Adaptive Assistants for Customized E-Shopping

Filippo Menczer and W. Nick Street, *University of Iowa*

Alvaro E. Monge, *California State University, Long Beach*

Shopping bots today are biased toward vendors who pay to be listed. To make user-centered business models viable, we developed IntelliShopper, which learns shoppers' individual preferences and autonomously monitors vendor sites for items matching those preferences.

E-commerce has changed the way companies distribute their products and services to consumers. Traditional brick-and-mortar companies continue growing the online segment of the economy by establishing e-commerce presences. E-commerce helps them expand their markets and reduce their costs. Internet shopping also presents consumer benefits such as convenience and low prices. But such benefits are partly offset by a new cost—the time and frustration required to sift through information on innumerable vendor sites and to learn how to submit and manage orders. These tasks are difficult because the type, amount, and organization of information differ from vendor to vendor. Complicating matters, customers are unaware of changes in pricing, availability, and so on unless they visit the same sites frequently, removing a potential advantage of online stores over traditional ones. Can intelligent agents improve the e-shopping experience?

Currently, online shoppers can adopt a number of strategies when looking for a product. The most straightforward approach is to visit various vendor sites; for each site, the shopper browses or searches for a particular product. This simple approach has several drawbacks. First, because no single site caters to all shopping needs, the user's search time increases for each new product category. Second, getting acquainted with individual nonstandard vendor interfaces slows browsing and hinders impulse shopping. Third, this approach likely favors only the largest vendors (owing to name-branding), which reduces the market's efficiency by providing fewer competitive choices to consumers.

Several services let shoppers sign up to receive price alerts that notify them when a product's price changes or falls below a specified amount. Some of these services require shoppers to fill out lengthy surveys, yet most of the sites offer little or no customization. This shopping approach also weakens user privacy.

Another solution involves compilation of voluntary user ratings and reviews of vendors and products.[1] Such recommendation systems might reduce the marketplace's size and introduce bias, because obtaining a sufficient number of ratings for every vendor and controlling the sources' reliability are difficult.

Another alternative further automates and generalizes the search.[2,3] As early as 1995, researchers proposed comparison-shopping agents (also known as shopping bots) as a solution for finding products under the best terms (price was typically the most important feature) among vendor sites. A shopping agent queries multiple sites on behalf of a shopper to gather pricing and other information on products and services. Most comparison-shopping agents, however, present a marketplace that is biased in favor of the vendor sites that collaborate with (pay fees to) the shopping agent. In addition, a shopper has only a limited number of vendor sites to choose from, and often the participating sites do not offer the best prices. (For more on shopping agents, see the related sidebar.)

Intelligent customization techniques can greatly improve the accessibility and consumer benefits of e-shopping by creating a personalized (and thus more efficient) marketplace. We designed and built *IntelliShopper*, a new type of shopping agent that can provide customers with adaptive and customized shopping assistance. IntelliShopper learns users' personal preferences and autonomously shops on their behalf while protecting their privacy.

Goals

We developed IntelliShopper as an intelligent

1094-7167/02/$17.00 © 2002 IEEE

Shopping Agents

Shopping agent research dates to the Web's early years. In 1995, Andersen Consulting developed BargainFinder, the first shopping agent.[1] It let users compare prices of music CDs from Internet stores. However, some retailers blocked access because they did not want to compete on price, and BargainFinder ceased operation.

PersonaLogic, another unbiased comparison-shopping agent, let users create preference profiles to describe their tastes. This approach allowed the shopping agent to identify products with features important to users. However, vendors had to provide interfaces that explicitly disclosed product features such that the shopping agent could match them with user profiles. AOL (America Online) acquired PersonaLogic in 1998, and the technology disappeared.

Ringo was an agent that recommended entertainment products (such as CDs and movies) on the basis of collaborative filtering, using opinions of like-minded users.[2] This became one of the earliest commercialized software-agent technologies when it evolved into the FireFly agent. FireFly addressed privacy issues by initiating and promoting the P3P standard. Microsoft acquired FireFly Network Inc. in 1998, and the FireFly agent ceased operation shortly thereafter. However, collaborative filtering has become a common technique—for example, large commercial vendors such as Amazon use it, although in simplified ways.

ShopBot, another agent, could submit queries to e-commerce sites and interpret the resulting hits to identify lowest-priced items.[3] ShopBot automated the building of "wrappers" to parse semistructured (HTML) documents and extract features such as product descriptions and prices. Our goals with IntelliShopper are similar, but we focus on learning user preferences, and we use a manual approach for specifying how to extract those features from vendor sites. The ShopBot technology's fate was similar to those of PersonaLogic and FireFly. Excite acquired and commercialized it (under the name Jango) but soon replaced it with a biased vendor-driven agent.

Tete@Tete was an agent that integrated product brokering, merchant brokering, and negotiation.[4] A startup called Frictionless Commerce is applying the technology to business-to-business markets (e-sourcing) rather than to business-to-customer markets.

Most comparison-shopping agents now available to consumers (MySimon, DealTime, and RoboShopper, for instance) are biased, presenting results only from partner companies who pay fees to participate. The current business model for shopping bots is based on vendor revenue, not buyer revenue; users are reluctant to pay fees for these services. However, a vendor-revenue model produces hidden costs such as higher prices, limited choice, and poor service. In this context, the established vendors' reluctance to be friendly to shopping bots is certainly understandable.

