A Grammatical Approach to User Model Resolution for Electronic Shopping

A.G. Manousopoulou & G. Papakonstantinou

Computer Systems Laboratory, National Technical University of Athens
Iroon Polytechneioy 9, 15773 Athens, Greece
{natasha,papakon}@cslab.ece.ntua.gr

Abstract

Electronic shopping is a rapidly expanding market and can reap benefits from the employment of user modeling techniques in order to personalize the shopping procedure for each customer. In this paper we present a methodology for user modeling that uses attribute grammars to extract a user representation from the navigation steps and actions the user takes inside the electronic shop. This method can be applied incrementally, demanding little space for user information, and avoiding the frequent recalculation of the entire user model.

1. Electronic shopping and user modeling

Shopping on the Internet constitutes a rapidly growing part of the worldwide market. E-commerce revenues, which are reported to be around \$100 billion for 1999, are expected to grow as high as \$1300 billion in 2003 [3]. Research conducted in the US points out a 30% increase in Internet shopping from 1998 to 1999 and UK surveys demonstrate even higher increases, with an expected doubling of online sales between 1998 and 1999. European buyers in general are expected to be spending 17 billion Euro for online purchases by 2004 [4].

Benefits of electronic shopping are twofold, applied both to the seller and the consumer. The most important of these is the potential for elimination of intermediaries between producer and consumer, which leads to the lowering of transaction costs and improvement of competition.

A wider, possibly global market, is open to the seller, who can display the available products at a very low cost to a big audience. The lists and descriptions of available goods can be maintained with very little cost, especially compared to the costly maintenance procedure of printed catalogs.

The customer, on the other hand, can save a significant amount of time and money with online shopping, not to mention the greatly improved convenience of the shopping experience, often a determining factor in consumer preferences. The information available to customers is ample and always up to date; it can be filtered by the potential buyer who wants to restrict the viewed data, in order to improve the selection and ordering process.

Filtering of displayed information can also be performed by the sellers themselves, so that they can present the data in a manner and order more appealing to their audience. In order to select which data to display and how, the electronic shop software must be provided with clues about the user's preferences and buying habits. These clues can be incorporated under the general hood of a user model, which can be generally used to:

- Collect information about the customer base, and create the collective profile of the target audience of the store and its merchandise, in order to make decisions on a future move, expansion or specialization.
- Provide each user with "appropriate" (that is, most possible to succeed) targeted advertisement.

- Adapt the presented product information according to the user's personality (computer literacy, general education level, age group).
- Provide additional benefits to selected customer groups in order to increase revenues.

For example, a user model can answer questions such as:

- How often does the user conclude a visit with a purchase?
- What is the user's manner of browsing the catalog and searching for specific merchandise? How can this manner be better provided for?
- Does the user follow "Special offer today" links?

User modeling is a field of Artificial Intelligence generally used in person-machine interfaces, natural language dialog systems, and interactive applications such as help or tutoring systems [6]. Its purpose is to record information relevant to the user and adapt the system behavior according to this information, which comprises of descriptions of the user's goals, plans and beliefs. It takes software and data customization one step forward, as the user does not take steps to perform the customization, but rather allows the software to indirectly record preferences and adapt to them unobtrusively (see for example [8]).

User modeling has also been used on the Web in order to determine user goals and assist the searching and browsing procedure [1], [5], [7].

2. A grammatical model of user behavior

A user navigating an e-shop visits Web pages, moving forward and backward in the catalog, adding and removing items from shopping carts, specifying shipping and handling preferences, and, possibly, concluding a purchase. This linear (in time) sequence of events, spanning one or more visits, can be represented as a string, the user navigation string, which can be parsed to extract the user's pertinent model.

2.1. Modeling user behavior with attribute grammars

A very popular way of representing user models is that of stereotypes [10], which are descriptions of prototypical users representing user classes. The typical manner of encoding stereotypes is that of frames, which correspond roughly to attribute-value sets. In a frame, each slot (i.e. attribute) contains a value that best summarizes the corresponding feature of the specific class population.

