationships with customers [42]. However, the AI technologies behind the graphical representations of today’s avatars are not yet up to meeting their users’ expectations. Due to this and other reasons, the anthropomorphization of agents is still a controversial approach [4].

Interface agents help a user accomplish tasks by acting like a personal assistant. From user interactions, they are able to learn and adapt themselves to user preferences and work habits. Patti Maes [49] at MIT identifies four ways that learning can occur. First, an agent can learn by observing what the user does and imitating the user. Second, the agent can offer advice or take actions on the user’s behalf and then learn by receiving feedback from the user. Third, the agent can get explicit instructions from the user. Finally, by asking other agents for advice, an agent can learn from their experiences. An important point to note is that interface agents collaborate primarily with the user and not with other agents. Asking advice is the only exception. Using various learning techniques, interface agents can customize the user interface of a computer system or application for a particular user and her unique working style.

Additionally, some agents rely on the iterative process of browsing and user feedback via an intuitive user interface to make recommendations. For example, Apt Decision interface went through a number of iterations to make it more intuitive and responsive to the user’s actions. Adding the drag-and-drop feature was crucial to this effort. Apt Decision also takes an interactive learning approach, that is, it learns from each interaction with the user. Interactive learning makes the assumption that all the user’s actions have some meaning, and the agent is designed so that this is true. Each time the user drags an apartment feature to the profile, the reinforcement learning algorithm changes the weightings on the features in the user’s “ideal” apartment. This approach differs from traditional machine learning in several ways. First of all, it works with very

small, but precise, amounts of data. Also, it is an interactive technique, in that the user is in constant contact with the agent; there is no batch processing of datasets. Each feature of an apartment in Apt Decision has a base weight. Weights on individual features change when the user chooses to place them in or remove them from a profile slot. The new weight depends on which slot the feature occupies, whether the feature is crucial, and whether the slot was filled using profile expansion. Crucial features are weighted more heavily; features automatically added to the profile are weighted less heavily. In addition, Apt Decision records the history of a user's interaction with the agent. If at some point in the profile building process, there are suddenly no apartments that match the profile, the agent can offer the recourse of backtracking to a prior point in the interaction.

Other research involving interface agents include the BlueEyes project [13] at IBM Almaden Research Center features a camera that can figure out where a user is looking on the screen (gaze identification) to determine what article they are reading. Gesture recognition software allows computers to respond to waves of the hand, and even understand facial expressions. And, no surprise here, intelligent software forms the basis for these types of applications. Similarly, the COLLAGEN project at Lotus Research and Mitsubishi Research develops agents that can watch a user interact with an application and figure out the task that the user is trying to perform and give assistance [60]. The OpenSesame application on Macintosh watches a user, learns their behaviour, and offers to automate repetitive tasks [19].

2.7 Challenges

Unfortunately, the successes of these recommender systems have been restricted to simple consumer goods such as movies, books, clothing and food. When the IR and/or CF techniques plus other reinforcement learning methods are applied in online applications for complex consumer products such as real estate, vacation plans, insurance, mutual funds, and mortgage, they fail to enjoy much success. The proposed learning model incorporating the initial policy will solve the problems faced by current implementations of agent-mediated e-commerce systems for complex domains. The details will now be found in the next chapter.

Chapter 3

BETTER RECOMMENDATIONS WITH HUMANE

3.1 Summary

This chapter presents an introduction to HumanE – an online profiling agent for recommending real estate properties. We give a concise explanation on HumanE's design and explain our approach to accelerate agent learning with the provision of an initial policy and discuss some of the solutions taken to overcome existing problems in creating online profiling agents for complex multi-dimensional domains.

3.2 Problems in developing profiling agents for complex domains

Based on our discussion in earlier chapters, let us recap the problems encountered when developing online profiling agents for complex multi-dimensional domains:

- Assumption that the agent knows nothing and must acquire its knowledge through exploration and subsequent exploitation of learned knowledge results in slow agent learning for complex domains and makes online implementation difficult
- Difficult to give an agent large amount of application-specific and domain-specific knowledge
- Difficult to encode this knowledge in an abstract language
- Difficult to transfer agent knowledge and the control architecture for building agents for other applications
- Difficult to maintain the individual rules in the agent rule base over time
- Static agent knowledge (i.e. cannot be customized to individual user habits and preferences)
- Making "random" exploratory recommendations can frustrate and disappoint the user

- Difficult to allow for serendipitous discoveries of user preferences
- Difficult to obtain user trust when an interface agent is very sophisticated, qualified and autonomous from the start
- Too much data is required in an online setting for typical learning methods (e.g. reinforcement-learning methods)

3.3 Practical approach to building online profiling agents

We strongly believe that practical agent learning for online applications is possible by integration with human-supplied knowledge. This is because humans can provide a lot of help to assist agents in learning, even if humans cannot perform the task very well. Humans can provide some initial successful trajectories through the space. Trajectories are not used for supervised learning, but to guide the learning methods through useful parts of the search space leading to efficient exploration of the search space.

Online profiling agents can be bootstrapped from a human-supplied policy which basically gives some sample trajectories. The purpose of the policy is to generate "experiences" for the agents. This policy can be hand-coded by domain experts. It need not be optimal and may be very wrong. The policy shows the agents "interesting" parts of the search space. In fact, "bad" initial policies might be more effective.

In brief, this gives us a natural way to insert human knowledge and a simple method to bootstrap information into a utility function.

Our online profiling agent, HumanE, is based on the aforementioned approach and it offers users the opportunity to find products that will best meet their requirements. HumanE guides users through a product selection process. Users get to specify information about their individual requirements and restrictions by creating and refining their profiles.

Based upon the profile (and initial policy if the profile is newly created), HumanE offers an initial selection of products. Users can then select from these matching products to view more detailed product information such as product features. HumanE also tries to

be helpful by providing products that are newly added as well as products that are popular among other users.

To refine the profile, users can specify which features are desirable or undesirable through an intuitive and friendly interface, and HumanE will offer a new selection of products matching the revised profile. If no matching products are found, users can backtrack to their previous profile.

Furthermore, users can add an unlimited number of desired products to their profile using the “favourites” feature. Moreover, users can specify HumanE to send email alerts if there are any new products that fit the profile.

We discuss in greater detail the working of HumanE with regards to its learning approach and other features in later sections.

3.4 HumanE components

This section explains the main functionalities of the various components used by HumanE. A schematic diagram showing the main components of HumanE is shown in Figure 3.1.

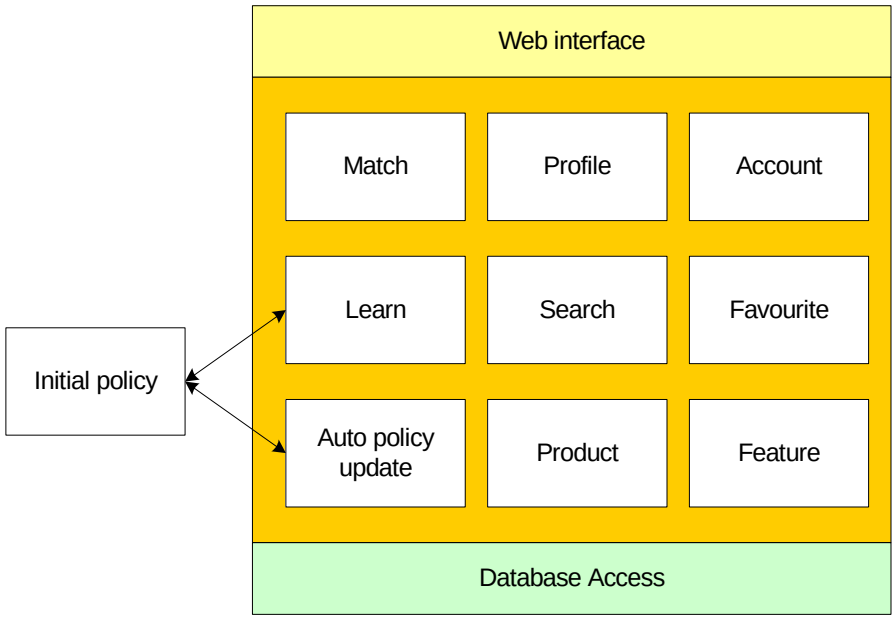

Figure 3.1 Main components of HumanE

3.4.1 Account component

This component provides user authorization, authentication and registration services. If the user is not a member, it allows the user to register as a member. The component provides the registration form and saves the details as provided by the user. After the registration is completed successfully, it returns some of the user's particulars such as name, address, and email address back to HumanE. If the user is already a member, then similarly some of the user's particulars such as name, address, and email address are sent back to HumanE after successful login. Furthermore, this component is used whenever the user makes any changes to the account.

3.4.2 Product component

This component manages the creation, modification, and deletion of products. And it provides parameter-based retrieval of product listings. HumanE makes extensive use of this component for the display of matching product listings or whenever the user requests for more information about a particular product.

3.4.3 Database component

This component provides the functionality for data access used by other components. All common database functions (i.e. reading data from database and populating the data read into a dataset) are consolidated in this component for ease of reusability.

3.4.4 Favourite component

This component handles all the work relating to the creation and modification of a "favourites" list. It is called whenever the user adds or removes a product from a "favourites" list.

3.4.5 Feature component

This component provides the functionality of parameter-based retrieval of feature lists in the form of name-value pairs and is used extensively by the Match Component and Profile Component.

3.4.6 Match component

This component looks for real estate products that best match the criteria provided by the user to HumanE. It uses the profile to return a list of real estate product information. And if the profile is newly created, it uses a combination of user profile and initial policy to

return a list of matching products. As the list of matching products returned is typically small, a filtering algorithm is used to ensure that only the most appropriate products are added to the list. The component also keeps track of the number of times a product is added to a "favourites" list and the number of times the detailed information of a product was been viewed. Additionally, it provides product retrieval based on pre-defined criteria such as popularity and "viewership".

3.4.7 Profile component

This component provides the user interface to allow the user to explicitly manipulate and save the resulting profile. It retrieves the existing profile from the database, provides the mechanism to allow the user to add new features or modify existing ones to the profile. It then saves the modified profile as a new profile under the same user ID. This allows for backtracking during the profile refinement process. In addition, the component contains the agent learning algorithm that allows HumanE to learn user preferences and to customize the profile accordingly. Other functions include deletion of profile and management of email alert.

3.4.8 Auto policy update component

This component provides HumanE with the ability to automatically maintain the initial policy based on the history of past user interactions. It is called on a periodic basis to update the knowledge encoded inside the initial policy.

3.5 Agent workflow

HumanE performs tasks based on a predefined workflow sequence as shown in Figure 3.2 below.