References

1. B. Krulwich, "The BargainFinder Agent: Comparison Price Shopping on the Internet," *Agents, Bots and Other Internet Beasties*, J. Williams, ed., Sams (Macmillan), Indianapolis, 1996, pp. 257–263.
2. U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating 'Word of Mouth'," *Proc. ACM Conf. Human Factors in Computing Systems* (CHI 95), ACM Press, 1995, pp. 210–217; www.acm.org/sigchi/chi95/Electronic/documnts/papers/us_bdy.htm.
3. R.B. Doorenbos, O. Etzioni, and D.S. Weld, "A Scalable Comparison-Shopping Agent for the World Wide Web," *Proc. 1st Int'l. Conf. Autonomous Agents*, ACM Press, New York, 1997, pp. 39–48.
4. P. Maes, R.H. Guttman, and A. Moukas, "Agents That Buy and Sell," *Comm. ACM*, vol. 42, no. 3, Mar. 1999, pp. 81–91.

agent that could provide real shoppers with real assistance. We had three design goals.

First, the shopping assistant should customize its behavior adaptively by learning each user's preferences. The shopping assistant can achieve this personalization by observing a user's shopping actions. When a user considers the items available at Internet shopping sites, he or she indirectly provides feedback by browsing. The agent can infer user preferences from this feedback and apply that knowledge to take initiative on future searches and to predict when an item might interest a user. Adaptive customization means that the shopping assistant should be able to track and cover each user's dynamic and diverse shopping behaviors.

Second, the shopping assistant should provide the best possible service by remaining as independent (autonomous) as possible from both customers and vendors. Independence from vendors ensures that the service remains unbiased by performing wide searches (instead of searching only the databases of "partner" vendors). Such autonomy results from more-uniform interfaces and improved methods for interpreting potential hits. Independence from the customer means that the shopping assistant proactively monitors vendor sites on the user's behalf, notifying the user of products that might interest him or her. This autonomy relieves users of tediously searching for information and adjusting to different vendor sites.

Third, the shopping assistant should protect the shoppers' privacy by concealing their identities and behavior from vendors and from the shopping assistant itself. However, this privacy would be revoked if users abuse it.

We first discussed these goals in the context of an early model and prototype of IntelliShopper.[4] Here we describe a more advanced model and implementation, which has benefited from many lessons learned through the early prototype. We focus on how IntelliShopper presents shoppers with new information, customized on the basis of heterogeneous and dynamic profiles that the agent learns by observing users' online shopping behavior.

IntelliShopper architecture

The current IntelliShopper prototype resides at http://myspiders.biz.uiowa.edu/~fil/intellishopper. Figure 1 illustrates IntelliShopper's logic and high-level architecture.

The following numbers (see Figure 1) indicate the sequence of shopping-assistance activities:

1. The user creates an account and one or more personae.
2. The user takes on a persona.
3. The persona initiates a shopping ses-

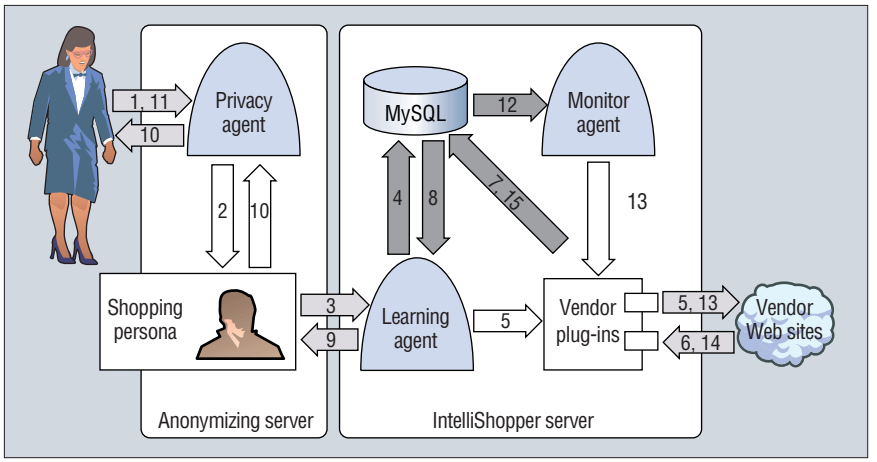


Figure 1. The IntelliShopper architecture. Light-gray arrows represent Web interactions based on the HTTP protocol. Dark-gray arrows represent SQL-based interactions. White arrows represent internal logic. Numbers indicate the sequence of shopping-assistance activities (see the main text for an explanation).

sion by submitting a query to the learning agent.
4. The learning agent stores the user's request in the database.
5. The learning agent uses vendor plug-ins to send requests to vendors.
6. Results from vendors are parsed through the vendor plug-ins.
7. IntelliShopper stores the results in the database.
8. The learning agent uses the persona profile to rank the hits.
9. The learning agent presents the results to the persona.
10. The privacy agent forwards the results to the user.
11. The user can further interact with the learning agent.
12. The monitor agent loads standing queries from the database.
13. The monitor agent uses the vendor plug-ins to autonomously check for any new results from the vendors.
14. IntelliShopper parses new and updated hits.
15. IntelliShopper stores the hits in the database until the user logs in for a new interactive session.

Steps 1 through 11 describe the basic interaction loop, during which the learning agent observes the persona's actions and updates the profile accordingly. Steps 12 to 15 occur offline.

The privacy agent lets the user take on a shopping persona and hides all identifying user information from the rest of the system. It resides on an anonymizer—one or more servers between the user and the IntelliShopper server. These servers forward requests and responses, and anonymize by permutation, stripping of IP addresses, encryption, and decryption.[1] Users can communicate privately because the anonymizer hides from merchants information such as the IP address from which requests emanate and to which responses are directed. In a standard implementation, a number of servers might act together as the anonymizing agent. In such a case, the anonymizer would work even if all but one of the servers were compromised. So, for efficiency's sake, our prototype uses only one anonymizing server. This is a special (degenerate) case in which the server itself is the privacy agent. Users must be able to trust this server; it should not have any commercial relation to vendors or other parties wishing to determine shoppers' identities. We assume that the user's privacy is to be protected only during shopping—the customer will not conduct purchases via IntelliShopper. (For more on personae and privacy, see the "Privacy" sidebar.) A shopping persona is a unique identity that reflects

Privacy

There is a strong connection between our concept of shopping personae and well-known cryptographic techniques. Our personae allow fine-level customization across heterogeneous and dynamic shopping preferences. Similarly, established cryptography can confer various types and degrees of privacy to online shoppers through the use of *pseudonyms*.[1]

Users may adopt different personae to hide their identities from IntelliShopper and other parties. From a privacy perspective, a persona is a pseudonym that a user employs for a particular type of activity that he or she wishes to separate (decorrelate) from other activities. However, if a user employs a particular pseudonym or persona for a long time, it becomes part of the user's identity. In fact, the possibility of linking a pseudonym to the real identity only once (with some reasonable probability) is sufficient for the association to remain. So, privacy requires that users migrate between personae; the migration frequency depends on the degree of privacy users desire.

