Agents that represent buyer's interests in E-commerce

Sandip Sen, Partha Sarathi Dutta, & Rajatish Mukherjee
Department of Mathematical & Computer Sciences
University of Tulsa
e-mail:sandip,partha,rajatish@euler.mcs.utulsa.edu

Abstract

Personal agents have been developed that assist user with information processing needs by generating, filtering, collecting, or transforming information. On the other hand internet stores are providing services customized by the needs and interests of individual customers. Such services can be viewed as "seller's agents" whose goal is to push merchandise and/or services on to the users. This leads us to believe that there is a growing need for deploying "buyer's agents" whose goal is to best serve the user's interests. We propose several key functionalities of such buyer's agents: informing consumers of complex interactions between specified preferences and prevailing market conditions, providing differential analysis for decision support, use of ontology to help the user reformulate queries.

Introduction

In the past few years, agent technology has caught the attention of both application developers and system designers (Jennings, Sycara, & Wooldridge 1998). Agents are viewed as a useful metaphor both for developing desktop software designed to assist a particular user (Maes 1994), as well as for internet-based server-side software that enables e-commerce (Nwana *et al.* 1998). The first wave of agents to catch our attention were those that enabled us to skip some of the grunt work, e.g., filtering e-mail (Maes 1994), scheduling meetings (Mitchell *et al.* 1994), collect newsgroup articles (Lang 1995), etc. Most of these were desktop applications. With the rapid explosion of the internet and the World Wide Web a different class of agents came to the fore: agents that can gather and collate information on behalf of their user. With simple queries, the user could now perform powerful searches that would have previously required considerable time and effort investment on his/her part in the past.

The internet also produced an abundance of information which is often overwhelming. Typical search engines return too many links in response to queries. This limits their usefulness as it becomes difficult for the user to differentiate between the relevant and irrelevant data. Personalized web-page recommender systems provide partial solution to this problem as it can suggest web pages of interest to a user based on his/her query and usage patterns (Pazzani, Muramatsu, & Billsus 1996; Rucker & Polanco 1997). Collaborative filtering mechanisms were also developed by which users can obtain or view usage of information by like-minded users (Firefly ; Kautz, Selman, & Shah 1997).

A completely different kind of agent application was also enabled by the growth of internet users. Agent based applications allow merchants and retailers to make their goods and services available to net surfers at the click of a button. In addition to sites which directly sell commodities, other agent-based sites that allow users to buy and sell goods became popular overnight (Maes, Guttman, & Moukas 1999). Such electronic auction-houses allow users to set up or bid in auctions for both new and used goods (AuctionBot ; eMediator ; Kasbah).

The remarkable growth in agent-oriented internet-based applications is encouraging. However, most of these applications appear to open up new possibilities or choices for the user without providing much guidance or help about how best to use this additional information. Though there does exist considerable research in comparison shopping agents (BargainFinder ; Jango ; Maes, Guttman, & Moukas 1999), these agents are not designed to educate the customers about the changing marketplace or the interrelationships between user preferences.

Our goal is to enhance the scope of agent usage by developing agents whose purpose is to educate the user to become a more informed consumer. These agents will serve the interest of the user by understanding the user's goals and recommending products/services or suggesting modification to user queries or requirements that will be more likely to produce results at a higher level of user satisfaction. In particular, these agents will have to educate the user both about possible interactions between his/her preferences for different features and the effects of a rapidly changing marketplace.