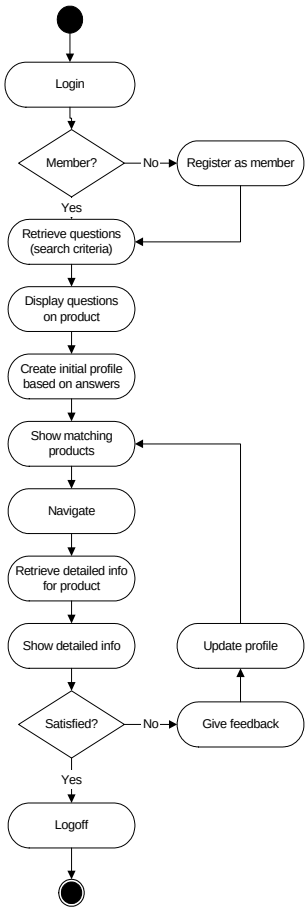

Figure 3.2 Agent workflow diagram

- The Account component is called for authorization and authentication.
- If the user is not a member yet, the Account component is called to present the registration form to the user.
- Upon successful registration, the Profile component is called in order to present the questions to the user. It receives the user's answers and generates a set of selection criteria from these answers.
- The Profile component is called to store this set of criteria as an initial profile.
- The initial profile is sent to the Match component and the component retrieves a list of matching products based on the initial profile and initial policy and displays the list.

- When the user decides to view more information about any product, the Product component is called and displays the relevant product details.

- The user is asked to provide feedback on the product shown by specifying the features that he likes best or dislikes most. The user can also rank the features selected using an intuitive interface.

- The Profile component is called to save the user's feedback as part of a new profile.

- The updated profile is sent to the Match component to obtain a new list of matching products.

- This iterative process of product browsing and user profile modification will take place until the user is satisfied with the profile.

- Finally, the Security component is called when the user logs off the system.

3.6 Component-based model

HumanE adopts the component-based software development model which enables reuse of core functionality within the application and across applications. In addition, HumanE uses a three-tier architecture (i.e. presentation, business logic and data access layers) and components-based model plays an important role in developing all three tiers. In order to ensure HumanE can be used successfully in other domains without major reworking, the components are designed to be as generic as possible; any profiling agent designed to find and recommend products can use them. HumanE is also generic in its design and its modularized architecture makes it easy to plug a different learning mechanism, for example, genetic algorithm or neural network into it. Although the HumanE database contains real estate data, it could also be populated with data on insurance plans, vacation plans, mutual funds, or any other complex product.

3.7 Design assumption

Even until today, users are still using the simple search function provided by many local online real estate web sites [66, 67] when browsing for real estate properties. There is no

interactivity between each search attempt and users are bombarded with endless pages of real estate properties listing which they will never be able to finish viewing.

Our design approach assumes that the entire user experience is an iterative process of browsing and meaningful user feedback. The approach has in fact been adopted successfully by similar systems such as RentMe [17, 18], CASA [34] and Apt Decision [65]. As the user is actively involved throughout the entire profile creation process, the user can react independently to every feature of the real estate offerings.

3.8 Domain analysis

To test the feasibility of the proposed learning model, we chose the real estate domain. As the agent needed to have built-in knowledge about the domain, we analyzed online and offline apartment advertisements to determine the standard apartment features for the local real estate domain. After the ad analysis, we had a list of about one hundred features commonly advertised in local real estate listings and we added another eighty features.

Next, we considered how people choose apartments. After examining the features, we concluded that some of them (e.g. district, type, price) were pivotal to the final choice of apartment. That is, most people would reject an apartment if the value for a crucial feature were not to their liking. Other features (e.g. bridge, underpass, swimming pool) were less pivotal – some people would like them, some would be indifferent, some would dislike them. All this domain knowledge went into HumanE.

In addition, we examined two destinations of apartment seekers: real estate websites and human real estate agents, to determine what knowledge we could glean from those interactions.

3.8.1 Real estate websites

Many real estate websites adopt either the pure browsing metaphor [67] or the search-like metaphor [66]. One problem is that users are expected to enter many specific details about their ideal apartment. Since buying apartment is a complex decision, people find it difficult to articulate what they really want initially. What they think they want may change in the course of their exploration of what is available; they may have firm

constraints or weak preferences; they may have unstated goals, such as finding something quickly, or determining how reliable the agent is.

Another problem is that they must enter their preferences when they visit a new site and each time they visit the site. This is because there is no option to save multiple sets of preferences for a single site. Especially with a complex decision such as renting an apartment, people find it difficult to specify exactly what it is that they want.

HumanE empowers the user to quickly and easily ascertain preferences via a profile as it represents salient features of the real estate domain. It removes the cognitive burden of questions such as: What can I expect of apartments in Jurong? What features are common and which are unusual? What is the range of price I can expect to pay for a certain neighbourhood? As a result, it allows the user to concentrate on questions not easily solved by technology, such as: Can I trust this broker? Can I get a better bargain?

3.8.2 Humane real estate agents

To improve HumanE's ability to increase online real estate experience, we consider how people deal with the ambiguity and imprecision of real world decisions.

For example, when a customer interacts with a real estate agent, the agent does not make the customer fill out a questionnaire containing all the possible attributes of apartments, then search a database to present the customer with all the choices that fit the questionnaire. Instead, the agent asks, "How may I help you?" and the customer is free to respond however he or she wishes.

Typically, the customer will supply a few criteria such as price range, apartment type and district: e.g. "I would like to buy a three-room apartment in Jurong East for about $140,000." These criteria provide a rough "first estimate" for the agent. All of the criteria might be lies; the customer might very well buy something that fits none of the initial criteria.

The real estate agent uses the initial guidelines to retrieve a few examples: "I've got a three-room apartment in Jurong East for $150,000 but there are no nearby shops. And how about this nice three-room apartment for $130,000 in Jurong West that has a great view?" The agent then waits to see the customer's reaction.

The key point is that the customer may react in a variety of ways not limited by answers to explicitly posed questions. The agent's description will typically contain many details not asked for originally by the customer. The success of the interaction is determined largely by the agent's ability to infer unstated requirements and preferences from the responses. "Let's see the one in Jurong East." lets the agent infer assent with the initial criteria, but "What about my car?" establishes a previously unstated requirement that the car park is a must.

Near-miss examples, such as "I've got a three-bedroom for $190,000, but it is in Ang Mo Kio", "Would you pay $170,000 if the apartment was in Yishun and near MRT?" establish whether the ostensible constraints are firm or flexible. Good agents are marked by their ability to converge quickly on a complicated set of constraints and priorities.

3.8.3 Transferring domain knowledge

Much of the work done for HumanE would transfer well into any domain in which the user could browse the features of a complex object. That is, objects such as calling plans, mutual funds, homes, computers, vacation plans, or cars would work well, but simple consumer goods such as clothing or food would not. Transferring the agent into another domain would require the services of a subject matter expert who could identify salient features of the complex objects in the domain, alter the program to work with those features and determine which features were crucial to the final decision. After testing on a suitable list of objects, the "new" agent could be released.

3.9 Learning approach

We have adopted a two-phase learning approach for HumanE. In the first phase of learning, HumanE learns by reinforcement, observation and actions taken that arise from a supplied initial policy. In the second phase, HumanE learns by reinforcement and observation. The initial policy is dynamic as it is updated without human intervention from the actions taken by the agent. The next chapter discusses in detail the proposed learning approach using illustrations of a running example.

3.10 User interface

We do not expect an average user to have a high degree of computer skills. Hence, we have paid extra attention to the design of the agent interface. We have made several changes to the HumanE agent interface in the hope that the interface will be intuitive and

responsive to users' actions. Since HumanE is a web agent, it interacts with the user via a Web browser such as Internet Explorer or Netscape. To better capture user preferences, we have tried a few approaches such as using a combination of web controls such as radio buttons (range indication as shown in Figure 3.7) or check boxes (specific indication as shown in Figure 3.8) to allow the user to specify his or her liking of a specific feature.

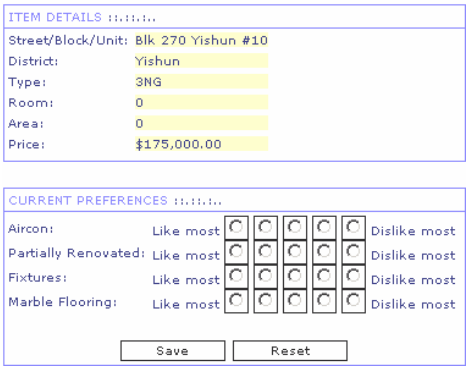

Figure 3.3 Earlier version of HumanE agent interface (range indication)

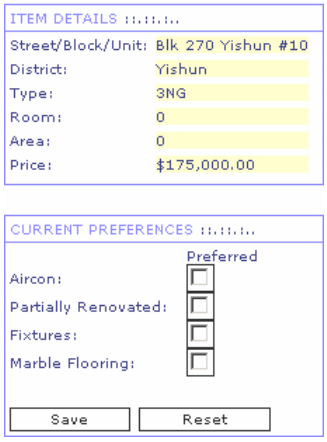

Figure 3.4 Earlier version of HumanE agent interface (specific indication)

After trying out a few approaches and gathering some useful user feedback, we decided to use two list boxes i.e. "desired" list box and "undesired" list box to allow the user to specify which feature is desirable or undesirable. Figure 3.9 shows the interface of the current version of HumanE.

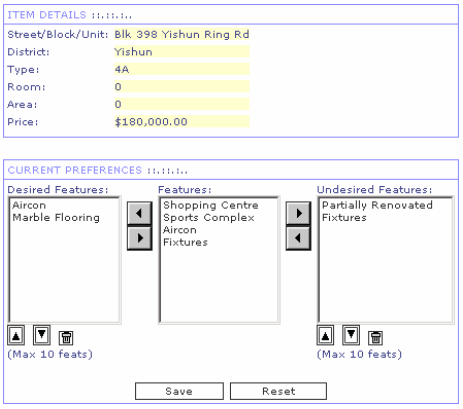

Figure 3.5 Current version of HumanE interface

The user clicks on the left arrow button to add a feature to the "desired" list box or clicks on the right arrow button to add a feature to the "undesired" list box. Furthermore, the user can rank the features in each list box in accordance to their liking. The features at the top of each list box correspond to those features well-liked most or dislike most. One potential limitation is that the interface cannot handle the situation where the user has no stated preference for features present in the two list boxes. We have decided against having a user-controlled function to turn off the ranking feature as it adds a certain amount of complexity to the learning algorithm. However, one positive observation we find is that the user is "compelled" to consider carefully their liking of the features as indicated in the list boxes and this may help to speed up the process of discovering unstated user preferences.

3.11 How HumanE works in real estate domain

Instead of letting the user browse through pages and pages of real estate listings, HumanE adopts the iterative process of browsing and user feedback. Through HumanE, the user is able to react independently to every feature of an apartment offering and not just the apartment itself. In addition, the user is able to participate in the entire profile creation process which gives him or her more flexibility in specifying the requirements. By soliciting feedback from the user through the critique of concrete examples, HumanE is able to infer user preferences and gives better recommendations. In the next chapter, we will explain in greater detail the proposed learning approach.

Chapter 4

LEARNING APPROACH

4.1 Summary

This chapter presents in detail the learning approach adopted by HumanE. We explain in greater detail our two-phase learning approach to accelerate agent learning with the provision of an initial policy using a running example. The discussion covers important aspects of the learning approach such as matching algorithm and reinforcement learning using a multidimensional utility function.

4.2 Introduction

The proposed two-phase learning approach has been tested successfully in past research on robotics [74]. Kaelbling et. al. found that robots using reinforcement learning learnt better when they were provided prior knowledge about their environment using a supplied initial policy. The policy generated example trajectories through the state-action space and showed the robot areas of high rewards and low rewards. After the robot had acquired a suitable amount of information through this initial phase of learning, the reinforcement learning system took control of the robot. Usually by this time, the robot had learned enough to make more informed exploration of the environment.