On the other hand, the learning agent's performance and the customization process's usefulness depend on the use of relatively long-lived personae. To balance these requirements, users may obtain descriptors that label their shopping behavior, allowing them to submit these along with new personae. Such descriptors, however, must not be detailed enough to allow strong cross-pseudonym correlations. Furthermore, multiple pseudonym updates should be performed at the same time, preferably by a large portion of the user population each time.

For more information on shopping-agent privacy, see the paper on IntelliShopper.[2]

References

1. R. Arlein et al., "Privacy-Preserving Global Customization," *Proc. 2nd ACM Conf. Electronic Commerce* (EC 2000), ACM Press, New York, 2000, pp. 176–184.

2. F. Menczer et al., "IntelliShopper: A Proactive, Personal, Private Shopping Assistant," *Proc. 1st ACM Int'l. Joint Conf. Autonomous Agents and MultiAgent Systems* (AAMAS 2002), ACM Press, New York, 2002, pp. 1001–1008.

a particular user's mode of use. Its public descriptors are independent of the owner's identity, location, and so on. The persona becomes the "public user" that the IntelliShopper system components see and could be disclosed to merchants without compromising the user's identity. IntelliShopper indexes its history-based preference database by persona descriptors rather than by user names, IP addresses, cookies, or other identifying information that current commercial notification and brokering systems use. The persona has two explicit purposes. First, it protects the user's privacy—the IntelliShopper database never stores any identifying personal information about the user. Second, it allows for customization of multiple heterogeneous user profiles so that the user can adopt different personae for different shopping needs (for instance, "gadget geek" versus "loving spouse"). IntelliShopper can learn different shopping preferences for each persona. Figure 2a shows the interface for adding a new persona.

When a user logs in, IntelliShopper displays the profiles it learned (see Figure 2a and 2b) on the basis of the current persona's shopping history. IntelliShopper displays the shopper's history, with live links to outstanding queries for which the monitor agent has found new hits.

Additionally, the user can submit a new shopping request via the query interface (see Figure 3) rather than learning each vendor site's format. For example, one auction site might report the absence of bids as "0" and another as "—." Auction sites also use many different formats to display the time remaining in an auction; the format sometimes varies even within a single site. A site might indicate that an auction ends "at 6:30PM" at one time and "in 10 minutes" at a later time for the same item. IntelliShopper converts all these formats to common data domains before it stores each feature's value in the database. (The time format shown in the interface is taken directly from the sites; the agent stores a normalized time remaining for each of the vendor results.) Once IntelliShopper has received the results from the various vendors and collated, parsed, and stored them in the database, the learning agent ranks them according to the persona's profile and presents them to the user (see Figure 3b).

Vendor plug-ins let IntelliShopper interface with online stores and auctions. From IntelliShopper's perspective, interfacing with a vendor has two aspects: submitting queries and parsing results. Submitting queries is simpler; it consists of identifying an appropriate form, submission protocol, and input syntax on each vendor site. Parsing results is more difficult because it consists of identifying items and extracting feature values for all desired features (product description, price, and so on). Vendors could readily simplify this task (by using XML-based output, for instance). However, many vendors are uninterested in price competition, so they use complex, changing HTML markup to make it difficult for shopping bots to extract information from their sites. Some vendors even exclude bots.

Research is ongoing in the development of intelligent wrappers that could automate submitting queries and parsing results. In fact, there is an "arms race" between the intelligent wrappers that shopping bots employ and the growing complexity of HTML interfaces. Rather than trying to build automatic wrappers, we simplified the task of hand-coding wrappers by designing a language for speci-