Let us use a few example scenario to illustrate our proposed functionality of a *buyer's agent*:

Product feature Selection: User A was looking for a lawn mower with 22" swatch, at least 4.5HP Briggs

& Strattan gas engine, mulching option, and at least two years manufacturer warranty in a price range of less than $300. A shopping agent can perhaps find one or more such product. But consider the scenario that all but one major manufacturer Y, which is not a major manufacturer of lawn mowers, provide warranties of only upto a year. In this case the user requirement of a 2-year warranty will eliminate all such options and restrict the results to only the products offered by manufacturer X. This may also result in an escalated price (even though it is still within user specification, it does not necessarily mean the user will be willing to pay 25% more for example for the extended warranty period), or elimination of other features offfered by other manufacturers. The "buyer's agent" should inform the user of the implied constraint "$Warranty > 1$ year $\Rightarrow$ $Manufacturer = Y$. Note that we are not arguing that the agent argues for the user to necessarily change the specified preferences. Our position is that by providing such additional information, the agent can inform the user about possible consequences of his/her choices. The user then can choose to relax or not relax his/her preferences. In addition, if the user wants the results to be ranked by some attribute, e.g., price, the agent can do "what if" type queries and suggest the reduction in price the user can get by relaxing one or more of the stated preferences. Such differential analysis will allow the user to best restate the query if he/she decided to relax some of his/her constraints. We believe that agents to do the kind of analysis mentioned above can be developed with current technology.

Suggesting alternate products: User B was looking to buy a portable CD player as a present to a friend. This friend is an avid walker/jogger and a portable music player would make a great present for the friend. On hearing B's gift idea, a common friend suggested that in place of the portable CD player, B should consider Diamon multimedia's Rio Diamond Multimedia's new Rio PMP300 player which is a portable and lightweight digital music player for mixing and storing up to thirty minutes of digital-quality music and up to twelve hours of voice quality audio from the Internet or a CD using MP3 compression. This product was recommended because it was easier to carry (being smaller than an audio cassette), has no moving parts and never skips, even during the most extreme movement. Even though this recently released product costs more, B found it to be more appropriate because of its features. The friend was able to suggest the alternate product because of an understanding of B's goal. It may be possible to automate such user's goal recognition for specific scenarios, automating this process in general without significant user guidance is probably infeasible with current technology.

One can list a number of such scenarios where the consumer's initial choice or preference can be modified in the light of new information. The assumption that the average consumer has all the latest information at his/her fingertips is unfounded. On the contrary, rapidly changing market conditions imply that it is next to impossible for the average consumer to keep track of the latest options, deals, package offerings, etc., all of which can influence his/her final choice of what he/she is going to buy and at what price. Our position is that a capable "buyer's agent" can keep track of changing market conditions and inform the user about interactions between stated constraints in queries and the prevailing market. In this paper, we use the term market synonymously with the information environment of the agent.

We have developed an instance of such an agent for the apartment locator domain. This buyer's agent analyses constraints in user queries to infer implied constraints based on current market condition (Sen & Hernandez 2000). In the following we describe the current functionalities of this agent and other functionalities that we are currently working to add to the system.

The lure of e-commerce

One can find e-commerce sites burgeoning all over the internet. While some of these sites are targeted towards business-to-business transactions, many sites are geared for end user to business interactions (Amazon ; eBay ; OnSale). E-commerce applications that fall into the latter category are of relevance to the discussion in this paper. With phenomenal success of e-commerce sites like *Amazon.com* and *eBay*, the entire business model appears to have been reinvented. Not only does e-commerce provide new avenues for selling goods and services and reduced capitals required for marketing and advertising, it also provides the key to tap into a rapidly burgeoning customer base. The netizens and web surfers are feeling empowered by the possibility to browse the goods and services available from the safe haven of their home and are increasingly comfortable in buying goods and services without the assurance of having checked out the offering 'in person'.

In whose interest is this anyway?

But we have to be careful about whether this business model is beneficial to all or most customers. Obviously these e-commerce sites are developed by merchants and retailers whose goal is to sell as much products and services as possible to the consumer visiting their sites. If possible, they would like to discourage comparison shopping of any form. In addition, the average consumer may be overwhelmed by the volume and diversity of information available on the net and may not have the patience or the time to search and shift through all the available information to make a judicious choice. Often word-of-mouth recommendations, which may have been outdated by a rapidly changing market, will be used to make purchasing decisions. All of these factors taken together seem to suggest that though the internet, and in particular the searching ca-

pability provided by WWW portals, can lead a consumer to an e-commerce site of interest to the user, in a significant number of scenarios the scale is tilted in favor of the merchant or retailer at the expense of the consumer.