In this work, we adapt a similar approach when building a agent-based online real estate system. To do so, we consider each user decision as a trajectory in the search space much like the trajectories in the robot motion.

4.3 Initial profile vs initial policy

To minimize any confusion, we feel that it is important that we explain the difference between initial profile and initial policy.

Initial profile refers to the profile that is created at the very beginning of the learning approach. The initial profile contains only the user-defined preferred district, desired apartment type, and price.

Initial policy refers to the set of trajectory samples that show HumanE areas of high rewards and low rewards in the search space.

4.5 Constituents of a profile

The main objective of HumanE is to create a user profile (or simply called a profile) to store user preferences and to assist the user to refine his or her profile intelligently using the supplied learning approach. In our scenario, a profile stores both static and dynamic (learned) user preferences in the form of desired and undesired apartment features. Examples of apartment features include “high floor”, “near MRT”, “marble floor”, etc.

The constituents of a profile are listed in the table below.

Profile Field	**Field Description**
RowId	Refers to a unique identifier tagged to each profile for identification purpose.
District	Refers to a single-valued user-specified preferred apartment district (i.e. Bishan, Tampines, etc).
Type	Refers to a single-valued user-specified preferred apartment type (i.e. 3-room, 4-room, etc).
Price	Refers to a single-valued user-specified preferred apartment price (user budget).
DesiredFeatures	Refers to an ordered list of user-specified “desired” features. The list can store up to a maximum of five “desired” features. The first feature in the list represents the “most desirable” of the desired features and the last feature in the list represents the “least desirable” of the desired features. This list is updated every time when a user indicates his or her liking of a particular feature. See Figure 3.5.
UndesiredFeatures	Refers to an ordered list of user-specified “undesired” features. The list can store up to a maximum of five “undesired” features. The first feature in the list represents the “most undesirable” of the undesired features and the last feature in the list represents the “least undesirable” of the undesired features. This list is updated every time when a user indicates his or her disliking of a particular feature. See Figure 3.5.
ActualDesiredFeatures	Refers to an ordered list of “desired” features learned by HumanE.

	The list can store up to a maximum of five "desired" features. The first feature in the list represents the "most desirable" of the learned desirable features and the last feature in the list represents the "least desirable" of the learned desirable features. The difference between DesiredFeatures and ActualDesiredFeatures is that the value of ActualDesiredFeatures takes into account past user selections of "desired" features. Hence, in most cases, the two values differ.
ActualDesiredFeaturesScore	Refers to the score ranging from one to five assigned to every feature which appears in the ActualDesiredFeatures list. Five is the maximum score assigned to a feature as we only allow a maximum of five features in the DesiredFeatures list. One is the minimum score that can be assigned to a feature. This score is added to the existing score for the same feature found in the corresponding ActualDesiredFeatures list to obtain the total score.
ActualDesiredFeaturesFreq	Refers to a counter that stores the number of times a feature is present in the ActualDesiredFeatures list. It is automatically incremented whenever a feature is found in both DesiredFeatures and ActualDesiredFeatures lists or present in the DesiredFeatures list but missing in the ActualDesiredFeatures list. However, it is automatically decremented whenever a feature is missing in the DesiredFeatures list but present in the ActualDesiredFeatures list.
ActualDesiredFeaturesNetScore	Refers to the value obtained when we divide the total score (ActualDesiredFeaturesScore) by the total frequency (ActualDesiredFeaturesFreq) for each feature in the ActualDesiredFeatures list. If the net score is less that 1.0, the feature is removed from the ActualDesiredFeatures list.
ActualDesiredFeaturesRanked	Refers to a "replica" of the ActualDesiredFeatures list in which the features are sorted based on their net scores (ActualDesiredFeaturesNetScore) in

	ascending order. This list is used to assist in the various computations performed by HumanE.
ActualUndesiredFeatures	Refers to an ordered list of "undesired" features learned by HumanE. The list can store up to a maximum of five "undesired" features. The first feature in the list represents the "most undesirable" of the learned undesirable features and the last feature in the list represents the "least undesirable" of the learned undesirable features. The difference between DesiredFeatures and ActualDesiredFeatures is that the value of ActualDesiredFeatures takes into account past user selections of "desired" features. Hence, in most cases, the two values differ.
ActualUndesiredFeaturesScore	Refers to the score ranging from one to five assigned to every feature which appears in the ActualUndesiredFeatures list. Five is the maximum score assigned to a feature as we only allow a maximum of five features in the UndesiredFeatures list. One is the minimum score that can be assigned to a feature. This score is added to the existing score for the same feature found in the corresponding ActualUndesiredFeatures list to obtain the total score.
ActualUndesiredFeaturesFreq	Refers to a counter that stores the number of times a feature is present in the ActualUndesiredFeatures list. It is automatically incremented whenever a feature is found in both UndesiredFeatures and ActualUndesiredFeatures lists or present in the UndesiredFeatures list but missing in the ActualUndesiredFeatures list. However, it is automatically decremented whenever a feature is missing in the UndesiredFeatures list but present in the ActualUndesiredFeatures list.
ActualUndesiredFeaturesNetScore	Refers to the value obtained when we divide the total score (ActualUndesiredFeaturesScore) by the

	total frequency (ActualUndesiredFeaturesFreq) for each feature in the ActualUndesiredFeatures list. If the net score is less that 1.0, the feature is removed from the ActualUndesiredFeatures list.
ActualUndesiredFeaturesRanked	Refers to a "replica" of the ActualUndesiredFeatures list in which the features are sorted based on their net scores (ActualUndesiredFeaturesNetScore) in ascending order. This list is used to assist in the various computations performed by HumanE.

Table 4.1 Profile constituents

4.5 Overview of the learning approach

We have adopted a two-phase learning approach for HumanE. In the first phase of learning, HumanE learns by reinforcement, observation and takes actions that arise from a supplied initial policy. This mode of learning will last for one iteration of the profile refinement process (i.e. the iterative process of viewing apartments and selecting/ranking desired and undesired features). In the second phase, HumanE learns by reinforcement and observation. The content of the initial policy is dynamic as it is updated without human intervention from the actions taken by HumanE. Figure 4.1 depicts the workflow of the two-phase learning approach.

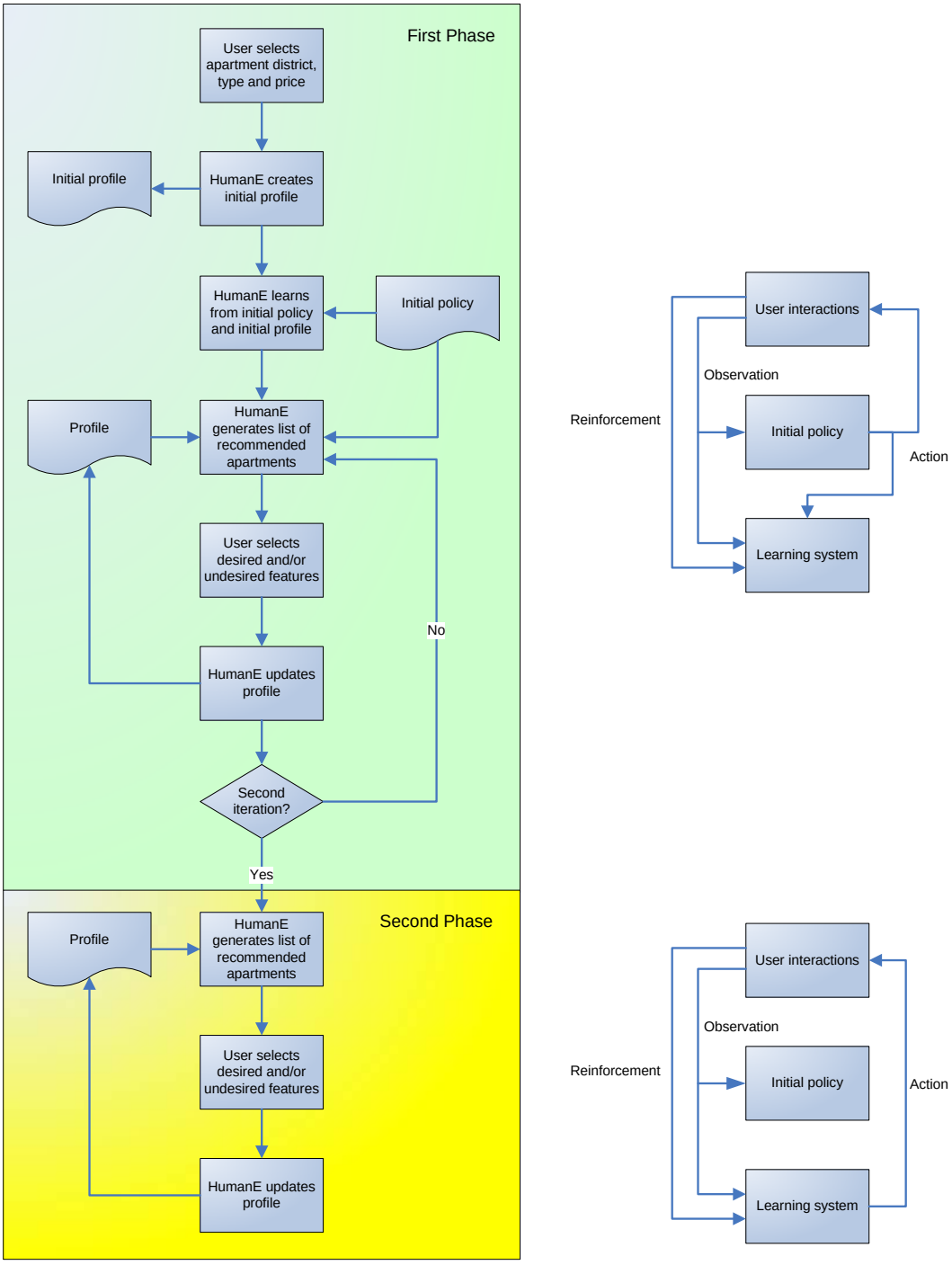

Figure 4.1 Workflow for the two-phase learning approach

4.6 Phase one learning

In phase one learning, HumanE initially learns and takes action from a supplied initial policy. This occurs right after when an initial profile is created. Using the information

stored within the supplied initial policy and initial profile, HumanE learns the locations of desired (high rewards) and undesired (low rewards) apartments in the search space that match the user preferences stored in the initial profile.

