IntelliShopper:
A Proactive, Personal, Private Shopping Assistant

Filippo Menczer
filippo-menczer@uiowa.edu

W. Nick Street
nick-street@uiowa.edu

Narayan Vishwakarma
nvishwakarma@yahoo.com

Department of Management Sciences
The University of Iowa
Iowa City, IA 52242

Alvaro E. Monge
Department of CECS
Cal. State Univ. Long Beach
Long Beach, CA 90840
monge@cecs.csulb.edu

Markus Jakobsson
RSA Laboratories
20 Crosby Drive
Bedford, MA 01730
mjakobsson@rsasecurity.com

ABSTRACT

The *IntelliShopper* is a shopping assistant designed to empower consumers. It is a *personal* assistant in that it observes the users while shopping and learns their preferences with respect to various features that characterize shopping items. It is *proactive* in that it remembers the users' requests and autonomously monitors vendor sites for new items that might match the users' needs and preferences. Finally, it protects users' *privacy* by means of pseudonymity, IP anonymizing, and trusted filtering. Pseudonymity is achieved through the use of personae; we show that this approach also behooves successful classification. IP anonymizing can be performed in at least two manners, which we discuss and compare in the context of our application. Trusted filtering — as opposed to merchant-based filtering — improves privacy by allowing users to select their preferred privacy representative. This paper introduces the IntelliShopper system, discusses its architecture and components, describes a prototype implementation, and outlines preliminary evaluations of its performance.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*User profiles and alert services*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering, Relevance feedback*; E.3 [**Data Encryption**]: *Public key cryptosystems*

Keywords

Shopping, Pro-activity, Monitoring, Personalization, Learning, Privacy

1. INTRODUCTION

E-commerce has changed the way companies distribute their products and services to consumers. Traditional brick-and-mortar companies continue growing this segment of the economy by creating their own e-commerce presence. Some companies have created (or reshaped) their image by having their entire operations based strictly on e-commerce. An e-commerce strategy has many benefits for the company as well as the consumer. In the research presented here, we aim at improving the accessibility and expanding the benefits of e-commerce shopping to consumers and at aiding the move to a personalized (and thus more efficient) marketplace.

The success of e-commerce has resulted in problems analogous to those earlier created by the growth and popularity of the Internet. Search engines were an early solution to the problem of finding information spread out over many different Web sites. Similarly now, a shopper must sift through the information provided by innumerable e-commerce sites. This task is a difficult one as the type, amount, and organization of the information provided on e-commerce sites differs from company to company. Complicating matters, a customer goes unaware of changes in pricing, availability, etc. unless she revisits the sites very frequently.

There are a number of things that shoppers can currently do when looking for a product. The most straightforward approach is to manually visit various e-commerce sites; for each site, the shopper browses and/or searches for the particular product of interest. This simple approach has several drawbacks. First, there is no single site that caters to all shopping needs, which increases the (user) search time for each new product category. Also, getting acquainted to new vendor interfaces slows down the user browsing and hinders impulse shopping. Finally, it is an approach likely to favor only the largest vendors (due to name-branding), which in turn reduces the effectiveness of the market.

A second approach increases the degree of automatization by site-provided alert services. Several services allow shoppers to sign up to receive price alerts that notify a shopper when the price of a product changes or falls below a specified amount. Some of the services require lengthy surveys to be filled out, while at the same time most provide little to no personalization. A further drawback of this approach is the weakening of user privacy that it implies.

A third proposed approach involves voluntary ratings and reviews of vendors and products by users, and the compilation of such information [9]. As in the previous approach, such recommendation systems are likely to reduce the size of the marketplace and to introduce bias, as it is difficult to obtain a sufficient number of ratings for every existing vendor, and to control the reliability of the sources.

Finally, a fourth general approach is to further automate and generalize the search process [11, 7]. As early as 1995, shopping agents (also referred to as comparison shopping agents) were proposed as a solution to find a product under the best terms (where price was the most important feature early on) among different e-commerce sites. A shopping agent queries multiple sites on behalf of a shopper to gather pricing and other information on products and services. Most of these comparison shopping agents however present a marketplace that is biased in favor of the e-commerce sites that collaborate with the shopping agent. In addition to the biased marketplace, a shopper has only a limited number of e-commerce sites to choose from and often the participating sites do not offer the best prices.

We propose a new type of shopping agent, called *IntelliShopper,* that extends the above approaches by providing the user with autonomy, personalization, and privacy.

Autonomy refers to the idea that a shopping agent can provide the best possible service by remaining as independent as possible from both customers and vendors. Autonomy from vendors implies that the service is to remain unbiased by performing wide searches (as opposed to only searching the databases of a few "preferred" vendors). This can be achieved by progress in making interfaces more uniform, and by improved methods for interpreting potential hits. Autonomy from the customer means that users can be relieved from the tedious task of searching for information and of needing to adjust to different e-commerce sites. Furthermore, our shopping agent proactively monitors vendor sites on behalf of the user, notifying him of new products of potential interest.

Personalization means that the shopping assistant strives to learn the user behaviors and preferences by observing his actions while shopping. When a user considers the items available at e-commerce sites, he indirectly provides feedback by clicking on items. The agent can internalize this feedback to infer user preferences and apply such learned knowledge in taking initiative about future searches, as well as in predicting when a user might be interested in an item, so that the user can be notified.