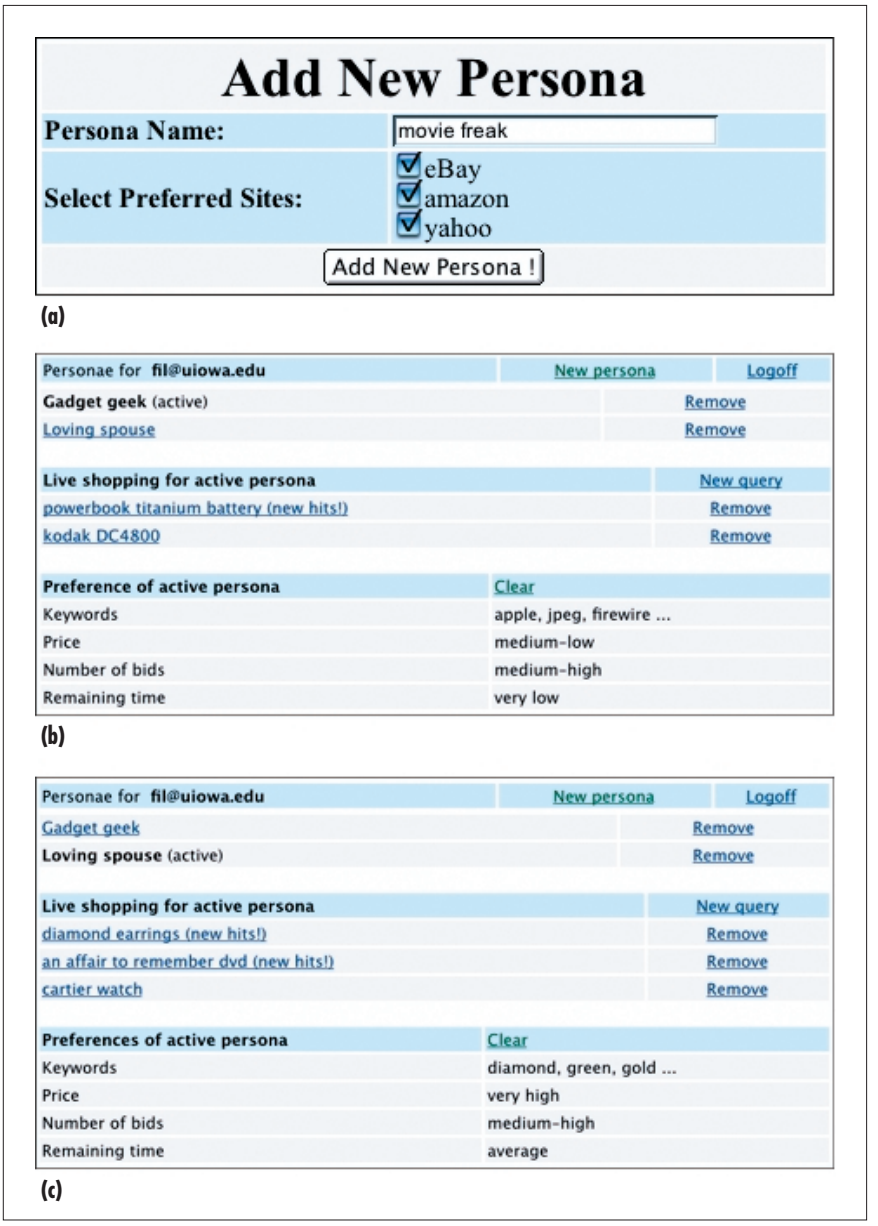
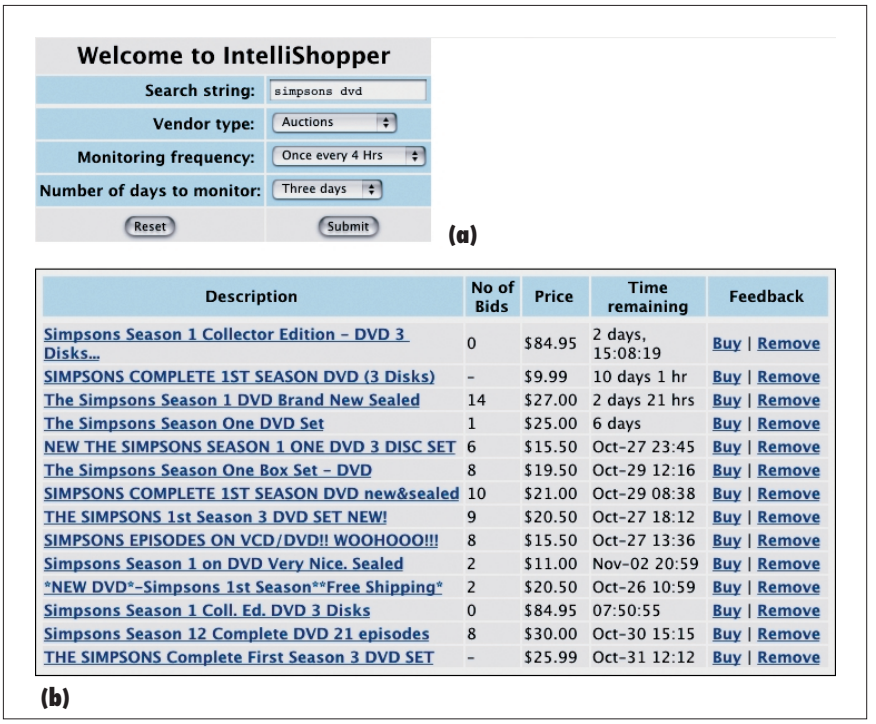


Figure 2. Handling multiple personae: (a) IntelliShopper's interface for adding a new persona; (b) summaries of shopping profiles for a user's two personae.



Description	No of Bids	Price	Time remaining	Feedback
Simpsons Season 1 Collector Edition – DVD 3 Disks...	0	$84.95	2 days, 15:08:19	Buy | Remove
SIMPSONS COMPLETE 1ST SEASON DVD (3 Disks)	-	$9.99	10 days 1 hr	Buy | Remove
The Simpsons Season 1 DVD Brand New Sealed	14	$27.00	2 days 21 hrs	Buy | Remove
The Simpsons Season One DVD Set	1	$25.00	6 days	Buy | Remove
NEW THE SIMPSONS SEASON 1 ONE DVD 3 DISC SET	6	$15.50	Oct-27 23:45	Buy | Remove
The Simpsons Season One Box Set – DVD	8	$19.50	Oct-29 12:16	Buy | Remove
SIMPSONS COMPLETE 1ST SEASON DVD new&sealed	10	$21.00	Oct-29 08:38	Buy | Remove
THE SIMPSONS 1st Season 3 DVD SET NEW!	9	$20.50	Oct-27 18:12	Buy | Remove
SIMPSONS EPISODES ON VCD/DVD!! WOOHOOO!!!	8	$15.50	Oct-27 13:36	Buy | Remove
Simpsons Season 1 on DVD Very Nice. Sealed	2	$11.00	Nov-02 20:59	Buy | Remove
NEW DVD–Simpsons 1st Season**Free Shipping*	2	$20.50	Oct-26 10:59	Buy | Remove
Simpsons Season 1 Coll. Ed. DVD 3 Disks	0	$84.95	07:50:55	Buy | Remove
Simpsons Season 12 Complete DVD 21 episodes	8	$30.00	Oct-30 15:15	Buy | Remove
THE SIMPSONS Complete First Season 3 DVD SET	-	$25.99	Oct-31 12:12	Buy | Remove

Figure 3. IntelliShopper queries: (a) the query interface; (b) query results based on the current persona's shopping profile.

fying vendor-dependent logic. This way, new vendor plug-ins can be written in minutes. IntelliShopper can integrate a new vendor any time a new plug-in appears in the appropriate directory. The current prototype has plug-ins for eBay, Yahoo, and Amazon auctions.