4.6.1 Learning from an initial policy

To emulate some of the inference power a human real estate agent might have, we have incorporated an initial policy to enhance the interactive learning ability of the agent. Basically, an initial policy is a XML file that stores information about which apartment features are generally considered as desirable and undesirable. In HumanE, the initial policy is stored in a file named "Bootstrap.xml". Figure 4.2 shows the content of this file.

```
<?xml version="1.0" encoding="utf-8" ?>
<features>
  <item id="desired" value="1,4,8,98"/>
  <item id="undesired" value="23,24,43"/>
</features>
<desired>
  <district>
    <item id="1" value="15,45,92,123,280,410,488,523,677,712"/>
    <item id="2" value="31,41,88,100,256,323,690,732,822,845"/>
    <item id="3" value="22,57,186,229,311,396,498,554,651,781"/>
    ……
  </district>
</desired>
<undesired>
  <district>
    <item id="1" value="23,34,52,166,232,359,390,416,509,682"/>
    <item id="2" value="27,55,75,98,167,293,531,567,734,802"/>
    <item id="3" value="16,56,78,139,266,345,482,573,680,744"/>
    ……
  </district>
</undesired>
```

Figure 4.2 Initial policy used in HumanE

The first piece of information encoded in the initial policy as shown above is the list of features generally considered as desirable and undesirable features. The attribute "id" denotes the attribute name and the attribute "value" refers to the value of the attribute "id". The second piece of information encoded is the list of top ten most popular apartments per district. The attribute "id" denotes the attribute name which in this case refers to the district id. The attribute "value" refers to the value of the attribute "id" i.e. the apartment id of the top ten most popular apartments per district. The third piece of information encoded is the list of top ten most unpopular apartments per district. The

attribute “id” denotes the attribute name which in this case refers to the district id. The attribute “value” refers to the value of the attribute “id” i.e. the apartment id of the top ten most unpopular apartments per district.

Coupling with the information stored in the initial profile, the initial policy will generate certain “interesting” trajectories through the search space, showing HumanE areas of high (popular apartments) and low (unpopular apartments) rewards. These trajectories and associated rewards are then used in this first, passive phase of learning.

As shown in Figure 4.1, the bootstrapping process occurs after the creation of the initial profile. Here, HumanE observes the states and rewards being generated and bootstraps this information into its “memory”. In our domain, the areas of high rewards are those apartments which have at least half of the “desired” features as specified in the initial policy. On the other hand, the areas of low rewards are those apartments which have at least half of the “undesired” features as specified in the initial policy. We have labeled each feature as either “desired” or “undesired” based on commonsense rules. In short, this corresponds to a real-life situation in which a human real estate agent always has in mind a small number of real estate properties that he or she knows that are popular or unpopular by virtue of the features they had.

The initial policy will show HumanE where the locations of potentially “desired” and “undesired” apartments based on the profile. For example, if the user has specified in the profile that the features desired are “MRT”, “Schools” and “High Floor”, then HumanE will search for matching apartments that have a combination of the following criteria:

1. All desired features as stated in the profile (“MRT”, “Schools” and “High Floor”).

2. More than half of the desired features as stated in the initial policy (e.g. “MRT”, “Bus Stop”, “Lift Level”, “Mid Floor”, “Good View”, and “Windy”).

3. Less than half of the undesired features as stated in the initial policy (e.g. “Playground”, “Rubbish Dump”, “Low Floor”, “Blocked View”, and “Facing Afternoon Sun”).

By combining the information from the initial profile and the initial policy, HumanE is able to generate a larger but potentially interesting set of matching apartments. This gives the user an opportunity to learn about apartments that do not exactly match the initial profile but may be of interest to him. In this way, HumanE allows for the serendipitous discoveries of new user preferences without the danger of random, unguided "exploratory" recommendations.

4.6.2 Reinforcement learning using a multidimensional utility function

After HumanE has generated a list of recommended apartments, it adopts reinforcement learning as the next learning technique to learn user preferences. It learns a multidimensional utility function on states (or histories) and uses it to select actions that maximize the expected utility of their outcomes. The reinforcement learning approach is used through the entire profile refinement process.

As mentioned in Section 4.5, every profile stores a set of "desired" (DesiredFeatures) and "undesired" (UndesiredFeatures) feature lists and a set of aggregated "desired" (ActualDesiredFeatures) and "undesired" (ActualUndesiredFeatures) feature lists. The DesiredFeatures and UndesiredFeatures lists store exactly the current user preferences as specified by the user when he or she makes changes to the profile. This information is required as HumanE needs to repopulate the profile information whenever the user wishes to update the profile.

However, the ActualDesiredFeatures and ActualUndesiredFeatures lists store the aggregated user preferences for desired and undesired features. They store the history of user interactions with regards to profile creation. The utility function uses the total scores (ActualDesiredFeaturesScore or ActualUndesiredFeaturesScore), total frequencies (ActualDesiredFeaturesFreq or ActualUndesiredFeaturesFreq) and net scores (ActualDesiredFeaturesNetScore or ActualUndesiredFeaturesNetScore) to generate these two lists.

In this way, the multidimensional utility function is able to capture past profile changes (i.e. the agent remembers history information) and incorporate the knowledge learned into a simple representation to be used by the matching algorithm.

4.6.3 Learning by observation

To augment the serendipitous discoveries of apartments which can be of potential interest to the user, we have implemented the "favourites" and "views" functions. First, the user can specify an apartment to be added to a "favourites" list for a particular profile. When this happens, HumanE increments the value of the "FavouriteNum" column in the "Apartment" database table of this particular apartment. The "FavouriteNum" column stores the number of times a particular apartment has been added to some "favourites" lists. Second, the user can select an apartment to develop the profile or simply to view more details about it. Similarly, when this happens, HumanE increments the value of the "ViewNum" column in the "Apartment" database table of this particular apartment.

As part of the matching process, HumanE selects the top ten apartments which have the greatest value for both the "FavouriteNum" column and "ViewNum" column and display the resulting apartments in the "Top Ten Most Popular Apartments" and "Top Ten Most Viewable Apartments" sections of the web interface. This encourages the user to make more serendipitous discoveries of apartments which the user may be interested in. The assumption taken here is that there is a high possibility that a typical user may be interested in apartments which are generally considered by other users to be "good".

4.6.4 Matching algorithm

HumanE employs a matching algorithm that is based on the concept of property relaxation. It uses the following rules are shown below in the order of execution when searching for matching apartments:

1. Search based on the exact specifications as stated in the current profile

2. Search using ActualDesiredFeatures and ActualUndesiredFeatures attributes instead of DesiredFeatures and UndesiredFeatures attributes

3. Search neighbouring districts based on the exact specifications as stated in the current profile

4. Search neighbouring districts using ActualDesiredFeatures and ActualUndesiredFeatures attributes instead of DesiredFeatures and UndesiredFeatures attributes

5. Search but ignore apartment type

6. Search from neighbouring districts but ignore apartment type

7. Search but relax on price restriction

8. Search from neighbouring districts but relax on price restriction

9. If no matching apartments can be found, HumanE displays the apartments listed in the "top ten most popular apartments per district" information contained in the initial policy for the district specified in the profile.

We have specified that only twenty matching apartments can be retrieved and shown to the user. This is to prevent the user from being overwhelmed if many apartments are to be shown to him or her. If less than twenty apartments are found using the first rule, then HumanE uses the subsequent rules to search for more matching apartments until there are twenty apartments selected.

Rule 5 and 6 use a relaxed price restriction when searching for matching apartments. Here, we consider price as an upper bound with a slight deviation of ±10%, meaning apartments having price that is within 0.90 and 1.10 times of the price specified by the user or less are returned.

Every apartment offering stores information about its neighbouring districts. If no apartments are found in the district as specified in the profile, HumanE will return apartments in the neighboring districts. This is used in Rule 2, 4 and 6.

4.7 Phase two learning

The purpose of having the initial policy in phase one learning is simply to generate experiences of the world which is tantamount to incorporating prior knowledge into HumanE. After a suitable amount of information has been acquired in the bootstrapping process, the second phase of learning takes over where HumanE learns primarily using

reinforcement learning and learning by observation. Usually by this time, HumanE is more "knowledgeable" which allows for more informed exploration of the search space.

4.8 Example of the profile refinement process

We present a simple example to further illustrate how the two-phase learning approach functions in the profile refinement process. Beginning with the creation of the initial profile, we trace the various steps that HumanE take to learn the user preferences and make better recommendations.

4.8.1 1st Iteration: Creating the initial profile

Let us assume that John (a fictitious user) is looking for a three-room apartment in the Ang Mo Kio district and is willing to pay about S$200,000. He enters these preferences into the web interface provided by HumanE to create the initial profile. Figure 4.3 shows the content of the initial profile.

Field name	Value
District	Ang Mo Kio
Type	3NG
Price	200,000
DesiredFeatures	
UndesiredFeatures	
ActualDesiredFeatures	
ActualDesiredFeaturesScore	
ActualDesiredFeaturesFreq	
ActualDesiredFeaturesNetScore	
ActualDesiredFeaturesRanked	
ActualUndesiredFeatures	
ActualUndesiredFeaturesScore	
ActualUndesiredFeaturesFreq	
ActualUndesiredFeaturesNetScore	
ActualUndesiredFeaturesRanked	

Figure 4.3 Initial profile

4.8.2 1st Iteration: Bootstrapping using the initial policy

Since this is the first iteration of the profile refinement process, HumanE bootstraps the initial policy into its "memory". This takes place right after the creation of the initial profile and before HumanE recommends any apartments to John. For the sake of illustration, let us assume that the content of the initial policy is as shown in Figure 4.4.

Field name	Value
Desired features	MRT, Bus Terminal, Central, Well Renovated
Undesired features	Industrial Estate, Port, Cemetery
Desired apartment ids (for Ang Mo Kio district)	15,45,92,123,280,410,488,523,677,712
Undesired apartment ids (for Ang Mo Kio district)	23,34,52,166,232,359,390,416,509,682

Figure 4.4 Initial policy

4.8.3 1st Iteration: Making the first recommendation

After the bootstrapping process has completed, HumanE has some initial knowledge about John's preferences (from initial profile) and some of the popular and unpopular apartments in Ang Mo Kio (from initial policy). To generate the twenty apartments for recommendations, HumanE first tries to locate popular apartments from the initial policy that matches the user preferences. If matching apartments are found, they are selected for display later.

Typically, less than twenty apartments (and sometimes no apartments) are found using this method and the remaining apartments are located using the matching algorithm as described in Section 4.6.4. During the execution of the matching algorithm, HumanE continuously checks that any apartment selected is not found within the list of top most unpopular apartments in the Ang Mo Kio district. And if it is found, then the matching algorithm discards the selected apartment and selects the nearest matching apartment. Once twenty apartments are selected, HumanE displays the selection to John for his consideration. HumanE also displays other information such as the Top Ten Most Popular Apartments (for all districts) and Top Ten Most Viewable Apartments (for all districts) to create the opportunity for John to discover and consider other potentially interesting apartments.

4.8.4 2nd Iteration: Making the first feature selection

Suppose John examines one of the apartments from the list and selects some desired and undesired features using the user interface provided (see Figure 3.5). Let us assume that the desired features selected are "MRT" and "Market" and the undesired features selected are "Community Club" and "School" in the order as shown, meaning "MRT" is more preferred than "Market" and "Community Club" is less preferred than "School".

Next, we explain how HumanE computes the values of each of the profile fields.

Field name	DesiredFeatures
Value	MRT, Market
Computation Description	No computation is needed. Its value is taken directly from the web interface and includes ranking information i.e. “MRT” is more desired than “Market”.

Field name	UndesiredFeatures
Value	Community Club, School
Computation Description	No computation is needed. Its value is taken directly from the web interface and includes ranking information i.e. “Community Club” is more undesired than “School”.

Field name	ActualDesiredFeatures
Value	MRT, Market
Computation Description	If ActualDesiredFeatures is null, then its value is equal to the DesiredFeatures.