The empowered consumer

There are of course consumer oriented services like comparison-shopping agents that mostly allow to compare products based on price (BargainFinder). There also exist other products that allow several products to be compared based on user preferences for multiple attributes (LogicalDecisions). These services enable the user to do some form of comparison shopping and hence to make more informed purchasing decisions. Our proposed agent provides a different kind of service in the sense that it makes the consumer aware of his/her preferences and constraints given the current market conditions. In addition, it might educate the user about which constraint or constraints to relax to obtain desired query results. The desired results may be single or multiple attribute based, e.g., products below a given price, products from a particular manufacturer below a given cost, etc. or may involve other preferences like obtaining at least 5 different products to choose from.

User constraints and market conditions

We now present a formal model to represent constraints explicitly specified in a user query and the nature of results returned by querying the current market. The basic assumption in our model is that the environment can be modeled by a relational data model (Elmasri & Navathe 2000). We believe that the relational model is sufficiently general to effectively model the data requirements of most application domains of interest to us. In the following discussion, we will use the term 'database' to any collection of data, irrespective of whether it is stored locally on the user site or distributed over several sites on the network. We will not concern ourselves with the implementation of the relational model (and we have no control on how the data will be implemented in databases across the network), and hence use a universal relation schema $R(A_1, A_2, \ldots, A_n)$, where R is the name of the domain, e.g., Resorts, and each A_i correspond to an attribute in the domain, e.g., Location. A particular element of the relation, e.g., a given resort, is denoted by an n-tuple $t = \langle v_1, v_2, \ldots, v_n \rangle$, where $v_i \in \text{Domain}(A_i)$. For example, a given resort is completely described by a vector of values corresponding to each of the resort attributes.

In addition to domain, key, and integrity constraints, each relational database also comes with a set of *functional dependencies* (FDs). The latter is of particular interest to us as it specifies relationships between different attributes that must alwasy hold. For example, in the resort domain a typical FD would be $Location \rightarrow Tax\_Rate$ which implies that any two resorts located in the same location will also have the same tax rate. We do not expect that an average user will be aware of most of the FDs in the domain, but can benefit from such knowledge. Additionally, the user can significantly benefit from other relationships currently existing in the database. For example, the relationship captured by the rule"Location=*Bahamas* $\Rightarrow$ Ski-slope = *No*" may hold at a given point in time, though it may not have been true in the past or may not be true in the future. Such *dynamic constraints* differ from FDs in two major ways: (a) they are more specific than FDS, i.e., are of the form $(A_i = a_i) \wedge (A_j = a_j) \wedge \ldots \wedge (A_x = a_x) \Rightarrow (A_z = a_z)$ whereas FDs are of the form $A_i A_j \ldots A_x \rightarrow A_z$; (b) whether a rule holds or not on the database changes over time with insertion and deletion of records, but any legal relation state must satisfy all FDs. Since the relationships captured by rules can aid the user's understanding of market dynamics, we propose that our agent infer such rules periodically by analyzing market data. Note that though such inference from data is not justified for inferring FDs (because current relationships do not imply permanent relationships), such inference do serve our purpose of keeeping abreast with market data.

Recognizing implied constraints

The technical question then is how to infer relationships of the above type by looking at the current instance of a relation. There exists several data mining approaches for inferring relationships in data. Our goal is to present these relationships to the user in the form of easy to comprehend rules and hence we did not consider mechanisms like neural or Bayesian nets (we have not considered the possibility of inferring simple rules from learned neural or Bayesian networks). Several rule learning mechanisms have been developed in the machine learning literature that can possibly be used (Clark & Niblett 1989; Michalski *et al.* 1986; Quinlan 1990). We believe that propositional rule learners will be sufficient for most domains, and hence did not use systems that can learn first order rules, e.g., FOIL (Quinlan 1990). In our implementation, we decided to use the rule generation facility associated with C5.0, perhaps the most well-known decision tree learning software (C5.).