Finally, our research addresses the *privacy* of the shopper by concealing the identity and behavior of the user in a variety of ways. However, we note that the privacy provided is conditional, and should be selectively revoked if abuse is suspected. The personalization and privacy aspects of our proposed agent provide for an unbiased personalized marketplace where the user benefits in many respects.

2. BACKGROUND

Research in the area of shopping agents dates back to the early years of the Web. In 1995, Andersen Consulting developed BargainFinder, the first of the shopping agents. It allowed users to compare prices of music CDs from stores selling over the Internet. At the time however, some of the retailers blocked access because they did not want to compete on price, and BargainFinder ceased operation. Since then, there have been additional shopping agents that started providing unbiased comparison of products from different shopping sites. In PersonaLogic, users created preference profiles to describe their tastes. The approach allowed for the identification of products with features important to the users, but the vendors had to provide an interface that explicitly disclosed the features of the products in a way that could be matched with user profiles. PersonaLogic was acquired by AOL in 1998 and the technology disappeared.

Ringo was an agent that recommended entertainment products (music, movies) based on collaborative filtering, i.e., on opinions of like-minded users [3]. This was one of the earliest software agent technologies to be commercialized, when it was incorporated into a company named FireFly. FireFly also addressed the issue of privacy by initiating and promoting the P3P standard. FireFly was acquired by Microsoft in 1998 and the FireFly agent ceased operation shortly thereafter. However the concept of collaborative filtering has become widely used, including — in simplified ways — by large commercial vendors such as Amazon.

The ShopBot was an agent that could learn how to submit queries to e-commerce sites and interpret the resulting hits to identify lowest-priced items [4]. ShopBot automated the process of building "wrappers" to parse semistructured (HTML) documents and extract features such as product descriptions and prices. Our goals are similar but we focus on learning the user preferences (with respect to many features) and we use a different approach for extracting those features from vendor sites. The ShopBot technology had a similar fate to those of PersonaLogic and FireFly; it was acquired and commercialized by Excite (under the name Jango), and soon replaced with a biased vendor-driven agent.

Tete@Tete was an agent that integrated product brokering, merchant brokering, and negotiation [15]. A startup called Frictionless Commerce is applying the technology to B2B markets (e-sourcing) rather than to B2C markets. The only comparison shopping agents available to consumers that are surviving in the commercial realm are biased, presenting results only from companies with whom they collaborate. Examples include MySimon, DealTime, PriceScan, RoboShopper, and many others.

Learning user behaviors and preferences by "looking over the user's shoulders" is an example of an interface agent. These have been widely employed in information filtering and Internet recommendation systems [14]. Two user interface agents that learned from the actions taken by a user are Letizia [12] and WebWatcher [10]. Similarly to these agents, IntelliShopper presents information to the user in a way that allows her interaction to be easily incorporated into the learning process.

In the area of Web querying and monitoring, the most relevant work is WebCQ [13]. In WebCQ, specific pages can be monitored for changes to their content. The system can track changes on arbitrary pages by computing the difference between the page at some given time and the same page at a later time. More recently research has begun on the monitoring of XML data [17]. Here, the structuring of documents allows for database-precision queries. In this particular work, the focus is on the architecture of a scalable system that supports the monitoring of millions of pages per day serving millions of subscribers to the monitoring service.

A variety of well-known cryptographic techniques are applicable to preserve the privacy of online shoppers. To protect the identity of users, a type of pseudonym called a *persona* can be used (see [1] for a discussion of the type and degree of privacy obtained with this technique). An *anonymizer*[1] (a.k.a. *mix network* or *onion router*) can be used to obtain communication privacy, i.e., to hide the IP address from which requests emanate and to which responses are directed. An anonymizer consists of one or more servers located between the user and the merchants; these servers forward requests and responses, and anonymize by means of permutation, stripping of IP addresses, and encryption/decryption. We refer to [9] for a detailed description of how to integrate these tools in an architecture such as ours.

An alternative to anonymizers is the "Crowds" approach, which can be thought of as a "buyers club" for privacy [18]. In practice, this is implemented by one or more users passing each other's requests back and forth to hide their origin from a recipient/merchant. Such an approach is slightly less palpable than anonymizers in an architecture like the one proposed in this paper, but is still possible to use.

An alternative to hiding the identity and the whereabouts (IP address) of the users is to hide what they request. In particular, it is possible to design a system in which users can access elements of a database in a manner that makes it impossible to determine what elements were accessed [6]. However, such an approach (naturally) prohibits the central collection of statistics, forcing all the filtering to be performed by the user. Furthermore, this approach makes user mobility difficult and collaborative filtering impossible. We therefore do not consider techniques of this type.

3. SYSTEM ARCHITECTURE

Figure 1 illustrates the high-level architecture of the IntelliShopper system. The *privacy agent* allows the user to take on a *shopping persona* and hides all identifying information about the user (e.g., IP address, username, and email) from the rest of the system. The privacy agent resides on one or more collaborating servers, located between the user and the IntelliShopper server. It is possible for several privacy agents to co-exist, allowing each user to select the one in which he has the most confidence. It is also possible for the user to employ different privacy agents for different functions, personae or requests. This selection can be made locally, on the user's client, or by a first privacy agent server whose main function is to keep and maintain the selection policies.