Field name	ActualDesiredFeaturesScore
Value	5,4
Computation Description	If ActualDesiredFeaturesScore is null, then its value is equal to the position occupied by each feature in ActualDesiredFeatures. For example, “MRT” (in ActualDesiredFeatures) is given a score of five as it occupies the first position in ActualDesiredFeatures. “Market” is given a score of four as it occupies the second position.

Field name	ActualDesiredFeaturesFreq
Value	1,1
Computation Description	Since both “MRT” and “Market” appear for the first time in ActualDesiredFeatures, their frequency is one.

Field name	ActualDesiredFeaturesNetScore
Value	5,4
Computation Description	The net score of “MRT” is calculated using the formula below: ActualDesiredFeaturesNetScore = ActualDesiredFeaturesScore / ActualDesiredFeaturesFreq. For example, the net score for “MRT” is 5 / 1 = 5 and the net score for “Market” is 4 / 1 = 4.

Field name	ActualDesiredFeaturesRanked
Value	Market, MRT
Computation Description	This value is derived when ActualDesiredFeatures is sorted against ActualDesiredFeaturesNetScore in ascending order.

Field name	ActualUndesiredFeatures
Value	Community Club, School
Computation Description	If ActualUndesiredFeatures is null, then its value is equal to the UndesiredFeatures.

Field name	ActualUndesiredFeaturesScore
Value	5,4
Computation Description	If ActualUndesiredFeaturesScore is null, then its value is equal to the position occupied by each feature in ActualUndesiredFeatures. For example, "Community Club" (in ActualUndesiredFeatures) is given a score of five as it occupies the first position in ActualUndesiredFeatures. "School" is given a score of four as it occupies the second position.

Field name	ActualUndesiredFeaturesFreq
Value	1,1
Computation Description	Since both "Community Club" and "School" appear for the first time in ActualUndesiredFeatures, their frequency is one.

Field name	ActualUndesiredFeaturesNetScore
Value	5,4
Computation Description	The net score of "Community Club" is calculated using the formula below: ActualUndesiredFeaturesNetScore = ActualUndesiredFeatures Score / ActualUndesiredFeaturesFreq. For example, the net score for "Community Club" is 5 / 1 = 5 and for "School" is 4 / 1 = 4.

Field name	ActualUndesiredFeaturesRanked
Value	School, Community Club
Computation Description	This value is derived when ActualUndesiredFeatures is sorted against ActualUndesiredFeaturesNetScore in ascending order.

Field Name	Value
District	Ang Mo Kio
Type	3NG
Price	200,000
DesiredFeatures	MRT, Market
UndesiredFeatures	Community Club, School
ActualDesiredFeatures	MRT, Market
ActualDesiredFeaturesScore	5,4
ActualDesiredFeaturesFreq	1,1
ActualDesiredFeaturesNetScore	5,4
ActualDesiredFeaturesRanked	Market, MRT
ActualUndesiredFeatures	Community Club, School
ActualUndesiredFeaturesScore	5,4
ActualUndesiredFeaturesFreq	1,1
ActualUndesiredFeaturesNetScore	5,4
ActualUndesiredFeaturesRanked	School, Community Club

Figure 4.5 Profile after first feature selection

4.8.5 3rd Iteration: Making the second feature selection

After John has made the first feature selection, HumanE selects another set of apartments for John to consider. Suppose John examines one of the apartments from the list and selects again the desired and/or undesired features. Let us assume that the desired feature selected is "High Floor" and the undesired feature selected is "Next to Corner Unit". He also re-orders the features in the DesiredFeatures list by moving "High Floor" to the top of the list. In addition, he moves "School" to the top of the UndesiredFeature list and removes "Community Club" from it.

We explain how HumanE computes the new values of each of the profile fields below:

Field name	DesiredFeatures
Value	High Floor, MRT, Market
Computation Description	No computation is needed. Its value is taken directly from the web interface and includes ranking information i.e. "High Floor" is more desired than "MRT" and "MRT" is more desired than "Market".

Field name	UndesiredFeatures
Value	School, Next to Corner Unit
Computation Description	No computation is needed. Its value is taken directly from the web interface and includes ranking information i.e. "School" is more undesired than "Next to Corner Unit".

Field name	ActualDesiredFeatures
Value	MRT, Market, High Floor
Computation Description	Any new feature is automatically added to the tail of the ActualDesiredFeatures list. Hence, the new feature "High Floor" is added to the tail of the list even though it appears at the head of the DesiredFeatures list.

Field name	ActualDesiredFeaturesScore
Value	9,7,5
Computation Description	If ActualDesiredFeaturesScore is not null, then the score = current score + previous score. The current score is calculated using the following formula: Current score = maximum number of features in DesiredFeatures list – individual feature's current position on DesiredFeatures list. <u>Score for "MRT"</u> 5 (maximum number of features in DesiredFeatures list) – 1 (second position of the DesiredFeatures list) + 5 (previous score) ----- 9

	Score for "Market" 5 (maximum number of features in DesiredFeatures list) – 2 (third position of the DesiredFeatures list) + 4 (previous score) ----- 7 Score for "High Floor" 5 (maximum number of features in DesiredFeatures list) – 0 (first position of the DesiredFeatures list) + 0 (previous score) ----- 5

Field name	ActualDesiredFeaturesFreq
Value	2,2,1
Computation Description	If current frequency is not null, then the new frequency is calculated based on the following formula: New frequency = Previous frequency + 1 Frequency for "MRT" = 1 + 1 = 2 (second occurrence) Frequency for "Market" = 1 + 1 = 2 (second occurrence) Frequency for "High Floor" = 1 (first occurrence)

Field name	ActualDesiredFeaturesNetScore
Value	4.5,3.5,5.0
Computation Description	The net score is calculated using the formula below: ActualDesiredFeaturesNetScore = ActualDesiredFeaturesScore / ActualDesiredFeaturesFreq. Net score for "MRT" = 9 / 2 = 4.5 Net score for "Market" = 7 / 2 = 3.5 Net score for "High Floor" = 5 / 1 = 5.0

Field name	ActualDesiredFeaturesRanked
Value	High Floor, MRT, Market
Computation Description	This value is derived when ActualDesiredFeatures is sorted against ActualDesiredFeaturesNetScore in ascending order.

Field name	ActualUndesiredFeatures
Value	Community Club, School, Next to Corner Unit
Computation Description	Any new feature is automatically added to the tail of the ActualUndesiredFeatures list. Hence, the new feature "Next to Corner Unit" is added to the tail of the list. For the sake of illustration, "Community Club" is still reflected in the value shown above as its net score is less than 1.0. It will be removed eventually as shown in Figure 4.6. See the calculation for ActualUndesiredFeaturesNetScore for more details.

Field name	ActualUndesiredFeaturesScore
Value	0,9,4
Computation Description	If a feature is removed from the UndesiredFeatures list, the feature is given a current score of five. In this case, the new score is calculated using this formula: new score = previous score – current score. <u>Score for "Community Club"</u> 5 (previous score) - 5 (current score) ----- 0 <u>Score for "School"</u> 5 (maximum number of features in UndesiredFeatures list) – 0 (first position of the UndesiredFeatures list) + 4 (previous score) ----- 9 <u>Score for "Next to Corner Unit"</u> 5 (maximum number of features in UndesiredFeatures list) – 1 (second position of the UndesiredFeatures list) + 0 (previous score) ----- 4

Field name	ActualUndesiredFeaturesFreq
Value	2,2,1
Computation Description	If current frequency is not null, then the new frequency is calculated based on the following formula: New frequency = Previous frequency + 1 Frequency of "Community Club" = 1 + 1 = 2 Frequency of "School" = 1 + 1 = 2 Frequency of "Next to Corner Unit" = 1

Field name	ActualUndesiredFeaturesNetScore
Value	4.5,4
Computation Description	If the net score of a feature is less than 1.0, it is removed from the ActualUndesiredFeatures list. The net score is calculated using the formula below: ActualUndesiredFeaturesNetScore = ActualUndesiredFeaturesScore / ActualUndesiredFeaturesFreq Net score of "Community Club" = 0 / 2 = 0 (removed) Net score of "School" = 9 / 2 = 4.5 Net score of "Next to Corner Unit" = 4 / 1 = 4.0

Field name	ActualUndesiredFeaturesRanked
Value	Next to Corner Unit, School
Computation Description	This value is derived when ActualUndesiredFeatures is sorted against ActualUndesiredFeaturesNetScore in ascending order.

Field Name	Value
District	Ang Mo Kio
Type	3NG
Price	200,000
DesiredFeatures	High Floor, MRT, Market
UndesiredFeatures	School, Next to Corner Unit
ActualDesiredFeatures	MRT, Market, High Floor
ActualDesiredFeaturesScore	9,7,5
ActualDesiredFeaturesFreq	2,2,1
ActualDesiredFeaturesNetScore	4.5,3.5,5.0
ActualDesiredFeaturesRanked	High Floor, MRT, Market
ActualUndesiredFeatures	School, Next to Corner Unit
ActualUndesiredFeaturesScore	9,4
ActualUndesiredFeaturesFreq	2,1
ActualUndesiredFeaturesNetScore	4.5,4.0
ActualUndesiredFeaturesRanked	Next to Corner Unit, School

Figure 4.6 Profile after second feature selection

Figure 4.5 shows the profile after second feature selection with "Community Club" feature removed from the corresponding profile fields as its net score is less than 1.0.

4.8.6 Summary

The preceding example shows in detail how HumanE refines the profile with each user interaction. During the first iteration, HumanE makes use of the initial policy to make the apartment recommendations (first phase) and after which HumanE relies primarily on reinforcement learning and learning by observation to make subsequent apartment recommendations. The longer each feature appears or is ranked higher in the ActualDesiredFeatures or ActualUndesiredFeatures list, the higher will be its net score. This implies that HumanE will make greater use of this feature for making recommendations. Similarly, every time when a feature is removed or is ranked lower from either one of the lists, its net score will be lower, meaning that HumanE will make less use of this feature for making recommendations. In this way, the ActualDesiredFeatures and ActualUndesiredFeatures lists store the history of the user's past and present preferences. Hence, even though a user may remove a feature selected

previously, HumanE will still consider the feature during the apartment selection process as long as it is still present in the ActualDesiredFeatures or ActualUndesiredFeatures list. Eventually, when the net score of a feature falls below 1.0, the feature is removed from the list signaling the end of this feature's influence during the apartment selection process.

4.9 Crafting an initial policy

Typically, a domain expert will craft the initial policy during the initial system setup and after which the maintenance of the initial policy is handled by HumanE automatically. HumanE will basically rank each apartment feature in descending order based on aggregated feature data obtained from apartments that have been added to some "favourites" lists. The top three to five features will be taken as the new "desired" features and replaced the existing ones in the initial policy. HumanE will also rank each apartment feature found in the "UndesiredFeatures" part of each profile in terms of the frequency of appearance in descending order. And HumanE will take the bottom three to five features as the new "undesired" features to replace the existing ones in the initial policy.