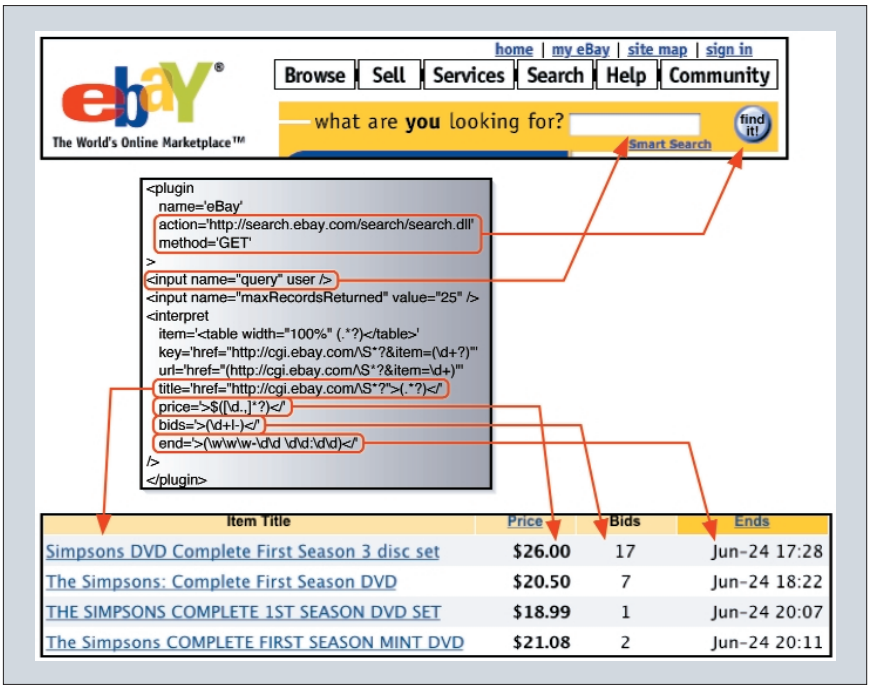


Figure 4. A simplified example of a vendor plug-in, which has logic for submitting queries (the top half) and interpreting the results (the bottom half).

Figure 4 exemplifies the plug-in language, which is based on XML and is inspired by Apple's Sherlock engine. For querying, there are tags with fields that specify the form's URL, the submission protocol (GET/POST), and the necessary input parameters. For parsing, there is a tag with fields that specify feature names and Perl regular expressions that extract the corresponding feature values. The representation makes updating plug-ins easy. However, because vendor site design changes rapidly, this approach would be significantly more robust if vendors were to use an open interface. An open interface would allow such changes to take place automatically rather than manually.

IntelliShopper stores data about its shopper personae and their profiles, queries, product hits, and features in a relational database. Figure 5 is an entity-relationship diagram that outlines the database design. The Preference and Keyword entities store each persona's profile, which IntelliShopper learns by observing the persona's shopping behavior and uses to customize the information it presents to the user. We denormalized the database schema for improved efficiency; we merged into a single table the Item and Feature entities that store information about hits and user feedback.

Customization

IntelliShopper adapts to user preferences to better rank hits with continued use. Our approach is based on gathering maximum information while requiring minimum user feedback. Information-filtering and Internet-recommendation systems have widely employed the idea of learning user behaviors by "looking over the user's shoulder."[5,6] IntelliShopper presents information to the user in a way that lets the learning process easily incorporate the user's actions. The system increases the rankings of hits similar to those that previously interested the user and reduces the rankings of hits similar to items that the user has ignored or actively disliked.

Our adaptation scheme, as is typical in inductive machine learning, is based on a collection of features extracted from the hits. IntelliShopper chooses features that might be relevant to the user's evaluation of the item. Features can be either numerical or textual. For instance, the current IntelliShopper prototype uses these numerical features for auction sites: price, number of bids, and time remaining in the auction. Textual features are keywords that appear in product descriptions.

For each numerical feature x, we maintain a distribution of *temperatures* across the range of possible values. A feature–value pair's temperature should correspond with the user's desire for an item with that characteristic; high temperature signifies a desirable value, low temperature an undesirable one. We maintain temperatures for each possible value of discrete features; for example, a "color" feature might have a low temperature for the value "pink." Continuous variables are discretized; the "price" feature might have a high temperature for the value "very low." Cut-off points for the discretization are based on the mean μ and standard deviation σ of feature values observed among hits:

- *Very low* ($x < \mu - (3/2)\sigma$)
- *Medium low* ($\mu - (3/2)\sigma \le x < \mu - (1/2)\sigma$)
- *Average* ($\mu - (1/2)\sigma \le x < \mu + (1/2)\sigma$)
- *Medium high* ($\mu + (1/2)\sigma \le x < \mu + (3/2)\sigma$)
- *Very high* ($\mu + (3/2)\sigma \le x$)

For textual features, we maintain a vector of keywords associated with each persona profile. The agent extracts keywords from product descriptions. (It removes very common word endings and "noise" words such as "the," and conflates the remaining terms with a standard stemming technique used in information retrieval.)[7]

IntelliShopper updates temperatures for feature values and keywords after any user action related to a given hit. The temperature associated with a hit's feature values and description keywords follows a simple update rule: $T(t + 1) = (1 - \alpha)T(t) + \alpha\Delta T$, where the learning rate α ($0 \le \alpha \le 1$) determines how quickly a profile forgets old preferences and tracks new ones. (We set $\alpha = 0.25$ in the current prototype.)