In this model it is assumed that the customer will not conduct purchasing transactions via the IntelliShopper; if the user's privacy is to be protected throughout the buying process as well as while shopping, then a privacy agent (acceptable to the merchants) would have to reside between the user (or the user's purchasing agent) and the merchants. We do not further discuss this issue in the present paper.

A shopping persona is a unique identity reflecting the

[1]See www.anonymizer.com for a commercially proposed anonymizer.

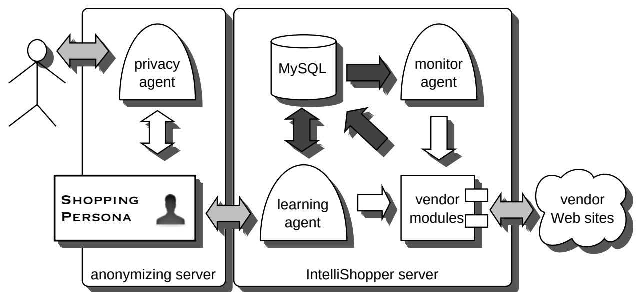


Figure 1: Architecture of the IntelliShopper system. Light gray arrows represent Web interactions based on the HTTP protocol. Dark gray arrows represent SQL-based interactions.

mode of use of a particular user. Its public descriptors are independent of the owner's identity, location, etc. The persona becomes the "public user" seen by the other components of the system, and could even be disclosed to merchants without compromising the user's identity. The IntelliShopper can build its history-based preference databases indexed by the persona descriptors, rather than by user names, IP addresses, or by using cookies as is currently done by commercial notification and brokering systems. The persona has two explicit purposes: (i) the privacy of the user is guaranteed, assuming the user trusts the anonymizing server, because no identifying personal information about the user is ever stored in the IntelliShopper database; and (ii) the user can take different personae for different shopping needs, e.g., "gadget geek" vs. "loving spouse," and IntelliShopper can learn a different shopping profile for each persona.

The remaining modules of the system are hosted on the main IntelliShopper server. They interact with users only via shopping personae. The *learning agent* takes user requests, saves them on the database, forwards them to online vendors, retrieves the resulting hits from the local database, and displays the results (ranked according to the persona profile) back to the user. The learning agent also observes the actions of the user — removing, viewing, and/or buying items — and adjusts the profile accordingly.

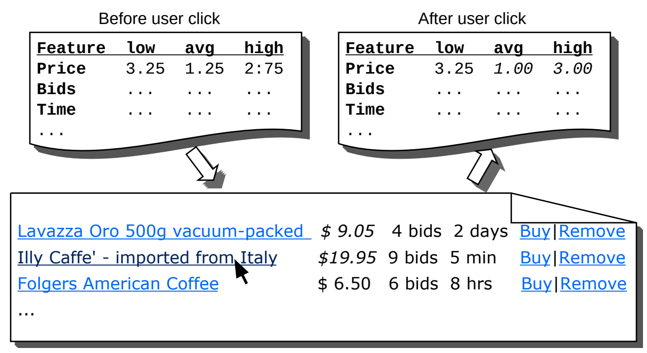
Logic about individual Web-based e-commerce and auction sites is stored in the *vendor modules.* These specify how to query vendors and how to interpret the results and extract structured information, such as product descriptions and prices, from the HTML pages returned by the vendor sites. The hits returned by the vendors are parsed, recorded in the database, and made available to the learning agent.

The *monitor agent* periodically queries the database to retrieve outstanding queries, i.e., shopping requests in which the user is still interested. At intervals specified by the user, the monitor agent queries vendors (via the vendor modules) to see if new items have appeared that might match the persona profile. The new hits are recorded in the database so that the user can be notified.

The following subsections illustrate the models behind the learning, monitor, and privacy agents in greater detail. Implementation issues are discussed in Section 4.

3.1 Learning Agent

IntelliShopper adapts to user preferences to better rank hits with continued use. Our approach is based on gath-


Before user click

Feature	low	avg	high
Price	3.25	1.25	2:75
Bids	...	...	...
Time	...	...	...
...			

After user click

Feature	low	avg	high
Price	3.25	*1.00*	*3.00*
Bids	...	...	...
Time	...	...	...
...			

Lavazza Oro 500g vacuum-packed	*$ 9.05*	4 bids	2 days	Buy|Remove
Illy Caffe' - imported from Italy	*$19.95*	9 bids	5 min	Buy|Remove
Folgers American Coffee	$ 6.50	6 bids	8 hrs	Buy|Remove
...				

Figure 2: Illustration of how the learning agent changes a profile following a user action. In this case, focusing on the price feature, the user clicks on the second hit, from which the learning agent infers a mild interest for high-priced items and a mild disinterest for average-priced items (since the first hit was skipped over). The temperatures of the corresponding price ranges are updated accordingly.

ering the maximum amount of information while requiring a minimum of extra feedback from the user. The system learns to increase the rankings of hits that are similar to those that have interested the user in the past, and reduce the rankings of those similar to items that the user has either ignored or actively disliked. The process is illustrated in Figure 2.

