A *Buyer's Agent*

Sandip Sen
Mathematical & Computer Sciences Department
University of Tulsa
sandip@kolkata.mcs.utulsa.edu

Karina Hernandez
Mathematical & Computer Sciences Department
University of Tulsa
karinah@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. In particular, such agents should alert the user of the complex interactions between specified preferences and prevailing market conditions. The availability of such information makes the user a more informed customer. We discuss the design and prototype implementation of a buyer's agent in an apartment locator application.

1. INTRODUCTION

In the past few years, agent technology has caught the attention of both application developers and system designers [21]. Agents are viewed as a useful metaphor both for developing desktop software designed to assist a particular user [25], as well as for internet-based server-side software that enables e-commerce [29]. 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 [25], scheduling meetings [28], collect newsgroup articles [24], 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 [20]. 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 [30, 32]. Collaborative filtering mechanisms were also developed by which users can obtain or view usage of information by like-minded users [8, 23].

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 [26]. Such electronic auction-houses allow users to set up or bid in auctions for both new and used goods [3, 7, 10].

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 [4, 9, 26], 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 X, 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 $\Rightarrow Manufacturer = X$. 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 recall that in the two example scenarios listed above, we hypothesized that we can develop automated intelligent assitant agents to help users in at least the first scenario. We propose a framework to develop such agents in this paper. The framework presents the capabilities and funcationalities of our proposed agent. We also provide a prototype implementation of such an agent in the domain of apartment selection. We present lessons learnt from developing this agent and identify future work needed to enhance the capability of this agent.

2. 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 [1, 6, 12]. 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'.

2.1 Sellers versus consumers

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 capability 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.

2.2 The empowered consumer

There are of course consumer oriented services like comparison-shopping agents that mostly allow to compare products based on price [4]. There also exist other products that allow several products to be compared based on user preferences for multiple attributes [11]. 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.

3. 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 [18]. 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.

3.1 Static and dynamic constraints

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.

3.2 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 [14, 27, 31]. 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 [31]. In our implementation, we decided to use the rule generation facility associated with C5.0, perhaps the most well-known decision tree learning software [5].

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 regarding purchasing goods and services of interest.

4. 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.

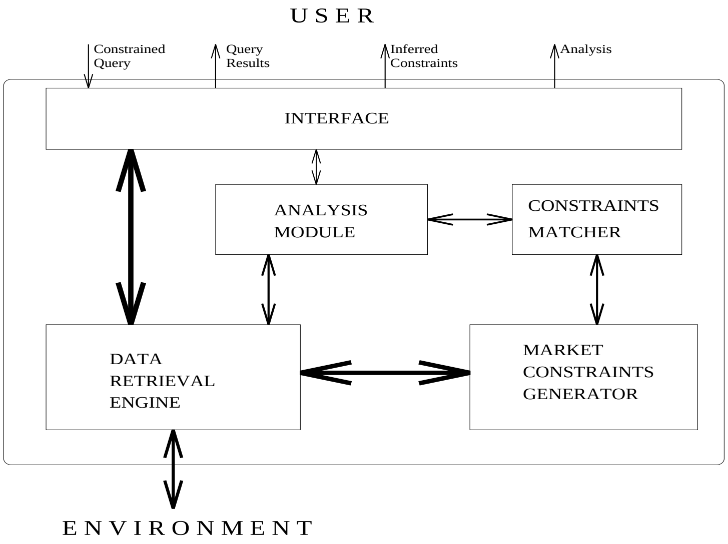

Figure 1: The architecture of our proposed "Buyer's Agent".

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.

Constraints matcher: The constraints 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 constraints matcher to generate the implied constraints from the user query. In general it can 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. If provided with a rich domain ontology, the analysis module can also be used to map a user query into a more appropriate query to return closely related product/service information.

5. THE APARTMENT LOCATION DOMAIN

To evaluate the feasibility and usefulness of the "buyer's agent" concept, we chose two disparate domains for initial exploration: a vacation resort selection domain, and an apartment location domain. In this paper, we report on our experience with the latter of these two domains. 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 [2, 13]. We visualize our "buyer's agent" to be an agent "higher up the food chain" [19] which can query such sources by its data retrieval engine.

5.1 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 they matched. We have used rules with accuracy above a threshold of 90%. 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. We observed that if the rule performance threshold was lowered below 80%, too many rules were being generated. These 'not very accurate' generalizations are more likely to confuse rather than aid the user. We believe that we have to experiment with the more data (e.g., from other cities) and/or other domain before we can prescribe a robust cutoff threshold that produces few succinct and useful generalizations.