When the user poses a query, the constraints in the query can be used to match the antecedents of the learned rules. The consequents of the set of matched rules constitute implied constraints given the user query and the current relationships in the market data. For example, if we reconsider the lawn mower example used in the "Introduction" section, when the user asks for 2-year warranties, this can generate the implied constraint that the manufacturer of all such products is X. This information may make the user relax some of the constraints in the original query. Our position is that being made aware of this additional implied constraints (which the user might not have been aware of), the user can take a more informed decision.

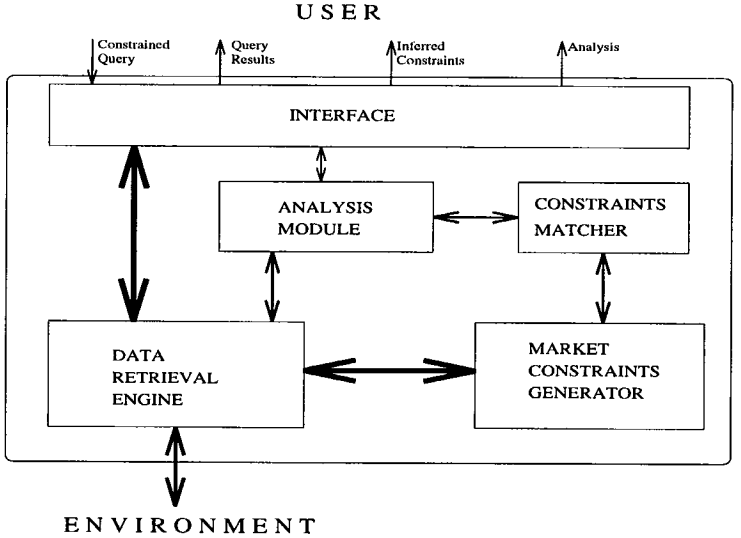

Figure 1: The architecture of our proposed "Buyer's Agent".

Agent architecture

We now present brief descriptions of the different modules in the "Buyer's agent" architecture (see Figure 1):

Interface: The interface allow the user to present a structured, constrained query in the application domain. This query is forwarded both to the data retrieval engine and to the analysis module. Results returned from both are displayed back to the user in an easy-to-browse format.

Data retrieval engine: The data retrieval engine's purpose is to query the market and gather data as required by the user or the other modules. The engine can be a database query engine in the case where all the information resides in a local database. It can also be as complicated as an internet based distributed information gathering mechanism when data resides at possibly multiple remote sites.

Market constraint generator: The market constraint generator uses the query engine to retrieve a sizable or representative portion of current data in the market. This retrieved data is then mined to unearth any significant existing relationships. Whereas in some domains a simple dump of the relevant relations may be sufficient to gather needed data, in other domains sophisticated statistical sampling may have to be performed to gather representative data from non-local sources.

Query matcher: The query matcher contains a relatively simple rule-matching mechanism that matches the constrained user queries with the inferred rules. The result is the generation of all rules that match the constraints of the user query. The consequents of these matched ruleset provides the implied constraints over and above the constraints present in the user query.

Analysis module: The analysis module calls the query matcher to generate the implied constraints from the user query. In general it can also perform other kind of analysis, e.g., differential analysis by relaxing some of the constraints in the user's query. This will enable the module to make specific recommendations to the user regarding how best to relax constraints in the query to obtain desired results.

The apartment location domain

To evaluate the feasibility and usefulness of the "buyer's agent" concept, we chose an apartment location domain. The relational model for the apartment location domain has 17 attributes, e.g., NumBedrooms, DepositAmount, MinimumLease, Rent, WD (Washer-Drier), etc. There exists e-commerce sites using which one can search for apartments in a particular city or region (Apartments ; Rent). We visualize our "buyer's agent" to be an agent "higher up the food chain" which can query such sources by its data retrieval engine (Etzioni 1997).