As is typical in inductive machine learning, our adaptation scheme is based on a collection of features extracted from the hits. Features are chosen such that they might be relevant to the user's evaluation of the item. Features can be either continuous or discrete, although all features currently used are continuous. Considering auction sites for example, the following features are used in the current model:

- price of the item,
- number of bids that have been placed on the item,
- time remaining in the auction, and
- similarity between the query and the item description. This is computed using the standard *cosine similarity:*

$$sim(q,d) = \frac{\sum_{k \in q \cap d} f_{kq} f_{kd}}{\sqrt{\left(\sum_{k \in d} f_{kd}^2\right)\left(\sum_{k \in q} f_{kq}^2\right)}}$$

  where q is the query, d is the item description, and f_{ki} is the frequency of term k in i.

For each feature x, we maintain a distribution of *temperatures* across the range of possible values. The temperature of a given feature/value pair should correspond with the user's desire for an item with that characteristic; high temperature signifies a desirable value, low temperature an undesirable one. Temperatures are maintained for each possible value of discrete features, e.g., a"color" feature might have a low temperature for the value "pink." Continuous variables are discretized, e.g., the "price" feature could have a high temperature for the value "low." Cutpoints for the discretization are based on the mean μ and standard deviation σ of feature values observed among hits, as follows:

- Very low: $x < \mu - 2\sigma$
- Low: $\mu - 2\sigma \leq x < \mu - \frac{\sigma}{2}$
- Average: $\mu - \frac{\sigma}{2} \leq x < \mu + \frac{\sigma}{2}$
- High: $\mu + \frac{\sigma}{2} \leq x < \mu + 2\sigma$
- Very high: $\mu + 2\sigma \leq x$

Note that Figure 2 denotes only three possible values for discretized continuous features.

Temperatures are updated on any user action (or inaction) related to a given hit. In Figure 2, the user clicks on the second item, which is high-priced. This causes a rise in the temperature for high price, i.e., the system considers this evidence that the user is interested in expensive coffee. The temperature associated with each feature value follows a simple update rule:

$$T(t+1) = \alpha_1 T(t) + \alpha_2 \Delta T \qquad (1)$$

where α_1, α_2 determine how quickly a profile forgets old preferences and tracks new ones.[2] There are four possible reactions to any given hit, each with its own effect on ΔT for the corresponding feature values:

- Buy: Clicking on the "Buy" option is considered strong positive feedback; it results in a temperature increase $\Delta T = +0.5$ for all the feature values of that item.
- Browse: Clicking on the item description is weak positive feedback; the temperature increase is $\Delta T = +0.25$.
- Ignore: If a user bypasses an item, as indicated by clicking on one farther down the list of hits, this is considered weak negative feedback: $\Delta T = -0.25$. The Lavazza Oro item in Figure 2 is one such example.
- Remove: Actively deleting an item is strong negative feedback: $\Delta T = -0.5$.

Hits that appear on the list below the last one with which the user interacted do not cause any temperature updates.

Hits are given a temperature based on a simple sum of the temperatures for the values of their features. The hits are then ranked according to their temperatures, from high to low. User interactions during a query session cause a re-ranking of the hits based on updated temperatures.

3.2 Monitor Agent

The monitor agent makes the IntelliShopper autonomous in that it proactively shops on behalf of users (or, more accurately, on behalf of personae). This agent is a background process that wakes up periodically and queries the databases for any requests not yet expired or removed by the users.

The user can specify the duration and frequency of shopping requests. For any outstanding request, the agent checks the frequency at which the user has requested to monitor vendors. If the time elapsed since the last check is longer than the one corresponding to the monitoring frequency, the agent queries the vendors again and updates the database with the new hits. New hits might include previously seen items whose characteristic feature values (say, price) have changed. The user (possibly notified via email) will find the

[2]We set $\alpha_1 = \alpha_2 = 1$ in the current prototype.

new results waiting the next time she logs into the IntelliShopper. The results will be ranked based on the profile of the shopping persona used to make the request. As the user looks at the new hits, the learning agent can get additional feedback and further adjust the persona profile.

3.3 Privacy Agent

A multitude of approaches must be taken to implement privacy in our system. Privacy could mean either information about who is performing a search, or what is being searched for. In order to uncouple the request from the requester, it is necessary to hide both his identity and whereabouts. (We do not consider the issue of hiding the *geographical* location of the user.)

The whereabout of the user — most typically the IP address — is best hidden by passing all requests through an anonymizer, which hides the origin of the request from the IntelliShopper agent, and the contents of the request from the anonymizer's servers.

Anonymizers are typically implemented via distributed control. The request may be a multiply encrypted message. For each anonymizer server, this gets partially decrypted, and passed along to the next server, accompanied by a set of other such requests. Each server takes a list of inputs and permutes and operates on these using decryption, re-encryption, or similar cryptographic operations [2, 8]. After the last such server has operated on the list, the corresponding outputs are forwarded to the IntelliShopper server. The latter replies by passing a message back to the user, employing some user-created temporary alias to address the reply as it is passed back through the privacy agent.