More complex knowledge can be encoded within the initial policy as it is based on XML which supports hierarchical data structures. For instance, it is possible to specify which features are loosely related. For example, "MRT", "LRT", and "Bus Stop" are features that are loosely related in the sense that they both relate to public transport. If the user selects "MRT" feature as desirable, then HumanE can infer that the user may also prefer "LRT" feature or even "Bus Stop" feature. Since the user has indicated the "High Floor" feature as desirable, HumanE will similarly infer that apartments with "Mid Floor", "Good View", "Windy" features could be of interest to the user too. This is because the "High Floor" feature is loosely associated with both the "Good View" and "Windy" features.

4.10 Benefits of proposed learning approach

One of the main reasons why many reinforcement learning implementations fail to achieve much success for complex goods is that it is assumed that the agent developed knows nothing about the environment to begin with and that the agent must gain all of its information by exploration and subsequent exploitation of learned knowledge. When

dealing with a real, complex online system such as a large-scale real estate listing and brokerage application, however, this approach is simply not practical. Typically, the search space is too large to explore satisfactorily within the lifetime of the agent (much less within the attention time-span of typical online users). Worse still, making "random" exploratory recommendations can frustrate and disappoint the user, potentially causing the user to abandon the system totally.

For example, Apt Decision [65] suffers from the possibility that the user may get frustrated and disappointed if no suitable recommendations are found during the initial use of the system. This scenario can result as the Apt Decision agent has no prior knowledge about the real estate domain and cannot make good recommendations initially. Moreover, the agent needs time to learn the user's preferences from scratch and the time taken could be significantly long enough to cause the user to give up on the agent.

Another example is the SurfJet Agent [69] which is an intelligent assistant (much like HumanE) that acts as an autonomous learning agent. It is non web-based and uses an interest profile to perform searches on the Internet for articles on the user's behalf. SurfJet is able to make more accurate and useful searches as compared to traditional searching techniques as the user can give it a profile describing many of his or her interests, including how interesting (or uninteresting) each item is and how they relate to each other. However, SurfJet does not store any prior knowledge and rely solely on the iterative process of user feedback and profile refinement to make increasing accurate recommendations. Making good recommendations in the early stages of learning could be difficult and, like Apt Decision, a considerable amount of time may be spent to train SurfJet to understand a user's stated and unstated interests. It is likely that many users may not be prepared to commit that kind of time and effort to train the agent until it is sufficiently capable of making fairly good recommendations.

Accumulated knowledge in the form of memories and experiences allows humans to go about performing daily tasks. In the real world, we often go to a human real estate agent for assistance in selling or acquiring real estate properties. We naturally expect the agent to be an expert in the real estate domain, and hence able to offer suitable advice and

recommendations. Certainly, we do not expect the real estate agent to have no knowledge about the real estate domain.

Hence, in order to take our prior knowledge (which are often implicit) and incorporate them into a reinforcement learning framework, we have examined the idea of supplying HumanE with an initial policy about the real estate domain and in this chapter, we have described in detail the learning approach which we are confident that it can aid profiling agents in making better recommendations faster with the ultimate aim of soliciting greater satisfaction, confidence and trust from users. We will support our claims using experimental results and the details can be found in the next chapter.

Chapter 5

EXPERIMENTAL ANALYSIS

5.1 Summary

In this chapter, we evaluate the effectiveness of HumanE in contributing to customer satisfaction using various experiments and real-life data.

5.2 Methodology

The following sections outline the methodology used for the experiments conducted with HumanE and our testers.

5.2.1 Metrics

There are four dimensions to measure HumanE's ability to increase customer satisfaction:

- Number of profile changes
    - This refers to the overall number of times that a user has modified the profile since the profile was created and until the time when the user is satisfied with the profile and decides to print out a hard copy of the "favourites" list.
- Time taken to create a profile
    - This refers to the overall time taken by the user starting from creating a profile to printing out a hard copy of the "favourites" list when the user is satisfied.
- Ease of use
    - This refers to the perceived level of user-friendliness when using HumanE with regards to the ease of developing a user profile using the interface provided.

- Performance

    - This refers to the perceived level of performance when using HumanE in terms of the quality of matching apartments returned and the time taken to return the matching apartments.

HumanE's ability to handle large databases was tested using the scalability metric.

- Scalability

    - This refers to HumanE's ability to handle large databases in terms of returning matching apartments within a reasonable period of time.

5.2.2 Test data

To ensure the realistic nature of the experiments to be conducted, we painstakingly created our test data set from more than eight hundred actual real estate ad postings from both offline and online sources.

To ascertain whether HumanE can scale when the database grows, we also wrote a simple program to fabricate another test data set of about sixty-five thousand records.

5.3 Experimental design

Basically, we want to test whether our proposed learning approach with the use of initial policy contributes to better performance for web profiling agents. Based on the findings in [44, 54], we decided to use survey research in our experimental design for the following reasons:

- Surveys are easy to administer.

- Surveys are simple to score and code.

- Surveys determine the values and relations of variables and constructs.

- Responses can be generalized to other members of the population studied and often to other similar populations.

- Surveys can be reused easily, and provide an objective way of comparing responses over different groups, times, and places.

- Surveys can be used to predict behavior.

- Specific theoretical propositions can be tested in an objective fashion.

- Surveys can help confirm and quantify the findings of qualitative research.

In addition, our experimental design is also strongly influenced by the findings from [31, 58] especially in the area of web page evaluation.

We invited one hundred and fifty genuine real estate buyers to evaluate HumanE based on the chosen metrics. Figure 5.1 and 5.2 shows the cross-section profiles of the testers in terms of age and occupation respectively.

Age	20-29	30-39	40-49	50-59	60-69
No. of testers	33	45	42	24	6

Table 5.1 Cross-section profiles of the testers in terms of age

Occupation	Business	Finance	Govt	Healthcare	IT	Retail
No. of testers	24	30	24	18	39	15

Table 5.2 Cross-section profiles of the testers in terms of occupation

Most testers were of the age between late twenties to fifty years old. This coincided with the age range where most people would consider buying apartments and would genuinely be interested in using HumanE as an intelligent assistant during the searching and selection of apartments.

We also took into consideration the occupational profiles of the testers. We wanted to avoid having many IT professionals as our testers and they would naturally be more IT-savvy and might be inclined to rate HumanE more favourably due to its sophisticated mechanics.

Another factor we considered was the size of the test groups. The size of each test group should be sufficiently large to allow for more precision in the analysis of the test results.

On the other hand, we did not want the test groups to be overly large as the returns in terms of the accuracy of the test results could be diminishing as the test group size grew.

Based on HDB Annual Report for FY 2002-03 [40], there were about fifty thousand people who either applied for new HDB apartments or registered for resale flats. Hence, the population size was taken as 50,000. The confidence level is fixed at 95% to get high confidence in our findings and the confidence interval is fixed at a low 8 due to the large population size. Using the sample size calculator found at [22], the recommended sample size was 150 and thus we invited one hundred and fifty people to be our testers.

The evaluation process consisted of the following three tests:

- First test: Test HumanE **without** learning approach
- Second test: Test HumanE with learning approach **without** initial policy
- Third test: Test HumanE with learning approach **with** initial policy

We assigned fifty different testers to each test. We could not repeat the three tests for the same group of testers as they might be influenced by the earlier test data. To obtain consistent feedbacks from the three groups of testers, we gave the testers some guidelines to follow when giving answers. For example during the measurement of the "ease of use" metric and "performance" metric, we instructed the testers to give their answers based on the following definitions:

Scale	Very Bad	Bad	Neutral	Good	Excellent
Number of times a tester requests for help or asks questions on the use of HumanE	>10	10 - 8	7 - 6	5 - 3	2 - 0

Table 5.3 Scale definitions for "ease of use" metric

Scale	Very Bad	Bad	Neutral	Good	Excellent
Time taken to return matching apartments (sec)	>60	60 - 30	29 - 21	20 - 11	10 - 0

Table 5.4 Scale definitions for “performance” metric

Before the actual evaluation took place, we gave the testers a quick introduction on how to use HumanE. To ensure the objectiveness of the testers’ assessments, we chose not to be directly involved throughout the evaluation process (except the scalability test). Instead, a test coordinator with adequate knowledge of HumanE was asked to conduct the experiment.

Typically, the main method to show that a variable affects the outcome is to hold all other variables constant while changing only that variable. Preferably the experiment should be conducted in such a way that the users do not know the state of the variable so that they are unable to help the result even if they want to. Thus to ensure the accuracy of the test results, an identical interface is used to test HumanE 1) without, 2) with the learning approach (excludes initial policy) and 3) with the learning approach (includes initial policy) while the user is not informed of whether HumanE is learning or not.

The objective of each test is to allow the tester to arrive at a satisfactory profile. Each tester was asked to select his or her desired apartments using HumanE’s web interface. The test was considered completed when the user declared that he or she was satisfied with the list of desired apartments stored in the “favourites” list. Subsequently, the user was allowed to keep the profile created by printing out a hard copy of the “favourites” list. In each of the three tests, the user was not told whether HumanE was used in helping him or her develop the profile.

The user went through the entire profile creation process without much intervention from the test coordinator. The only assistance that was provided by the test coordinator was to clarify some questions asked by a few users pertaining to navigation and program operation.

At the end of each test, the values of the four metrics were recorded. For the “ease of use” and “performance” metrics, each tester was asked to rate them from a scale of one to five (i.e. 1: Very Bad, 2: Bad, 3: Neutral, 4: Good, 5: Excellent) for the three tests. For each test, the values for the “time taken to create a profile” and “number of profile changes” metrics were recorded and computed by HumanE. Since HumanE recorded the login time and logoff time for each tester, HumanE was able to compute the value for the

"time taken to create a profile" metric by subtracting the logoff time from the login time. And because every profile modification was recorded, HumanE was able to provide the value of the "number of profile changes" metric for each tester.

Additionally, we tested the scalability of HumanE in terms of the time taken to retrieve matching apartments from large databases. We ran several tests involving different database sizes ranging from 20,000 to 100,000 mock records and recorded the average time taken for each display of the matching apartment list. One unit of Intel Pentium 4 2.4GHz machine with 1 GB of memory was used for this test. Internet Information Server 5.1, .NET Framework SDK 1.1 and SQL Server 2000 were installed on this test machine.

5.4 Experimental results

The results from the experiments conducted are tabulated in the following tables.