Implementation issues

For our prototype implementation, we adopted a more straightforward approach. First, we limited our domain to the city of Tulsa. Then we cached the data from all the listed apartments in Tulsa in a local database. The next step was to mine this database to generate the rules. As mentioned above, we used the C5.0 system to generate the rules that capture the existing relationships in the data. To do this we ran the rule generator many times, once for each of the attributes in the domain. Each attribute was chosen once as the target attribute, and the rest of the attributes were used to predict the values for this attribute. We had to process the raw data for this stage. In particular, when the continuous attributes were used as the target attribute we had to discretize them as the rule generation procedure works with only discrete-valued target attributes. For example, when we used *Rent* as the target attribute, we defined several ranges into which the apartments were classified, e.g., 500–599 was defined as midrange (MR). When *Rent* was used as an attribute to classify another target attribute, however, the continuous values were used, e.g., 450 rather than MR.

The rules learned were not necessarily 100% accurate. We decided to use rules that even though not completely accurate, were accurate in a large percenage of cases (we used 90% as the cutoff) they matched. This means that the implied constraints were not without exceptions, but the exceptions were small enough in number to warrant the presentation of this constraint. Typical rules learned include the following:

$$Location = C \Rightarrow CoveredParking = N,$$

which means apartments in Central Tulsa do not have covered parking;

$$Furnished = Y \wedge Playground = Y \Rightarrow Location = SE,$$

which means all furnished apartments with playgrounds are located in SouthEast Tulsa;

$$WDHookups = Y \wedge SqFeet <= 600 \Rightarrow Nobed = Studio,$$

which means all apartments of less than 600 square foot with washer-drier hookups are studio apartments.

Note that rules are not necessarily causal but appear to make sense. It is likely that every city will have certain locations with predominance of certain kinds of apartment features. The learning process did unearth a lot of patterns that we were not expecting. It is important to recognize that these patterns are necessarily impermanent and hence it may not be useful to search for any fundamental long-lasting correlation between the antecedents and consequents of the learned rules.

Ongoing work

Differential analysis

We are currently working on the differential analysis aspect of the analysis module of our implementation of the buyer's agent. The goal here is to offer the user a set of decision criteria he/she wants to be fulfilled by the system, e.g., maximize number of alternatives returned, minimize price, etc. If the user chooses any one of these criteria, then a differential analysis will identify the constraint or constraints to be altered/relaxed to maximally improve the quality of the query results. This facility allows the user to immediately understand different "what-if" scenarios which can allow him to identify constraints to relax. For example, the user may find it preferable to relax the constraint of covered parking to get a $75 per month reduction in rent.

To perform the differential analysis we create a list containing the user specified constraints. Based on these constraints, we create a set of queries, by dropping exactly one of the constraints in the list at a time. Let a query Q be represented by the set of constraints contained in it. Then the set of new queries to perform differential analysis, $Q_D$is given by:$Q_D = \cup_{c \in Q} Q \backslash \{c\}$. We also ask the user to provide a criterion to optimize. Typical criteria include minimizing rent, maximizing number of options returned, mazimizing square footage, etc. Let us illustrate the process by assuming that the user wants to minimize rent. For each of the queries generated above, we query the apartment database and obtain the minimum rent with these set of constraints. That is, for each of the constraints in the original query, we now know what is the minimum rent that can be obtained if we drop that constraint. Finally we sort the set of constraints in the originial query by these values, which gives a list of constraints the user should consider relaxing in order if he/she wants to get a lower rent than what was returned with the original query.