Typical rules learned by the process described above 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.

The algorithm for rule generation is given in Figure2. The LEARN-RULE-SET function can be any rule learning algorithm; we have used C5.0 in our implementation.

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

LEARN-DOMAIN-RULES($Attributes, Examples, Threshold$)
$Rules \leftarrow \phi$
For each $A \in Attributes$
  $A^{-} \leftarrow Attributes \backslash \{A\}$
  $r_A \leftarrow$ LEARN-RULE-SET($A, A^{-}, Examples$)
  $r_A \leftarrow$ ELIMINATE-INACCURATE-RULES($r_A, Threshold$)
  $r_A \leftarrow$ DROP-NEGATIVE-CONDITIONS(r_A)
  $Rules \leftarrow Rules \cup r_A$
$Rules \leftarrow$ ELIMINATE-DUPLICATES($Rules$)

Figure 2: Algorithm for generating all rules in a domain.

it may not be useful to search for any fundamental long-lasting correlation between the antecedents and consequents of the learned rules.

A post-processing step that we decided to invoke, DROP-NEGATIVE-CONDITIONS, removed negative conditions from the rule. The reason for removing negative conditions was that for most features it is unlikely that the user will specify a negative constraint; e.g., we do not envisage user's not wanting covered parking space. Alternatively we could have defined the matching process such that a query with no constraints for an attribute will also match a rule which has a negative constraint. In the following we present a rule before and after post-processing:
Rule before post-processing:

$$(Location = SE) \wedge (Playground = N) \Rightarrow Alarm = N,$$

Rule after post-processing:

$$(Location = SE) \Rightarrow Alarm = N.$$

The rule with the relaxed constraint is also tested for accuracy and is substituted for the original rule only if its accuracy is above the chosen threshold. This constraint-dropping process generated two types of redundancies in the resultant rule sets: identical rules, and rules that were more general than other rules. So far, we have eliminated only duplicate rules. Since specific rules are at times more accurate than their more general counterparts, we have not eliminated them from consideration. This will be useful when the user can dynamically change the threshold for the accuracy of rules to be used in identifying implied constraints.

The resultant rule base that we obtained through the above-mentioned process has a little more than 100 rules. This number depends on the rule accuracy threshold that we have selected.

6. SYSTEM IN USE

The user constraints regarding apartment features are input over two screens in the current screen. Since the number of features is quite large, we break the choices over two screens. In the first screen, the user chooses what we consider to be the important constraints, e.g., rent range, number of bedrooms, etc. (See Figure 3). On the second screen, the less important features are listed with the user having the option of choosing *Yes*, *No*, or *Don't Care* for any of the features (See Figure 4). The default option for each such feature is provided as *Don't Care*.

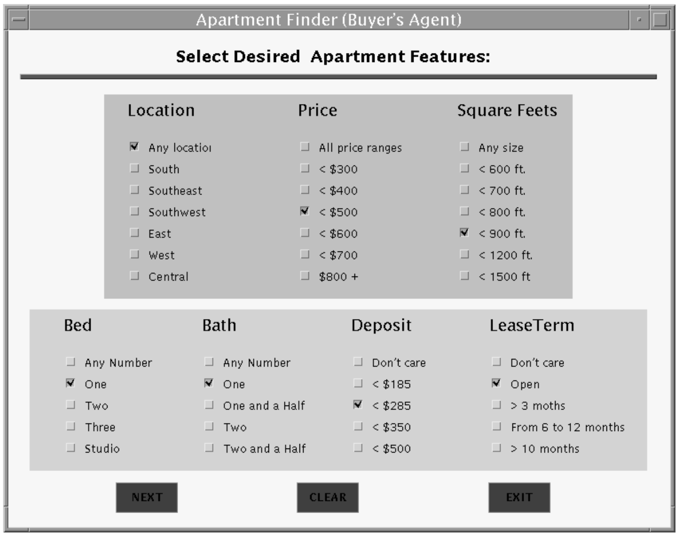

Figure 3: The first screen of preference choice.

Figure 4: The second screen of preference choice.

After the user has filled in the constraint, a new page is displayed with all the constraints selected by the user. If the user now confirms these selections, the query is processed by the agent and the results returned in a new page (see Figure 5). In the current form, two kinds of information is returned:

- the first 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 the user constraints.