If we represent the user model as a set with feature - value pairs, the use of attribute grammars can make the transition from parsing to modeling very smooth.

An attribute grammar representation of the user modeling procedure will associate Web page "types" (catalog, item, search, offer, options, checkout) with terminal symbols. These symbols will form the user navigation string.

Terminal symbols are associated with attributes that give additional information about the Web page visited: the specific catalog page, price of an item browsed, keywords entered into a search form, time the user spent on each page, total amount spent on a purchase, means of payment.

User features are synthesized attributes associated with the grammar start symbol. The goal of the user model evaluator is the computation of these attributes, while the structure of the parse tree and attributes of intermediate nodes are of significantly less importance.

The context free backbone of the grammar reflects the way that Web page types contribute to the user model, while semantic rules percolate and transform characteristics of this page to yield the user's features.

To illustrate the model extraction method, we will use a toy scale example. An electronic bakery, contains a single page catalog, which displays three different types of bread and four types of cookies, each one of which is also displayed on a separate page. There is no search feature, and the bakery provides home delivery. The user will browse the catalog and item pages, adding items to the shopping cart from their respective pages, and possibly complete the visit with a purchase.

The terminal symbols of the grammar will be:

- C, the catalog page, with a link for each item page. No attributes are associated with this symbol.
- I, an item page, which carries three attributes
  - *price*, the advertised price per weight for the item.
  - *type*, the type of the displayed item.
  - *weight*, the desired weight of an item to be purchased, as provided by a user at the page.
- K, for the checkout page, where the delivery address is provided. This symbol is associated with attribute *area*, the location code for the delivery address and *completed*, which shows whether the user actually finished the checkout procedures.
- $, which marks the end of a visit. This symbol is not associated with attributes.

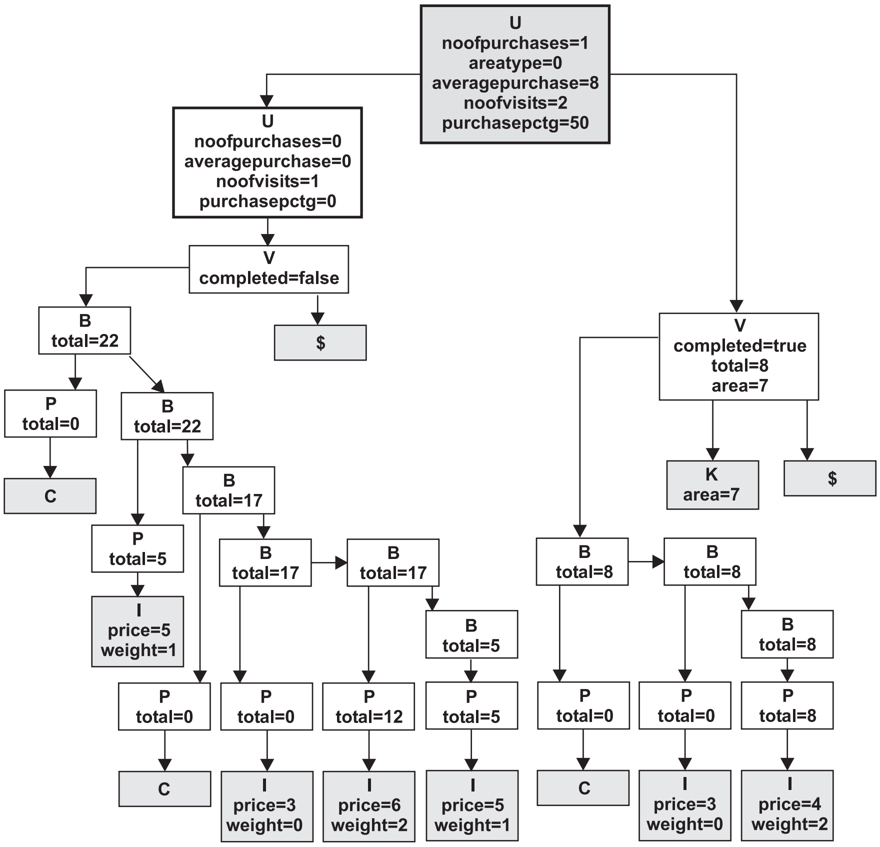

Figure 1: The parse tree

Nonterminal symbols of the grammar are:

- U, the start symbol of the grammar, which represents the user. This symbol carries the attributes that constitute the user model, namely:
  - *areatype*, the buyer's area type; areas are classified by average income
  - *averagepurchase*, the average cost expended per purchase
  - *noofpurchases*, the number of purchases made
  - *noofvisits*, the number of visits made to the e-store

  - *purchasepctg*, the percentage of visits that finish with a purchase.

- V, representing a visit to the e-store. The attributes associated with V are:
  - *completed*, which is boolean and indicates whether a visit ended with a purchase.
  - *area*, identical to the homonymous attribute of symbol K.
  - *total*, the value of goods purchased during the visit.
- B, which stands for catalog/item browsing and adding items to the shopping cart, having the attribute *total*, for the total value of items added to the cart.
- P, for a catalog or item page. This symbol also carries the attribute *total*, which is the value for an order of one item.

Figure 1 shows the application of this grammar to the user navigation string CICIIII\$CIIIK\$, with attributes appearing on the tree nodes.

The entire attribute grammar is:

U → UV	U_0.noofvisits = U_1.noofvisits + 1;
	if V.completed then begin
	   U_0.noofpurchases = U_1.noofpurchases + 1;
	   U_0.areatype = gettype(V.area);
	   U_0.averagepurchase =
	      (U_1.averagepurchase × U_1.noofpurchases + V.total) /
	      U_0.noofpurchases;
	end
	else begin
	   U_0.noofpurchases = U_1.noofpurchases;
	   U_0.areatype = U_1.areatype;
	   U_0.averagepurchase =
	      (U_1.averagepurchase × U_1.noofpurchases) /
	      U_0.noofpurchases
	end;
	U_0.purchasepctg = U_0.noofpurchases × 100 / U_0.noofvisits;

U → V	U.noofvisits = 1;
	if V.completed then begin
	   U.noofpurchases = 1;
	   U.areatype = gettype(V.area);
	   U.averagepurchase = V.total;
	   U.purchasepctg = 100
	end
	else begin
	   U.noofpurchases = 0;
	   U.areatype = invalid;
	   U.averagepurchase = 0;
	   U.purchasepctg = 0
	end

V → B\$	V.completed = false; V.total = 0; V.area = invalid
V → BK\$	V.completed := K.completed; V.total = B.total; V.area = K.area
B → PB	B_0.total = P.total + B1.total
B → P	B.total = P.total
P → C	P.total = 0
P → I	P.total = I.price × I.weight

This grammar contains a rule with left recursion, which is considered to be poor practice. Any parser that uses bottom up techniques, such as CYK or Earley's can easily cope with such a rule, which was put there to show the prospects for incremental model extraction. The rule could be expressed much more elegantly and without the recursion problem in BNF or any other form that uses regular expressions.

3. Incremental model extraction

The user model extraction method described above uses the navigation string for one or more consecutive visits to the store, which changes with time, as the user comes back. Therefore, the model of the user must be updated, if not at each visit, at least quite often, in order to reflect the user better.

Updating the user model is not a trivial problem, given that a store can have thousands of users that make big numbers of visits. Rebuilding the entire model with each visit can become soon very costly, and overcome whatever benefits the seller may gather from it. It is essential thus that the updating procedure is incremental, and recalculates the model using the new navigation information only. This also eliminates the need for storage of previous user visits and behavior, which is now encoded inside the model.

In a grammatical approach, incremental model extraction can be perceived as a continuous parsing procedure with the appropriate grammar. If the context free base of the attribute grammar contains recursive rules for the start symbol (the user model symbol), like the example grammar given in this paper, and the semantic rules calculate the root attributes from the attributes of the "visit" symbol and those of the previous model symbol, the parsing can be performed incrementally without cost as in Figure 2.