Figure 2 presents a typical result screen based on a user query in this domain. The results include a list contains the set of apartments that satisfies the user constraints, a second list contains the set of implied constraints given the prevalent market conditions and

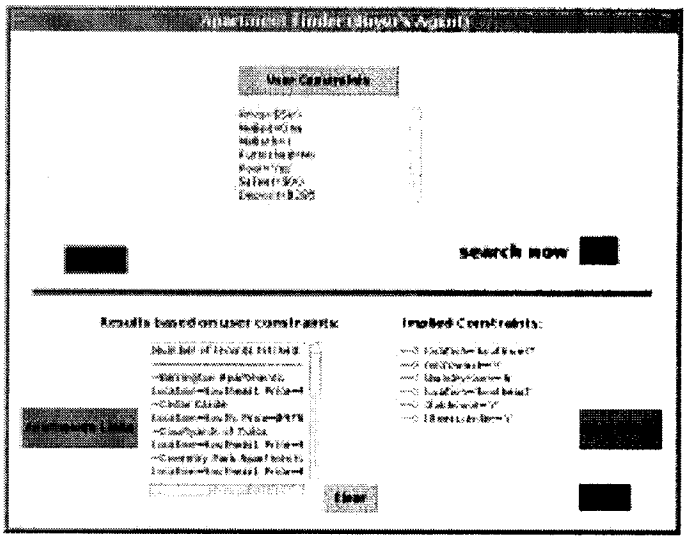

Figure 2: The result screen.

the user constraints, and a ranked order of the constraints in the user query in terms of their effects on the apartment rent. The user can then choose to view a page of links of using which he/she can visit selected apartment home pages. The user can also choose to find out the reason for the implied constraints and the rules that produced those constraints are displayed.

Ontology based guidance

A recent survey, published in Business Week magazine dated February 7, 2000, identifies the following three relevant problems to be among the top ten pet peeves for on-line shoppers during the last holiday season: lack of choices, lack of gift ideas, lack of relevant information. This suggests the necessity of our proposed agents be able to suggest alternate products as argued in the Introdcution section. We believe some of these concerns can be addressed by the careful use of structured ontology to represent domain information. Our proposed use of ontology is to store categorical information about product types and relationships across categories in terms of identified features like quality, price, popularity, etc.

There are very many different ways that ontologies can be used to build smarter agents (Guarino, Masolo, & Vetere 1999). We plan to concentrate on using ontologies to reformulate user queries, suggest alternatives, etc. In the following we list three different query reformulation strategies based on domain knowledge captured in ontological structures. We consider the domain of consumer electonics in all these examples; portions of the ontology is presented in Figure 3.

Query reformulation on failed search: Suppose no match is found when a buyer wants to buy a 900MHz cordless phone for under $30. Our ontology orders product subcategories under a category in the order