Any hit has five possible actions (or inactions), each with its own effect on ΔT for the corresponding keywords and feature values:

- *Buy:* Clicking on the Buy option is strong positive feedback; it results in a temperature increase $\Delta T = +2$ for all of that item's feature values.
- *Browse:* Clicking on the item description is weak positive feedback; the temperature increase is $\Delta T = +1$.
- *Ignore:* If a user does not get far enough down the list to assume that he or she looked at an item, no inference is made ($\Delta T = 0$).
- *Skip:* Bypassing an item (that is, clicking on one farther down the hits list) is weak negative feedback: $\Delta T = -1$.
- *Remove:* Actively deleting an item is strong negative feedback: $\Delta T = -2$.

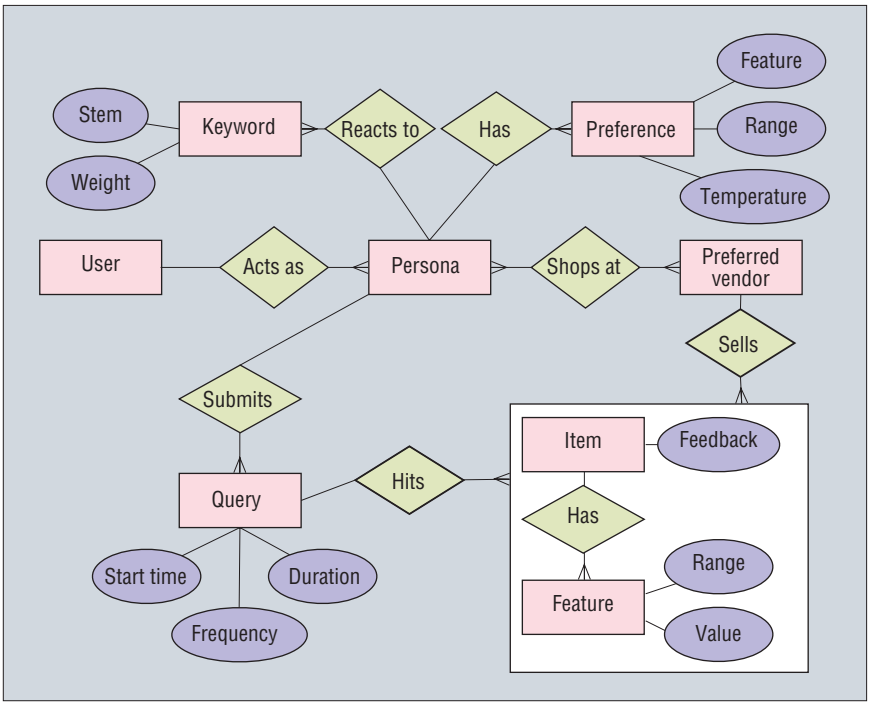


Figure 5. A simplified entity-relationship model of the IntelliShopper database.

There are certain precedence rules in assigning feedback. For example, if a user first browses and then buys or removes an item, we apply the stronger feedback ($\Delta T = \pm 2$). IntelliShopper assigns the Skip action only if it receives no other feedback for an item.

Figure 6 illustrates the profile update algorithm. The learning agent adjusts the persona profile after each user action. IntelliShopper uses the profile to customize the information it presents the user by ranking the hits according to a simple sum of the temperatures for their feature values and their description keywords. User interactions during a shopping session cause IntelliShopper to re-rank the hits on the basis of the updated profile.

In the case in Figure 6, the user clicks to buy the second item. From this action, the learning agent infers a strong interest for high-priced items and a mild disinterest for average-priced items (because the user skipped the first hit). The learning agent updates the corresponding price ranges' temperatures accordingly. Similarly, when focusing on keywords, the learning agent infers a strong interest for the terms "illi," "caff," "import," and "itali" (the stemming algorithm changes "italy" to "itali") and a mild disinterest for the terms "lavazza," "oro," "vacuum," and "pack." Once the learning agent updates the temperatures (weights) of these terms, it crops the keyword vector to retain only the 32 terms with highest and lowest T. For legibility, this figure denotes only three possible values for numerical features and only three terms in the keyword vector.

Evaluation

Evaluating an agent such as IntelliShopper is difficult because performance measures are subjective. Ideally, we should compare user satisfaction between shoppers using IntelliShopper and shoppers using other shopping agents, which is problematic for a number of reasons (different functionalities, survey bias, cost, and so on). Fortunately, evaluation is relatively straightforward if we limit it to the system's performance in customizing shopping information on the basis of learned persona profiles.

The goal is an evaluation measure that is quantitative, objective, and based on data from real users. So, we asked eight volunteers to use IntelliShopper during actual shopping sessions. During each session a subject could take on any persona, submit requests, examine new hits for previous