5.4.1 First test: Test HumanE without learning approach

Frequency	1-5	6-10	11-15	16-20	21-25
No. of testers	0	0	7	28	15

Table 5.4 First test: Test results for "number of profile changes" metric

Time taken (min)	1-5	6-10	11-15	16-20	20-30
No. of testers	0	0	5	30	15

Table 5.5 First test: Test results for "time taken to create a profile" metric

Scale	Very Bad	Bad	Neutral	Good	Excellent
No. of testers	0	0	8	26	16

Table 5.6 First test: Test results for "ease of use" metric

Scale	Very Bad	Bad	Neutral	Good	Excellent
No. of testers	8	25	14	3	0

Table 5.7 First test: Test results for "performance" metric

5.4.2 Second test: Test HumanE with learning approach (excludes initial policy)

Frequency	1-5	6-10	11-15	16-20	21-25
No. of testers	5	7	18	19	1

Table 5.8 Second test: Test results for “number of profile changes” metric

Time taken (min)	1-5	6-10	11-15	16-20	21-30
No. of testers	0	5	24	18	3

Table 5.9 Second test: Test results for “time taken to create a profile” metric

Scale	Very Bad	Bad	Neutral	Good	Excellent
No. of testers	0	0	5	22	23

Table 5.10 Second test: Test results for “ease of use” metric

Scale	Very Bad	Bad	Neutral	Good	Excellent
No. of testers	2	5	12	21	10

Table 5.11 Second test: Test results for “performance” metric

5.4.3 Third test: Test HumanE with learning approach (includes initial policy)

Frequency	1-5	6-10	11-15	16-20	21-25
No. of testers	11	16	15	8	0

Table 5.12 Third test: Test results for “number of profile changes” metric

Time taken (min)	1-5	6-10	11-15	16-20	20-30
No. of testers	8	19	16	7	0

Table 5.13 Third test: Test results for “time taken to create a profile” metric

Scale	Very Bad	Bad	Neutral	Good	Excellent
No. of testers	0	0	4	21	25

Table 5.14 Third test: Test results for “ease of use” metric

Scale	Very Bad	Bad	Neutral	Good	Excellent
No. of testers	0	0	8	18	24

Table 5.15 Third test: Test results for “performance” metric

5.4.4 Scalability

No. of records	50,000	100,000	150,000	200,000	250,000
Time taken (sec)	2	3	5	7	10

Table 5.16 Test results for “scalability” metric

5.4.5 Test result summaries

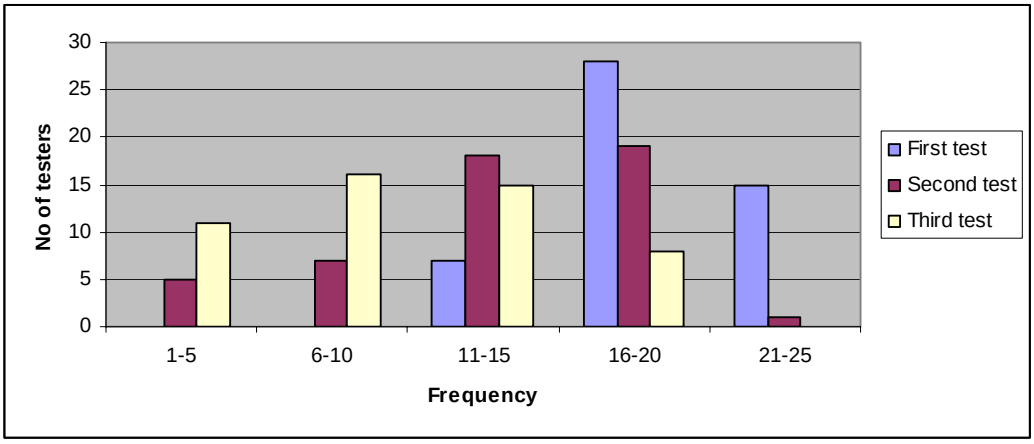

Figure 5.1 Test result summary for “number of profile changes” metric

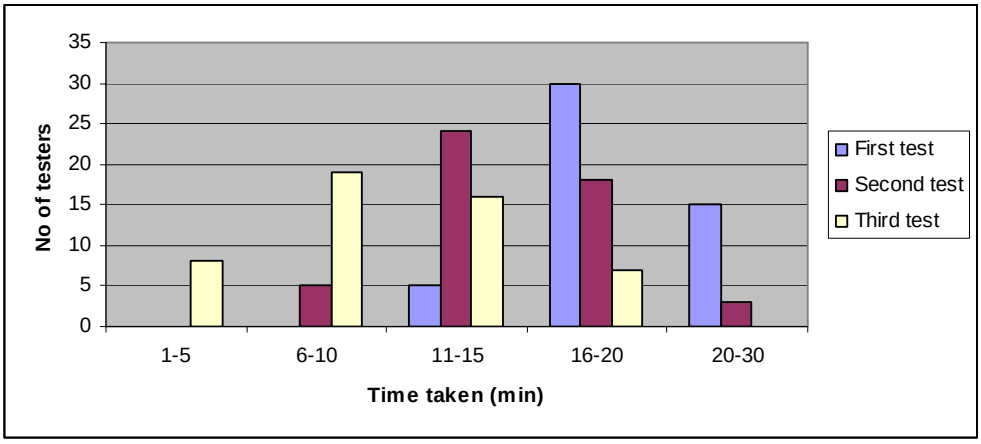

Figure 5.2 Test result summary for “time taken to create a profile” metric

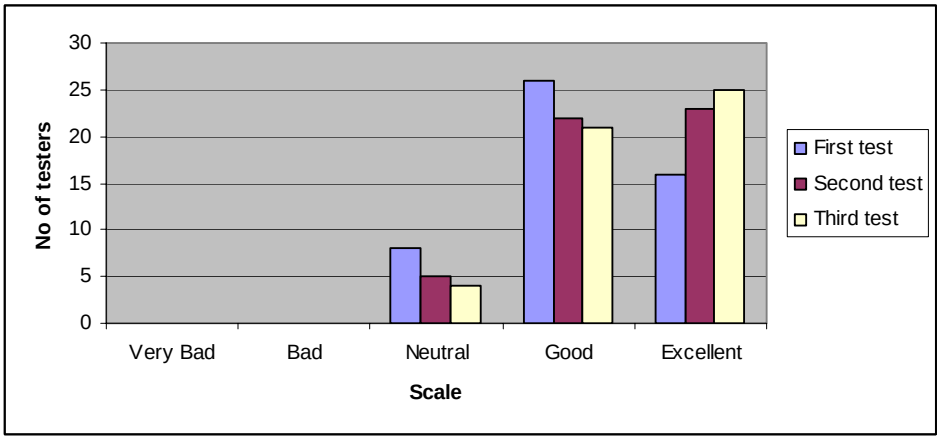

Figure 5.3 Test result summary for "ease of use" metric

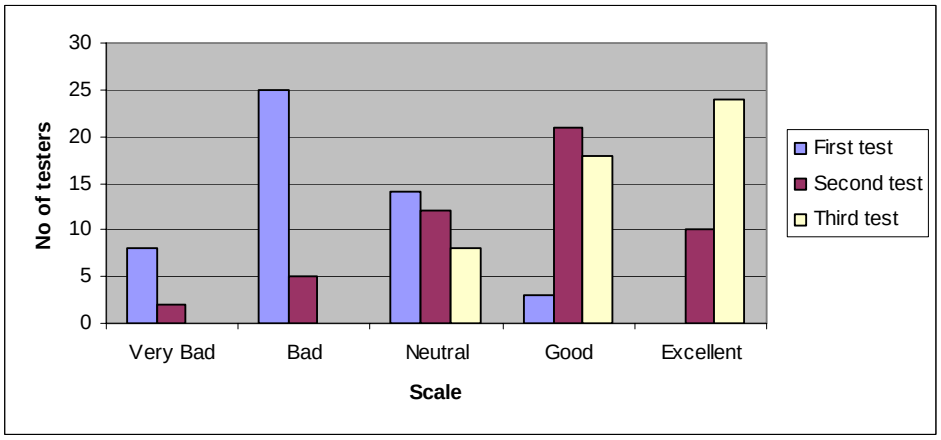

Figure 5.4 Test result summary for "performance" metric

5.5 Discussion

The test results for the first test for the "number of profile changes" metric showed that most testers took eleven to twenty-five profile changes before converging on a satisfactory profile. The test results for the second test for the "number of profile changes" metric showed some improvement as most testers took eleven to twenty searches. The test results for the third test for the "number of searches" metric showed further improvement as most testers took one to fifteen searches. Thus, it is evident that testers tend to make less number of profile changes with HumanE's assistance and even lesser number when HumanE becomes more intelligent with the supply of the initial policy.

Similarly, the time taken to create a satisfactory profile decreased as we introduced a more intelligent HumanE with each test. The test results for the first test for the "time taken to create a profile" metric showed that most testers took sixteen to thirty minutes while most testers in the second test took less time i.e. from eleven to twenty minutes. However, the testers from the third test took the least time as most of them spent six to fifteen minutes. Thus, it is clear that HumanE can reduce the time taken by users when creating and refining their profiles.

The test results for the three tests for the "ease of use" metric are quite similar indicating that almost all of the testers are happy with using HumanE regardless of whether the learning system with or without the initial policy was present or not. Hence, it is safe to say that using HumanE can result in increased customer satisfaction during the apartment selection process.

The test results for the "performance" metric for the first test apparently showed that majority of the testers were not satisfied with the average quality of the "recommended apartments" shown to them for selection and the average response time taken to display an apartment listing. Quite a number of them perceived HumanE as a search engine for apartment listings and they were not satisfied with the perceived browsing metaphor which is offered by typical search engines. Even though many testers were fairly happy that they were given complete control over the entire profile creation process, they also voiced out their displeasure of having to make many tedious profile changes before converging on a good profile. On the other hand, the test results for the second test and the third test showed that the majority of testers preferred to use HumanE to assist them during the apartment selection process. Obviously, the use of HumanE can increase customer satisfaction.

Test results for "scalability" metric showed that HumanE is quite comfortable when it comes to handling large databases. The ability of HumanE to scale is of utmost important with regards to commercial deployment as real-life ecommerce databases are typically large-scale (i.e. several hundred thousand records).

Any agent-based online system implementation should take this into consideration as agent performance affects the feasibility and viability of the implementation especially in

terms of costs as the system has high computing needs which imply that a high investment in hardware and software is required. Not many companies are willingly to make such large investments for ecommerce purpose, especially at this time of writing, due to the sluggish and uncertain global economy. However, we are confident that corporations dealing with complex domains can implement HumanE without massive hardware and software investment. The test results have shown that HumanE is quite comfortable residing on a single-CPU Intel Pentium 4 server with sufficient memory of about 1 GB.

In summary, the experimental results showed that the use of HumanE for complex multidimensional domains such as real estate can result in higher customer satisfaction as it can learn faster via a supplied initial policy and is able to elicit trust from users through its user-friendly interface, quality recommendations and excellent performance.

Chapter 6

CONCLUSION

6.1 Summary

In this chapter, we conclude this thesis with a summary of our research findings and outline our future directions.

6.2 Conclusion

HumanE addresses the problem of poor learning when implementing online implementation of large-scale autonomous agent-based recommender systems for several complex domains through the use of a supplied initial policy which allows it to make more "knowledgeable" exploratory recommendations.

We feel that existing implementations of interactive learning method for online systems are simply impractical as the state-action space is simply too large for the agent to explore within its lifetime. This is further exacerbated by the short attention time-span of typical online users.

It seems easier and more intuitive for the application developer to specify what the agent should be doing and to let it learn the fine details of how to do it. The key strength of our approach is that, by incorporating an initial policy or prior knowledge, HumanE is able to provide better recommendations within a shorter time span. This is because the initial policy has generated some experiences or knowledge about the real estate domain which HumanE can use throughout the interactive learning process. No longer does the user need to face an agent that does not know anything about the task to be completed. We believe that this approach is far more practical and effective than current implementations [16, 28, 29, 64].

We also postulate, contrary to the experimental results obtained from past research [15], that a good initial policy is critical to the success of HumanE from a reward perspective as the user usually takes less time to build a good profile. Good initial policies head directly for the goal state and they typically do not expose much of the state-space, since their trajectories through it are much more directed. This behavior is actually quite

desirable as most online users generally have little patience and want to see a good profile built quickly.