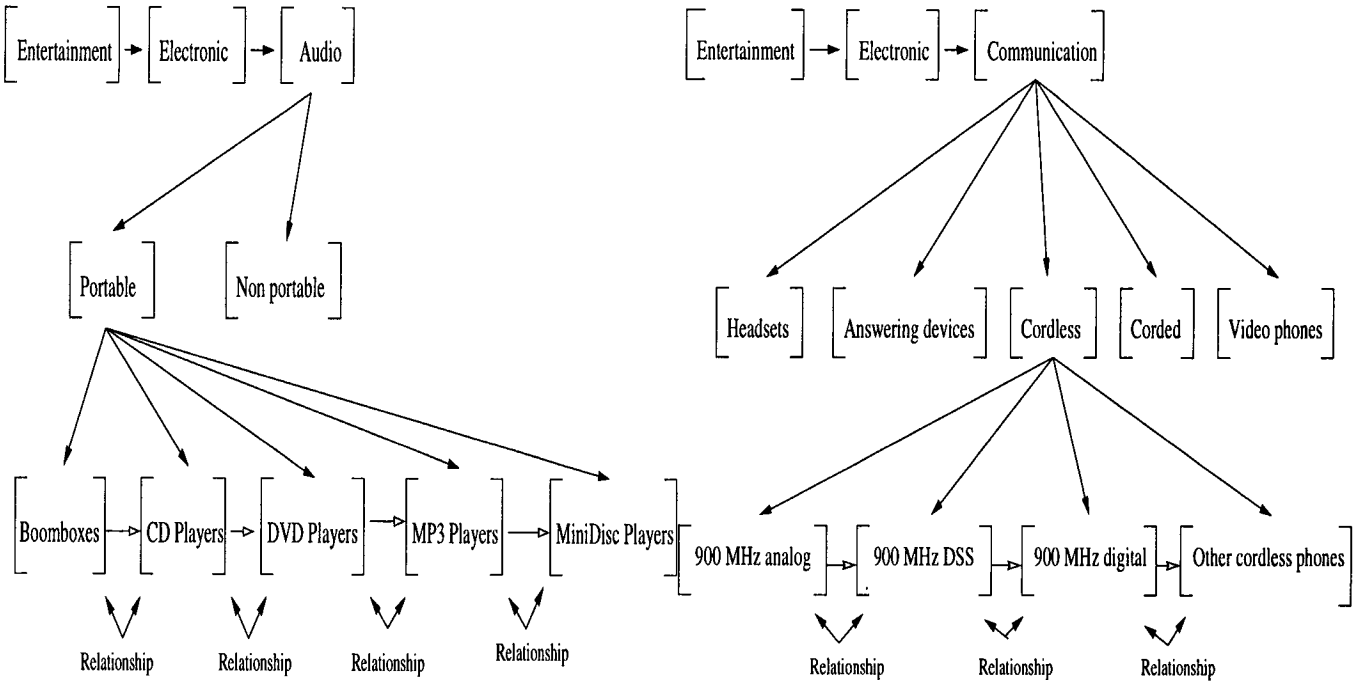

Figure 3: Sections of a consumer electronics ontology.

of increasing quality. In the ontology there is a relationship between 900MHz cordless sets, sets with digital spread spectrum technology, sets with 2 line option, etc. So, when a search for the given query fails to return any results, our agent looks for other subcategories with lower quality for a match in the price. If it finds a digital spread spectrum cordless phone or any other alternative product within the budget of the consumer it reports him/her about the prospective alternative. The consumer gets the information about other compatible products that was not known to him/her. The agent also returns by how much the user has to raise his/her price to get the product that was originally specified. This additional information allows the user to take the decision of either increasing the amount he/she is ready to pay or choose an alternate product.

Reformulating an overly general query: If a consumer searches for a camera in the price range of $150 to $200, a large number of matches can be returned. Faced with a very large list of options the user is likely to overlook preferred choices. In the ontology for cameras, there are relationships among camera categories. The agent can present the subcategories of cameras, e.g., point-and-shoot, SLR, digital cameras, etc. and ask the user to narrow down the search to the subcategory of interest. This suggestion may allow the user to sufficiently constraint the search to select products to meet his/her requirements.

Suggesting alternate products Even when the query of a user returns a reasonable number of matches, alternate product suggestion can be useful. This was the case in the example in the Introduction section when the Rio MP3 player was suggested as an alternative to a portable CD player. Suppose a buyer is looking for *portable CD players under $75.* In addition to returning the products that match this specification, the buyer's agent may suggest that an increased quality (no skipping) product can be purchased if the user was to increase the price range to $100. A consumer who is interested in a portable CD player is naturally looking for high qaulity portable audio system. Suggestion of alternative productions can both educate the user (who might not have been aware of new products/features in the marketplace) and allow him/her to select from a more comprehensive list of choices.

Conclusions

In this paper, we argued for the usefulness of a "Buyer's agent" which will enable an average user to make more informed choices while choosing products or services from electronic commerce sites. We posit that the average consumer finds it difficult to keep abreast of market conditions, and can look for features in a product he/she is interested in purchasing that can restrict the set of choices, increase price, etc. The buyer's agent can keep abreast with the prevailing market condition and to inform the user of implied constraints, relative effects of different constraints, alternative products, etc. The general concept of a buyer's agent is a very powerful one: the vision is that of a knowledgeable well-wisher

helping the user to select the product or service that is most satisfying for the user. We have presented our initial steps towards that vision.