The privacy of anonymizers rely on at least *one* of the servers performing this task correctly, and without revealing its secret decryption and permutation information. In fact, for the sake of efficiency, we will use only one anonymizing server, as was performed in early re-mailers and in at least one commercially tested Web access anonymizing system [5]. The one-server anonymizer is a degenerate case in which the server itself is the privacy agent. This server must be trusted by the users; it should not have any commercial relation to merchants and other parties wishing to determine the identity of shoppers.

Most anonymizers require servers to have knowledge of some secret key for decryption corresponding to the public key used to encrypt the request [2, 19]; others require knowledge of the public key alone [8, 16]. This makes the latter type useful for *ad-hoc* anonymizers, where keys are not pre-assigned to servers and anybody may act as a server.

Users may use different pseudonyms or personae in order to hide their identity from the agent (and other parties). Here, we let a persona refer to a pseudonym that a user employs for a particular type of activity that she wishes to separate (de-correlate) from other activities. However, using the case of social security numbers as supporting evidence, it is clear that if a particular pseudonym or persona is used for a long time, then this becomes part of the user identity. In fact, the possibility to link a pseudonym to the real identity only *once* (with some reasonable probability) is sufficient in order for the association to remain. Therefore, it is important for users to migrate between pseudonyms over time, where the migration frequency depends on the degree of privacy desired. Note, however, that the precision of the search depends on the use of (somewhat continuous) user naming. In order to balance these requirements against each other, users may obtain descriptors that label their search behavior, allowing them to submit these along with new personae. Note that the descriptors must not be detailed enough to allow strong "cross-pseudonym" correlations. Note also that such pseudonym updates should be performed in large numbers at the same time, and preferably by a large fraction of the user population each time. We refer to [1] for a discussion of how to establish and manage personae.

4. INTELLISHOPPER PROTOTYPE

A partial prototype of the IntelliShopper has been implemented to test the ideas discussed above.[3] The current system resides on a single server and does not yet include the privacy agent, which will be deployed on a different server. Therefore in the current prototype each user has a single shopping persona. Although privacy protection is not currently supported in the current prototype, we note that this is built according to the architecture required to later add the privacy protection mechanisms. Email notification is also not yet implemented in our prototype.

In the development and deployment of the prototype we have used free open source tools exclusively. The prototype is implemented in Perl, using LWP and DBI modules for its Web and database interfaces, respectively. The database is implemented using MySQL. The IntelliShopper is deployed on a Darwin-based PowerMac with an Apache HTTP server.

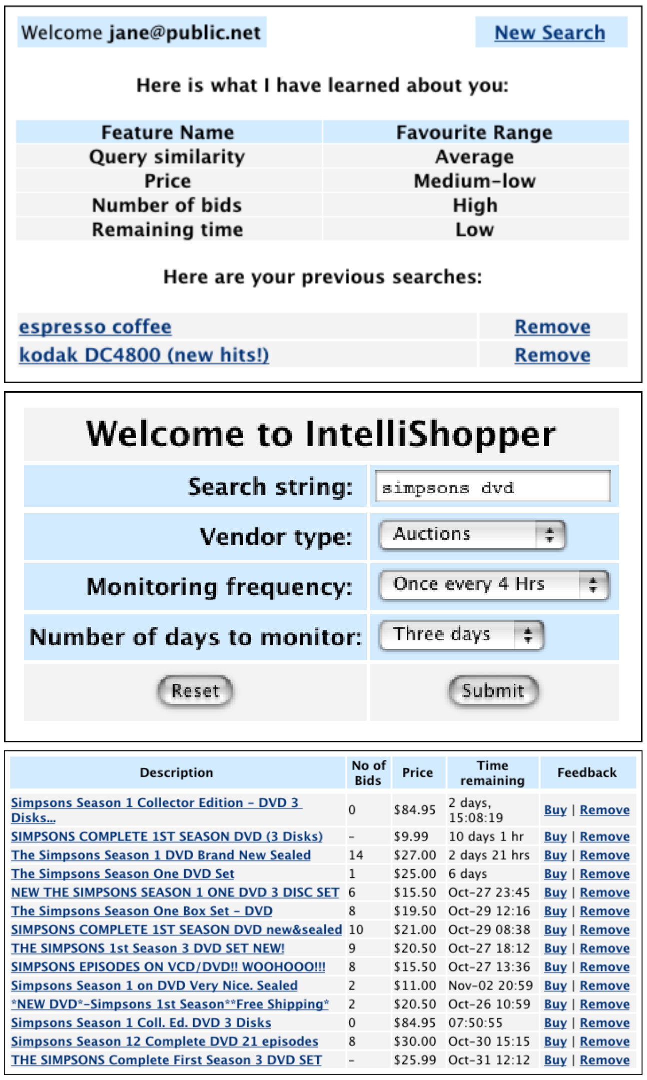
4.1 User Interface

Figure 3 shows some screenshots of the IntelliShopper user interface, illustrating how a user interacts with the system. When a user logs in, IntelliShopper displays the profile inferred by the learning agent based on the previous shopping activity of the current persona (user). The history of the shopper is also displayed, with live links to outstanding requests and a flag for queries for which the monitor agent has found new hits. The user can click to examine new or old hits, or remove requests he no longer wants to monitor.