Finally, transferring the work done here to another different domain such as vacation plans, insurance, mutual funds, and mortgages would not require a rocket scientist. The main requirement would be to find a domain expert who would be able to identify the key features of the complex objects in the domain. Creating an initial policy would require the identification of “good” and “bad” features and the classification of features into loosely connected groups.

6.3 Future directions

The development of HumanE will continue to evolve particularly in a different domain i.e. vacation plans. In future versions of HumanE, we would like to incorporate some of the following features to improve its usefulness.

- Refine the initial policy refining algorithm based on the results obtained using more sophisticated data mining tools.

- The ability to ask the user questions in natural language, allow the user to enter the response in natural language, and finally understand the response obtained for profile refinement.

- The ability to seek advice from users with similar profiles via email, interpret the reply so as to refine the profile.

- The ability to submit user profile to multiple domain-specific web sites, and show the user the results online. The agent will also need to parse and understand the listing obtained for profile refinement.

Chapter 7

REFERENCES

[1] A. Chavez, D. Dreilinger, R. Guttman, and P. Maes. "A Real-Life Experiment in Creating an Agent Marketplace." Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97). London, UK, April 1997.

[2] A. Moukas. "Amalthaea: Information Filtering and Discovery using a Multiagent Evolving System." Journal of Applied AI, Vol. 11, No. 5, 1997.

[3] A. Richmond. "Enticing Online Shoppers to Buy – A Human Behavior Study." In the Proceedings of the Fifth International World Wide Web Conference. Paris, France, May 6-10, 1996.

[4] A. Wexelblat and P. Maes. "Experiments on Anthropomorphizing Agents." Submitted to Interactions, ACM Press.

[5] AuctionBot URL: http://auction.eecs.umich.edu/

[6] Avery, C. and Zeckhauser, R. Recommender Systems for Evaluating Computer Messages. CACM. 40(3), pp. 88-89, March 1997.

[7] B. Shneiderman and P. Maes. "Debate: Direct Manipulation vs. Interface Agents." Interactions: New Visions for Human-Computer Interaction, vol. iv.6, December 1997.

[8] Balabanovic, M. and Shoham, Y. Fab: Content-Based, Collaborative Recommendation. CACM. 40(3), pp. 66-72, March 1997.

[9] BargainFinder URL: http://bf.cstar.ac.com/bf (offline)

[10] Basu C., Hirsh H., and Cohen, W.W. 1998. Using Social and Content-Based Information in Recommendation. In Proceedings of the AAAI-98,: AAAI Press.

[11] Belkin, N. J. and Croft, B. W. Information Filtering and Information Retrieval: Two Sides of the Same Coin? CACM. 35(2), December 1992.

Bill Trochim's Center for Social Research Methods URL: http://trochim.human.cornell.edu/

[13] BlueEyes URL: http://www.almaden.ibm.com/cs/BlueEyes/index.html

[14] Boone, G. 1998. Concept Features in Re:Agent, an Intelligent Email Agent. In The Second International Conference on Autonomous Agents, 141-148, Minneapolis/St. Paul, MN:ACM.

[15] Bratman, M. E. (1987). Intentions, Plans, and Practical Reasons. Cambridge, MA: Havard University Press.

[16] Brooks, R.A. (1986), A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation 2:14-23.

[17] Burke, D., Hammond, K. J., and Young, B. C. (1997). "The FindMe Approach to Assisted Browsing," IEEE Expert, pp. 32-40, July-August 1997.

[18] Burke, R., Hammond, K., and Cooper, E. (1996). Knowledge-based navigation of complex information spaces. In Proceedings of the 13th National Conference on Artificial Intelligence, pp. 462-468. AAAI, 1996.

[19] Caglayan, A., Snorrason, M., Jacoby, J., Mazzu, J., Jones, R., and Kumar, K. (1997). Learn Sesame: a learning agent. Applied Artificial Intelligence 11(5): 393-412.

[20] Chin, D. Intelligent Interfaces as Agents. In: Intelligent User Interfaces. J. Sullivan and S. Tyler (eds). ACM Press, New York, New York, 1991.

[21] Cohen, W.W. 1996. Learning Rules that Classify E-mail. In Proceeding of the AAAI Spring Symposium on Machine Learning in Information Access,: AAAI Press.

[22] Creative Research Systems URL: http://www.chartwellsystems.com/sscalc.htm

[23] D. Hawkins, K. Coney, and R. Best. Consumer Behavior: Implications for Marketing Strategy. Business Publications, Inc., 1980.

[24] Dent, L. Boticario, J. Mc Dermott, J. Mitchell, T. and Zabowski, D. A Personal Learning Apprentice. In: Proceedings of the National Conference on Artificial Intelligence, 1992.

[25] Don, A. Anthropomorphism: From Eliza to Terminator 2, panel description. In: Proceedings of the CHI'92 Conference, ACM Press, 1992.

[26] E. Tsang. Foundations of Constraint Satisfaction. Academic Press, 1993.

[27] Firefly URL: http://www.firefly.com/

[28] Franklin, Stan and Graesser, Art: Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents, Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

[29] Decision Support Systems and Intelligent Systems, Efraim Turban and Jay E. Aronson. Copyright 1998, Prentice Hall, Upper Saddle River, NJ.

[30] eBay URL: http://www.ebay.com/

[31] Evaluating Websites URL: http://trochim.human.cornell.edu/WebEval/webeval.htm

[32] Extempo URL: http://www.extempo.com/

[33] G. Salton. “The SMART Retrieval System.” Experiments in Automatic Document Processing. 1971.

[34] Gao X., Sterling L. (1998). Classified Advertisement Search Agent (CASA): a knowledge-based information agent for searching semi-structured text. Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, pp. 621-624.

[35] Google URL: http://www.google.com

[36] Google Groups URL: http://groups.google.com/

[37] GroupLens URL: http://www.grouplens.org

[38] Guttman, R., and Maes, P. “Agent-mediated integrative negotiation for retail electronic commerce.” In Proceedings of the Workshop on Agent-Mediated Electronic Trading AMET 98 (Minneapolis, May 1998).

[39] H. Lieberman. “Letizia: An Agent that Assists Web Browsing.” In Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI’95). Montreal, Canada, 1995.

[40] HDB InfoWeb URL: http://www.hdb.gov.sg/isoa032p.nsf/infoweb?openframeset

[41] Hill, W., Stead, L., Rosenstein, M., Furnas, G. Recommending and Evaluating Choices in a Virtual Community of Use. Proceedings of CHI ’95.

[42] J. Sculley. “The Knowledge Navigator.” Video, 1987.

[43] Jango URL: http://www.jango.com/

[42] K. Kraemer (ed.), The Information Systems Research Challenge: Survey Research Methods, K. Kraemer (ed.), Boston: Harvard Business School, 1991, pp 299-315.

[45] Kasbah URL: http://kasbah.media.mit.edu/

[46] Kay, A. User Interface: A Personal View. In: The Art of Human-Computer Interface Design, B. Laurel (ed), Addison-Wesley, 1990.

[47] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R. and Riedl, J. GroupLens: Applying Collaborative Filtering to Usenet News. CACM. 40(3), March 1997.

[48] Lieberman, H. (1997). Autonomous Interface Agents. In Proceedings of ACM CHI 97,67-74,:ACM.

[49] Maes, P. (1994). Agents that reduce work and information overload. Communications of the ACM 7:31-40.

[50] Maltz, D. and Ehrlich, K. Pointing the Way: Active Collaborative Filtering. Proceedings of CHI '95.

[51] MovieLens URL: http://movielens.umn.edu/

[52] N. Belkin and B. Croft. "Information Filtering and Information Retrieval." Communications of the ACM, 35, No. 12:29–37, 1992.

[53] N. Kushmerick, D. Weld, R. Doorenbos. "Wrapper Induction for Information Extraction." In Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'97), 1997.

[54] Newsted, P. R., W. Chin, O. Ngwenyama, and A. Lee, "Resolved: Surveys have Outlived their Usefulness in IS Research," Panel presented at the 1996 International Conference on Information Systems, December 17, 1997 ,Cleveland, Ohio.

[55] OnSale URL: http://www.onsale.com/

[56] P. Resnik, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. "Grouplens: An Open Architecture for Collaborative Filtering of Netnews." In Proceedings of Computer Supported Cooperative Work (CSCW'94), 1994.

[57] PersonaLogic URL: http://www.personalogic.com/

[58] Pinsonneault, A. & K. Kraemer, "Survey research methodology in management information systems," Journal of Management Information Systems, Fall, 1993, pp 75-105.

[59] R. Doorenbos, O. Etzioni, and D. Weld. "A Scalable Comparison-Shopping Agent for the World Wide Web." Proceedings of the First International Conference on Autonomous Agents (Agents '97). Marina del Rey, CA, February 1997.

[60] Rich, C. and Sidner, C. (1997). COLLAGEN: When Agents Collaborate with People. Proceedings of the First International Conference on Autonomous Agents (Agents ’97). Association for Computing Machinery. 284-291.

[61] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. Journal of Logic and Computation, 8(3):261-292, 1998.

[62] Salton, G. and McGill M. J. Introduction to Modern Information Retrieval. McGraw-Hill, Inc. 1983.

[63] Sarwar, B.M., Konstan, J.A., Borchers, A., Herlocker, J.L., Miller, B.N., and Riedl, J. 1998. Using Filtering Agents to Improve Prediction Quality in the Grouplens Research Collaborative Filtering System. In Proceedings of CSCW ‘98,Seattle, WA.: ACM.

[64] Schneiderman, B., Direct Manipulation: A Step Beyond Programming Languages, IEEE Computer, Vol. 16, No. 8, pp. 57-69, 1983.

[65] Shearin, S. and Lieberman, H. Intelligent Profiling by Example, in Proceedings of Conference on Intelligent User Interfaces, ACM Press, 2001.

[66] Singapore-Real-Estate URL: http://www.singapore-real-estate.com/

[67] SingaporeResidentialProperties URL: http://www.singaporeresidential properties .com/

[68] Sullivan, J.W. and Tyler, S.W. (eds.) Intelligent User Interfaces, ACM Press, 1991.

[69] SurfJet Agent URL: http:// www.leptonicsystems.com/surfjet/

[70] Terveen, L., Hill, W., Amento, B., McDonald, D., Creter J. 1997. PHOAKS: A System for Sharing Recommendations. Communications of the ACM 40(3):59-62.

[71] Tete-a-Tete URL: http://ecommerce.media.mit.edu/tete-atete/

[72] U. Shardanand and P. Maes. “Social Information Filtering: Algorithms for Automating 'Word of Mouth'.” Proceedings of the Computer-Human Interaction Conference (CHI'95), Denver, CO, May 1995.

[73] V. Kumar. “Algorithms for Constraint Satisfaction Problems: A Survey.” AI Magazine, 13(1):32-44, 1992.

[74] William D. Smart and Leslie Pack Kaelbling, “Effective Reinforcement Learning for Mobile Robots,” Proceedings of the IEEE International Conference on Robotics and Automation, 2002.

[75] Y. Lashkari. Webhound Master’s Thesis, MIT Media Laboratory Technical Report, 1995.

[76] Z. Collin, R. Dechter, and S. Kaiz. “On the Feasibility of Distributed Constraint Satisfaction.” Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI’91), 1991.