Acknowledgements: This work has been supported, in part, by an NSF CAREER Award: IIS-9702672. We also acknowledge the Karina Hernandez's work on implementing the buyer's agent in the apartment location domain.

References

Amazon.com. URL: http://www.amazon.com/.

Apartments.com. URL: http://www.apartments.com/.

Auctionbot. URL: http://auction.eecs.umich.edu/.

Bargainfinder. URL: http://bf.cstar.ac.com/.

C5.0/See5. URL: http://www.rulequest.com/see5-info.html.

Clark, P., and Niblett, R. 1989. The CN2 induction algorithm. *Machine Learning* 3:261–284.

ebay. URL: http://www.ebay.com/.

Elmasri, R., and Navathe, S. B. 2000. *Fundamentals of Database Systems.* Reading, MA: Addison-Wesley.

emediator. URL: http://ecommerce.cs.wustl.edu/emediator/.

Etzioni, O. 1997. Moving up the information food chain: Deploying softbots on the world wide web. *AI Magazine* 18(2):11–18.

Firefly. URL: http://www.firefly.com/.

Guarino, N.; Masolo, C.; and Vetere, G. 1999. Ontoseek: Content-based access to the web. *IEEE Intelligent Systems* 70–80.

Jango. URL: http://www.jango.com/.

Jennings, N.; Sycara, K.; and Wooldridge, M. 1998. A roadmap of agent research and development. *International Journal of Autonomous Agents and Multi-Agent Systems* 1(1):7–38.

Kasbah. URL: http://www.kasbah.media.mit.edu/.

Kautz, H.; Selman, B.; and Shah, M. 1997. Combining social networkds and collaborative filtering. *Communications of the ACM* 40(3).

Lang, K. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning,* 331–339. San Francisco, CA: Morgan Kaufmann.

Logical decisions. URL: http://www.logicaldecisions.com/.

Maes, P.; Guttman, R. H.; and Moukas, A. G. 1999. Agents that buy and sell. *Communications of the ACM* 42(3).

Maes, P. 1994. Agents that reduce work and information overload. *Communications of the ACM* 37(7):31–40.

Michalski, R.; Mozetic, I.; Hong, J.; and Lavrac, H. 1986. The multi-purpose incremental learning system AQ15 and its testing application to three medical domain. In *Proceedings of the Fifth National Conference on Artificial Intelligence,* 1041–1045. Philadelphia, PA: Morgan Kaufmann.

Mitchell, T.; Caruana, R.; Freitag, D.; McDermott, J.; and Zabowski, D. 1994. Experience with a learning personal assistant. *Communications of the ACM* 37(7):80–91.

Nwana, H. S.; Rosenschein, J.; Sandholm, T.; Sierra, C.; Maes, P.; and Guttman, R. 1998. Agent-mediated electronic commerce: Issues, challenges and some viewpoints. In *Proceedings of the Second International Conference on Autonomous Agents,* 189–196. New York: NY: ACM Press.

Onsale. URL: http://www.onsale.com/.

Pazzani, M.; Muramatsu, J.; and Billsus, D. 1996. Syskill & Webert: Identifying interesting web sites. In *Proc. of the 13th National Conference on Artificial Intelligence,* 74–79. MIT Press/AAAI Press.

Quinlan, R. J. 1990. Learning logical definitions from relations. *Machine Learning* 5(3):239–266.

Rent. URL: http://www.rent.net/.

Rucker, J., and Polanco, M. J. 1997. Personalized navigation for the web. *Communications of the ACM* 40(3).

Sen, S., and Hernandez, K. 2000. A Buyer's Agent. In *Proceedings of the Fourth International Conference on Autonomous Agents (to appear).* ACM Press.