Alternatively, the user can submit a new shopping request via the query interface. Here the user can specify a query string (to be forwarded to vendors) and the type of request, i.e., whether the user is interested in shopping at online stores or auctions sites. Each of these options corresponds to a set of vendor modules. In the future users will also be able to update their profile, including preferences among vendors. Furthermore, the user can specify for how long, and how frequently, the monitor agent should look out for new available items matching the query.

Once the results have been received from the various vendors, collated, parsed and stored in the database, the learning agent presents them to the user, ranked according to the persona (user) profile. As Figure 3 demonstrates, vendors have different formats for the displayed features. For example one auction sites might report the absence of bids as "0" and another as "–". Many different formats are also used to display the time remaining in an auction, even by a single vendor. An auction site might display "at 6:30PM" at one time and "in 10 minutes" at a later time, for the same item. All these formats are converted to common data domains before the value of each feature is stored in the database.

[3]The prototype can be accessed on the Web at `myspiders.biz.uiowa.edu/~nvish/IntelliShopper`



Welcome **jane@public.net** | New Search

Here is what I have learned about you:

Feature Name	Favourite Range
Query similarity	Average
Price	Medium-low
Number of bids	High
Remaining time	Low

Here are your previous searches:

espresso coffee	Remove
kodak DC4800 (new hits!)	Remove

Welcome to IntelliShopper

Search string:	simpsons dvd
Vendor type:	Auctions
Monitoring frequency:	Once every 4 Hrs
Number of days to monitor:	Three days
Reset	Submit

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: Top: The information displayed by IntelliShopper upon login. Middle: Query interface. Bottom: Results of a search.

4.2 Vendor Modules

The vendor modules allow IntelliShopper to interface with the various online store/auction sites. There are two aspects to a vendor logic from the IntelliShopper's perspective: (i) submitting queries, and (ii) parsing results. Task (i) is simpler; it consists of identifying an appropriate form, submission protocol, and input syntax on each vendor site. Task (ii) is more difficult; it consists of identifying items and extracting feature values for all desired features (e.g., product description, price, etc.). While vendors could readily simplify this task, say by using XML-based output, the opposite trend seems to be taking place; many vendors are not interested in competing on price alone, and therefore use complex and changing HTML markup to make it difficult for shopping bots to extract information from their sites.

There is much active research in the development of *intelligent wrappers* that could automate the above tasks. In fact, there is a sort of arms race between the intelligent wrappers employed by shopping bots and the growing complexity of HTML interfaces. Early shopping agents such as the ShopBot [4] demonstrated the interesting learning challenges stemming from this competition. However the IntelliShopper described here does not focus on this goal, therefore we followed a different route in our implementation. Rather than trying to build automatic wrappers, we simplified the task of hand-coding wrappers by designing a language for the specification of vendor-dependent logic. This way new vendor modules can be written in minutes.

```
<search
 name="eBay"
 action="http://search.ebay.com/search/search.dll"
 method="GET"
>
<input name="query" user />
<input name="maxRecordsReturned" value="50" />
<interpret
 item='<table width="100%"(.*?)</table>'
 key='<a href="http://cgi.ebay.com/.*?&item=(\d+?)">'
 url='<a href="(http://cgi.ebay.com/.*?&item=\d+)">'
 dsc='<a href="http://cgi.ebay.*?&item=\d+">(.*?)</a>'
 price='<font size=3><b>(.*?)</b></font>'
 bids='width="6%"><font size=3>(.*?)</font>'
 time='width="16%"><font size=3>(.*?)</font>'
/>
</search>
```

Figure 4: Simplified example of a vendor module. The module has logic to submit queries to a vendor site and to interpret the results.

A full description of IntelliShopper's vendor description language is outside the scope of this paper, however Figure 4 illustrates the idea with an example. The language is based on XML and is inspired by the plug-ins employed in Apple's Sherlock meta-search engine. A module contains two parts: querying and parsing. For querying, there are tags with fields specifying the URL of the form, the submission protocol (GET/POST), and the various necessary input parameters. For parsing, there is a tag with fields specifying field names and Perl regular expressions that extract the corresponding feature values. This flexible representation permits easy updates for vendor sites that are not particularly hostile to shopping agents.

All that is needed to allow IntelliShopper to integrate a new vendor is to drop its vendor module into the appropriate directory. The current prototype has modules for eBay, Yahoo, and Amazon auctions. It should be noted that while the use of regular expressions makes it easy to modify wrappers as necessary, it also yields wrappers that are not very robust in the face of changing vendor site design.

4.3 Database Design

IntelliShopper must store much data about its shopper personae, their profiles, queries, product hits, and their features. The prototype stores all this information in a relational database. Figure 5 is an entity-relationship diagram outlining the IntelliShopper database design.

The diagram is quite self-explanatory. The `preferences` table stores the profile of each persona; for each feature (e.g., price) the learning agent assigns a temperature to each of the value ranges taken by the feature, based on user feedback (cf. Section 3.1). For each hit in the `item` table, the `feature` table stores the value of each feature and the corresponding range. The learning agent uses this range to look

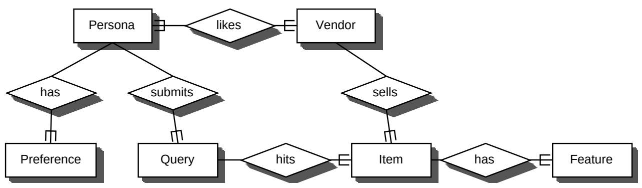


Figure 5: Data model of the database supporting the IntelliShopper. Single-line connections to entities represent relationships of cardinality 1, and trident connections represent cardinality N.

up the corresponding temperature in the `preference` table, for the persona who submitted the query that yielded this hit. The resulting temperatures for all feature values of an item are then combined to compute the item temperature and ultimately rank the hits.