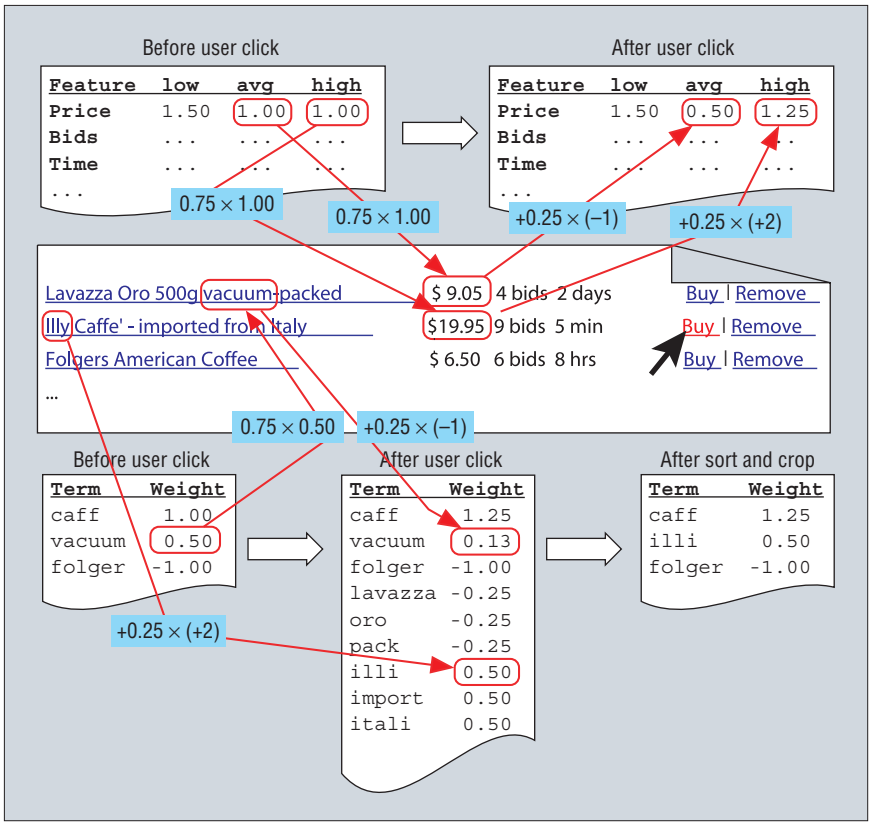


Figure 6. How the learning agent changes a profile following a user action. When the user clicks on the second item, the learning agent updates the temperatures of the price ranges and keywords. IntelliShopper crops the keyword vector so that only the most discriminating terms are retained.

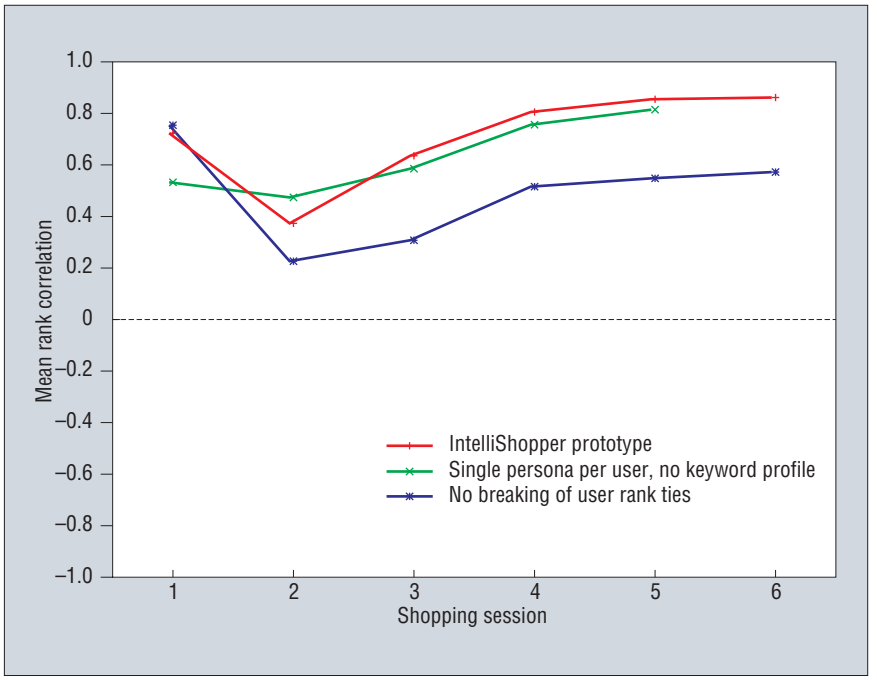


Figure 7. The Spearman correlation between IntelliShopper's ranking and ranks inferred from user feedback.

requests, and interact with the learning agent through the IntelliShopper user interface. We recorded all user requests, hits, and feedback, along with two rankings of all the hits in each session. The system used the first ranking to display hits based on learned user profiles. It computed the second ranking on the basis of the feedback it inferred from user actions during each session. For the hit set corresponding to each **(persona, session, query)** tuple, we measured the Spearman's rank correlation coefficient between these two rankings. The formula is as follows:

$$\rho = 1 - 6\frac{\sum_{i=1}^{n}\left(r_{IS}(i) - r_u(i)\right)^2}{n\left(n^2 - 1\right)},$$

where r_{IS} is the rank based on IntelliShopper's learned profile, r_u is the rank based on user feedback, and n is the number of hits.

If the shopping assistant is effective, the correlation between the ranks the system learned and the ranks it inferred from user feedback should be positive. Furthermore, if a persona has a consistent shopping behavior, the learning agent should be able to internalize this behavior, so the correlation should increase over multiple sessions.

Figure 7 plots the mean Spearman's rank correlation coefficient against the number of shopping sessions. In this case, the subjects submitted 42 queries as 15 distinct personae. The experiment involved 4,759 distinct hits. For every query in each session for each persona, we computed a correlation coefficient across hits; then we averaged these coefficients across personae and queries for each session. We plot the resulting mean rank correlation for sessions in which at least 10 **(persona, query)** measurements are available. For a set of hits whose feedback-based ranks are tied, we can break the tie optimistically (according to IntelliShopper's ranks). We used this method to evaluate an early prototype in which neither multiple personae per user nor keyword-based profiles were yet implemented; that data[4] is based on a larger number of subjects (51) and queries (97) than in the current example, but fewer sessions and hits. We obtained lower (but still positive) correlation values without breaking ties in ranking.