The user can then choose to view a page of links using which he/she can visit the corresponding apartment home pages of the selected apartments. The user can also choose to find

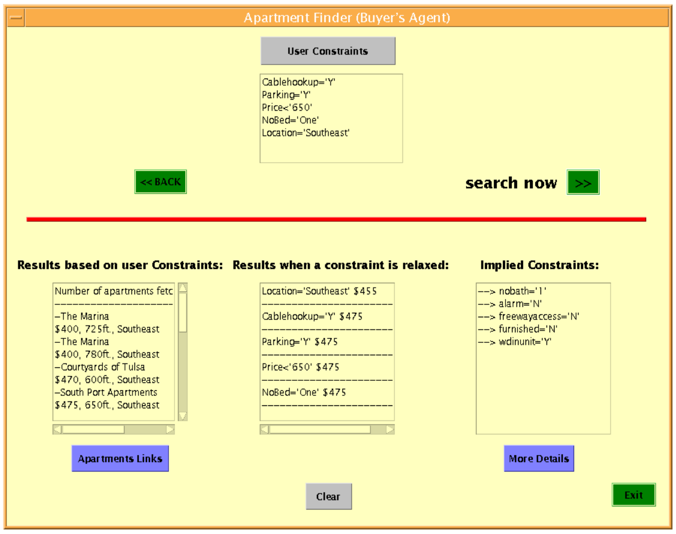

Figure 5: The result screen.

out the reason for the implied constraints and the rules that produced those constraints are displayed.

The system has been used by several lab members who are either looking for apartments, or have searched for an apartment in Tulsa in the recent past. Though there are some concerns about the interface layout, the added functionality provided by our "buyer's agent" was well appreciated. We plan to futher evaluate the system by first making the system more widely accessible and having an on-line structured survey [16, 15, 17] as well as a free-format comments feedback option [22].

7. 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 goal of our proposed buyer's agent is to keep abreast with the prevailing market condition and to inform the user of new constraints that will be imposed because of constraints explicitly specified in the user's query (e.g., asking for extended warranty may severely restrict the manufacturers of products or the products itself).

We identified the scope of our such a buyer's agent, proposed an architecture to deliver the suggested functionality, described the implementation in an apartment location domain, and presented sample interactions with the system.

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.

As proposed in the second example scenario in the "Introduction" section, the scope of the buyer's agent can be expanded if we are able to introduce more sophisticated reasoning on the part of the agent. This will require remembering the past interaction behavior of the user, tracking his/her interest changes, and inferring his/her actual goals from stated queries. 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. The work reported in the current paper is a small, but important step towards that vision.

8. ACKNOWLEDGEMENTS

This work has been supported, in part, by an NSF CAREER Award: IIS-9702672.

9. REFERENCES

[1] Amazon.com. URL: http://www.amazon.com/.

[2] Apartments.com. URL: http://www.apartments.com/.

[3] Auctionbot. URL: http://auction.eecs.umich.edu/.

[4] Bargainfinder. URL: http://bf.cstar.ac.com/.

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

[6] ebay. URL: http://www.ebay.com/.

[7] emediator. URL: http://ecommerce.cs.wustl.edu/emediator/.

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

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

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

[11] Logical decisions. URL: http://www.logicaldecisions.com/.

[12] Onsale. URL: http://www.onsale.com/.

[13] Rent. URL: http://www.rent.net/.

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

[15] D. S. B. Davis and A. M. Berstein. From survey to service: using patron input to improve customer satisfaction. *Technical Services Quarterly*, 14(3):47 – 62, 1997.

[16] F. D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *Management Information Systems Quarterly*, pages 319 – 340, September 1989.

[17] W. J. Doll, A. Hendrickson, and X. Deng. Using Davis's perceived usefulness and ease-of-use instruments for decision making: A confirmatory and multigroup invariance analysis. *Decision Sciences*, 29(4):839 – 869, 1998.

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

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

[20] O. Etzioni and D. Weld. Softbot-based interface to the internet. *Communications of the ACM*, 37(7):72–79, July 1994.

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

[22] N. C. R. Jr., C. Bauer, H. Chen, and J. F. N. Jr. The Mindmine comment analysis tool for collaborative attitude solicitation, analysis, sense-making and visualization. In *Proceedings of the Thirty-Third Annual Hawai'i International Conference on System Sciences*, 2000.

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

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

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

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

[27] R. Michalski, I. Mozetic, J. Hong, and H. Lavrac. 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*, pages 1041–1045, Philadelphia, PA, 1986. Morgan Kaufmann.

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

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

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

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

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