5. EVALUATION

Evaluation of an agent like IntelliShopper is difficult because measures of success and/or performance are subjective. Ideally one should compare user satisfaction between shoppers using IntelliShopper and shoppers using other shopping agents. This is problematic for a number of obvious reasons, therefore we have focused on the performance of the learning agent.

The goal is an evaluation measure that is both quantitative and objective, while based on data from real users. We thus recruited 51 distinct volunteer subjects who used IntelliShopper during actual shopping sessions. The subjects were asked to sit through two or more sessions over the weekend of 27-28 October 2001, shopping for any number of items of their choice. The subjects submitted a total of 97 distinct queries (1.9 queries per user on average) and sat through a total of 127 shopping sessions (2.5 sessions per user on average). The entire experiment involved a total of 3,425 distinct hits, or 27 hits per session on average (at most 10 hits per query were retrieved from each of the three auction sites).

During each shopping session a subject could submit requests, look at new hits for previous requests, and provide indirect feedback to the learning agent through the IntelliShopper user interface. All the user requests, hits and feedback were recorded along with two rankings of all the hits in each session. The first ranking was the one used by the system to display hits, based on the learned user profiles. The second ranking was computed based on the feedback inferred from user actions during each session. For the hit set corresponding to each `(user, session, query)` tuple, we measured the Spearman's rank correlation coefficient between these two rankings:

$$\rho = 1 - 6\frac{\sum_{i=1}^{n}(rank_{IntelliShopper}(i) - rank_{user}(i))^2}{n(n^2-1)}$$

where n is the number of hits ranked in each session.

The idea is that if the learning agent is effective, the correlation between the ranks learned by the system and the ranks inferred from user feedback should increase over sessions. Figure 6 plots the mean Spearman's rank correlation coefficient against the number of shopping sessions. After the first three shopping session we observe a significant

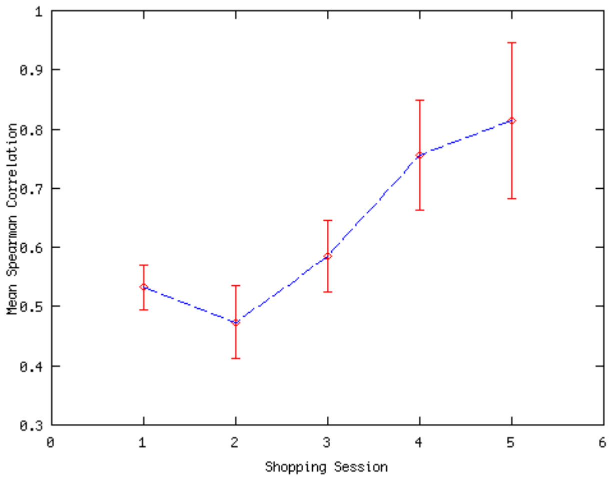


Figure 6: Spearman's correlation between IntelliShopper's ranking and ranks inferred from user feedback. A correlation coefficient is computed for every user/session/query, then these are averaged across users and queries in each session number. Error bars correspond to ±1 standard error.

improvement in performance, indicating that the learning agent is effectively predicting user preferences.

6. CONCLUSION

This paper introduced IntelliShopper, a shopping assistant designed to empower consumers by adapting to their personal preferences, searching proactively on their behalf, and protecting their privacy. IntelliShopper can learn a user profile without requiring explicit feedback from users, but rather observing their actions in an unobtrusive fashion. The feasibility of the approach has been demonstrated through an implemented, publicly available prototype. This prototype has also allowed us to evaluate the performance of IntelliShopper's learning agent, showing that the system can quickly build user profiles that can rank items according to user preferences.

Several extensions and improvements of the current prototype are under way. First, we will implement the privacy agent to demonstrate the feasibility of the proposed anonymity and pseudonymity protocols for guaranteeing the privacy of the users. The privacy agent will also enable a more straightforward implementation of multiple shopping personae for each user, allowing the learning agent to better adapt to the heterogeneous shopping needs of real users. Other immediate additions to the prototype will include new vendor modules for online stores and an option to allow users to specify preferences among vendors.

Future versions of IntelliShopper will include several updates to the learning agent, improving the system's ability to adapt to user preferences. More simple features (such as brand name) will be added,subject to our ability to consistently extract them from the item page. We will also add a more sophisticated similarity mechanism that compares hits to other items that have been judged by the user. The temperature combination scheme will be made adaptive, so that, for instance, if a particular user often makes decisions based on price, the price feature will be more highly weighted in ranking hits. The α_1, α_2 parameters in Equation 1 can also be tuned adaptively to efficiently track dynamic profiles.