The positive correlation in Figure 7 shows that the shopping assistant effectively customizes product information to show users, even when users have not seen the products

before. IntelliShopper requires a few sessions to learn persona profiles that are sufficiently general. The initial dip in performance is because a profile learned from feedback on a single query does not appropriately predict user preferences for different queries. The presence of keyword information in persona profiles exacerbates this "learning curve." The keyword features that effectively rank hits for a single query are often too specific to capture a persona's general preferences. After three sessions, the keywords in the profile effectively predict user behavior, and after four to five sessions IntelliShopper's learning agent reaches a stable tracking regime.

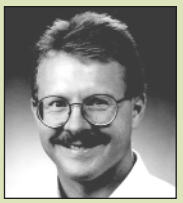
The customization advantages of agents such as IntelliShopper stem from a business model in which vendors may benefit economically from becoming friendly to shopping agents that put the customer at the center of the business relationship. IntelliShopper is based not only on price competition but also on several other factors. We believe this is a better business model than price-only bots and comparison-shopping agents that are biased by vendor memberships. The first generation of user-centered shopping bots did not survive the transition to the commercial realm; the sophistication of future online shoppers will decide the fate of the new generation of shopping assistants.

We plan to implement a few extensions and improvements to the system before IntelliShopper is ready for prime time. We will add more simple features (such as brand name), subject to our ability to consistently extract them from product hits. We can make the temperature-update procedure adaptive so that, for instance, the price feature will weigh more heavily in ranking hits for a user that often makes decisions based on price. We can also tune the learning rate adaptively to track dynamic profiles efficiently. The user interface will be improved as well. For example, we plan to allow users to choose multiple items to remove simultaneously from the hits list. Such an action will reduce the number of database transactions and will render feedback to the learning agent more consistent. We intend to study the opportunities for collaborative filtering that stem from centralized shopping assistants such as IntelliShopper. Given the established effectiveness of collaborative filtering, we may extend hit sets at the user's option by clustering personae with similar profiles. ❐

The Authors

Filippo Menczer is an assistant professor in the University of Iowa's Department of Management Sciences and a faculty member of the university's Program in Applied Math and Computational Sciences. His interdisciplinary research interests include Web, text, and data mining; Web intelligence; distributed information systems; adaptive intelligent agents; evolutionary computation; machine learning; neural networks; complex systems; artificial life; and agent-based computational economics. He received his Laurea in physics from the University of Rome and a dual PhD in computer science and cognitive science from the University of California at San Diego. He has received a National Science Foundation CAREER award as well as Fulbright, Rotary Foundation, and NATO fellowships, and is a Santa Fe Institute fellow at large. Contact him at S320 Pappajohn Business Bldg., Univ. of Iowa, Iowa City, IA 52242; filippo-menczer@uiowa.edu.

Alvaro E. Monge is an associate professor in the Department of Computer Engineering and Computer Science at California State University, Long Beach. His research interests include information retrieval, knowledge discovery, database system integration, software agents, data warehousing, and computer science education. He earned his BS in computer science from the University of California at Riverside and his MS and PhD in computer science from the University of California at San Diego. He is a member of the ACM, ACM SIGMOD, ACM SIGCSE, and ACM SIGAPP. Contact him at California State Univ. Long Beach, CECS Dept, 1250 Bellflower Blvd., Long Beach, CA 90840-8302; monge@cecs.csulb.edu.



W. Nick Street is an associate professor in the University of Iowa's Management Sciences Department. His research interests are in machine learning and data mining—particularly the use of mathematical optimization in inductive learning techniques. His recent work has focused on dimensionality reduction (feature selection) in high-dimensional data for classification and clustering; ensemble prediction methods for massive and streaming data sets; and learning shapes for image segmentation, classification, and retrieval. He received his PhD in computer sciences from the University of Wisconsin. He has received a National Science Foundation CAREER award and a National Institute of Health Individual National Research Service Award postdoctoral fellowship. He is a member of the IEEE, the ACM, the AAAI, and the Institute for Operations Research and the Management Sciences. Contact him at S232 Pappajohn Business Bldg., Univ. of Iowa, Iowa City, IA 52242; nick-street@uiowa.edu.

Acknowledgments

We are very grateful to Markus Jakobsson for many helpful suggestions and discussions on the privacy aspects of shopping agents. Narayan Vishwakarma contributed to the early prototype implementation. Thanks to the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) for support and to the volunteers who helped with the evaluation. The server hosting the IntelliShopper prototype was made available by a University of Iowa instructional improvement award.

References

1. M. Jakobsson and M. Yung, "On Assurance Structures for WWW Commerce," *Proc. 2nd Int'l Conf. Financial Cryptography*, Lecture Notes in Computer Science, no. 1465, Springer-Verlag, Heidelberg, 1998, pp. 141–157.
2. J.O. Kephart and A.R. Greenwald, "Shopbot Economics," *Autonomous Agents and Multi-Agent Systems*, vol. 5, no. 3, September 2002, pp. 255–287.
3. A.R. Greenwald and J.O. Kephart, "Shopbots and Pricebots," *Proc. 16th Intl. Joint Conf. Artificial Intelligence* (IJCAI 99), Morgan Kaufmann, San Francisco, 1999, pp. 506–511.
4. F. Menczer et al., "IntelliShopper: A Proactive, Personal, Private Shopping Assistant," *Proc. 1st ACM Int'l. Joint Conf. Autonomous Agents and MultiAgent Systems* (AAMAS 2002), ACM Press, New York, 2002, pp. 1001–1008.
5. T. Joachims, D. Freitag, and T. Mitchell, "WebWatcher: A Tour Guide for the World Wide Web," *Proc. Int'l. Joint Conf. on Artificial Intelligence* (IJCAI 97), Morgan Kaufmann, San Francisco, 1997, pp. 770–777.
6. H. Lieberman, "Autonomous Interface Agents," *Proc. ACM Conf. Human Factors and Computing Systems* (CHI 97), ACM Press, New York, 1997, pp. 67–74.
7. Martin F. Porter, "An Algorithm for Suffix Stripping," *Program*, vol. 14, no. 3, July 1980, pp. 130–137.