We will explore the use of temperature settings from previously-learned profiles to bootstrap the profiles of new personae. It is reasonable to assume that some information from a user's existing personae would be useful when ranking hits for a new persona's queries; for example, a user might often prefer low-priced items. However, this might compromise the identity of the user. In order to avoid the complete loss of learned profiles during the creation of a new persona, a few features of the profile may be transferred to a new persona by the user. There are a few ways to improve privacy at this stage: (i) perform the task simultaneously with many users; (ii) stagger the introduction of a new persona with the cessation of a previous one; (iii) shed personae frequently; or (iv) use very concise profile templates when starting a new persona. Solution (i) is logistically difficult; all other solutions, on the other hand, hinder efficient learning. Therefore the privacy and learning aspects of the shopping agent need to be balanced against each other. This may be done by the user — on a per persona basis — by selecting an appropriate trade-off between privacy and personalization.

The user interface can be improved to allow for better learning based on user actions and more efficient database transactions. For example, we plan to allow users to choose multiple items to be removed simultaneously from the presented list of hits. Such an action will reduce the number of database transactions and will render feedback to the learning agent more consistent. Furthermore, the interface should communicate better the learning that is taking shape to the user, by emphasizing and de-emphasizing items based on the learned knowledge.

A direction for future research is to study the opportunities for collaborative filtering that stem from centralized shopping assistants such as IntelliShopper. The effectiveness of collaborative filtering is well established. It would not be difficult, at the users' option, to cluster personae based on their profiles, and then extend a request's hit set with items previously located on behalf of other personae with similar profiles. This could be done irrespective of queries, or subject to some minimal similarity between queries.

We mentioned the need for more robust and/or adaptive wrappers to interface with vendor sites. Alternatively, vendors may find it economically advantageous to become "friendly" to shopping agents such as IntelliShopper, that put the customer at the center of the business relationship, not only based on price competition but on a number of other factors. We view this as a better business model than either price-only bots or, worse, comparison shopping agents that are biased by hidden fees from vendors. The first generation on user-centered shopping bots did not survive the transition to the commercial realm; bots such as PersonaLogic, Firefly and Jango were quickly replaced by the current vendor-biased agents. The ultimate fate of the new generation of shopping assistants proposed here will be likewise decided by the marketplace.

7. ACKNOWLEDGMENTS

Thanks to DIMACS for support, to the Adaptive Agents and Data Mining research groups at U. Iowa for their suggestions, and to the volunteers who helped with the evaluation. The server hosting the IntelliShopper prototype was made available by a U. Iowa instructional improvement award.

8. REFERENCES

[1] R. Arlein, B. Jai, M. Jakobsson, F. Monrose, and M. Reiter. Privacy-preserving global customization. In *ACM E-Commerce '00*, 2000.

[2] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Comm. of the ACM*, 24(2):84–88, 1981.

[3] U. Doo and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proc. ACM Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[4] R. Doorenbos, O. Etzioni, and D. Weld. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the First International Conference on Autonomous Agents*, pages 39–48, 1997.

[5] E. Gabber, P. Gibbons, Y. Matias, and A. Mayer. How to make personalized Web browsing simple, secure, and anonymous. In R. Hirschfeld, editor, *Financial Cryptography '97*, pages 17–31, 1997.

[6] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval. In *STOC '98*, 1998.

[7] A. R. Greenwald and J. O. Kephart. Shopbots and pricebots. In *Proc. 16th Intl. Joint Conference on Artificial Intelligence*, pages 506–511, 1999.

[8] M. Jakobsson. Flash mixing. In *PODC '99*, pages 83–89. ACM, 1999.

[9] M. Jakobsson and M. Yung. On assurance structures for WWW commerce. In *Financial Cryptography '98*, 1998.

[10] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A tour guide for the world wide web. In *Proc. Intl. Joint Conf. on Artificial Intelligence*, 1997.

[11] J. O. Kephart and A. R. Greenwald. Shopbot economics. *Autonomous Agents and Multi-Agent Systems*. forthcoming.

[12] H. Lieberman. Autonomous interface agents. In *Proc. ACM Conference on Computers and Human Interface*, Atlanta, GA, 1997.

[13] L. Liu, C. Pu, , and W. Tang. WebCQ: Detecting and delivering information changes on the Web. In *Proc. Int. Conf. on Information and Knowledge Management (CIKM)*, 2000.

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

[15] P. Maes, R. Guttman, and A. Moukas. Agents that buy and sell. *Comm. of the ACM*, 42(3):81–91, 1999.

[16] M. Mitomo and K. Kurosawa. Attack for flash mix. In T. Okamoto, editor, *ASIACRYPT '00*, pages 192–204, 2000. LNCS No. 1976.

[17] B. Nguyen, S. Abiteboul, G. Cobena, and M. Preda. Monitoring XML data on the Web. *ACM SIGMOD Record*, 30(2):437–448, 2001.

[18] M. K. Reiter and A. D. Rubin. Anonymity loves company: Anonymous Web transactions with Crowds. *Comm. of the ACM*, 42(2):32–38, 1999.

[19] P. Syverson, D. Goldschlag, and M. Reed. Anonymous connections and onion routing. In *Proc. of 18th Annual Symposium on Security and Privacy*, pages 44 – 54. IEEE Press, 1997.