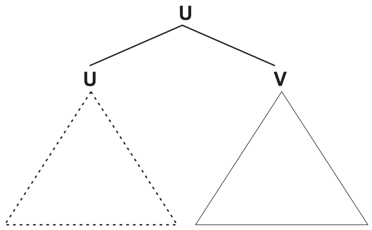

Figure 2: Incremental extraction

In our example, after each visit to the store, the user modeling system can throw away all the intermediate information and retain only the user model, i.e. the root of the parse tree. This root is, in a sense, the encapsulation of the entire navigation of a user in a single node. The system can use this single node as if is has also recorded the entire parse tree that the root dominates, and create a new user model.

In Figure 1, the navigation string consists of two visits to the store. After the first visit, the attribute grammar evaluator has built a parse tree with the root

$$U : \begin{bmatrix} noofpurchases = 0 \\ averagepurchase = 0 \\ noofvisits = 1 \\ purchasepctg = 0 \end{bmatrix}$$

With the addition of the symbols for the second visit, the parsing can continue from the point where the subtree of the leftmost symbol U has been created and the attributes evaluated. The subtree with root V is created for the new terminals, and the parsing is completed with the creation of a new symbol U. The attributes are then evaluated bottom up, using the old user model as a leaf. The result can be seen in Figure 3.

This incremental procedure of model creation offers another benefit as well: whenever the parser can create a full parse tree from the user's navigation string, the nodes of the new tree can be thrown away, together with the information they hold, and a new user model can be created. In this manner, the user model can be utilized during the visit as well, tailoring the presentation of the shop to a more up-to-date model. This is especially useful during the user's first visits to the store, when the model is rather crude, and the smallest piece of information makes a perceivable difference.

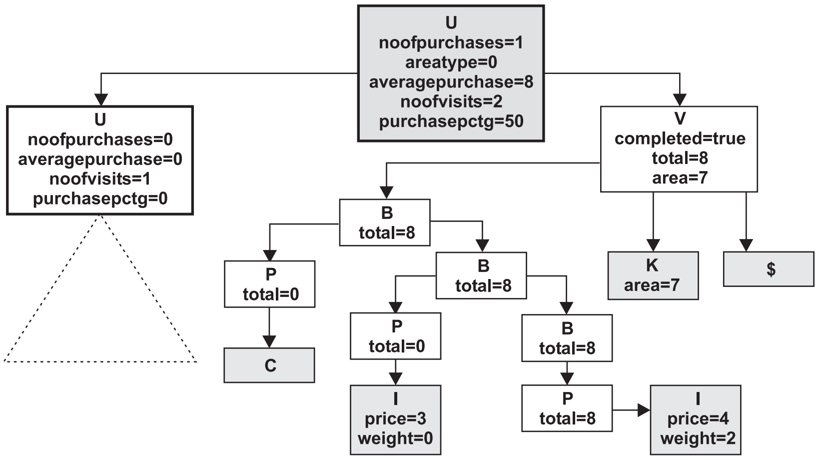

Figure 3: Incremental extraction — example

4. Creating the user models

The stage of matching a model with a user comes second in a user modeling system. The first stage is deciding how to represent the user models and what the user models will be.

The creation of user models can of course be performed manually, by an expert in the specific field of electronic commerce. The attribute grammar will be constructed according to these models, transforming the user information so that it matches the most appropriate stereotype. This approach demands that the user models change very rarely, if at all, and is rather inflexible.

A more appealing way will be that of automatic user model creation, from a sample database of user navigations.

In an attribute grammar environment, we create a user "model" for each user navigation, that is, we calculate attributes for the particular user. A rudimentary attribute grammar is necessary at this stage. Afterwards, from this new database of user descriptions, a classification method produces user classes and stereotypes, and the attribute grammar is enriched with new semantic rules that adapt the attributes of a user to those of the appropriate class. A frame, or attribute-value set, can also be used as a vector, or a point in a feature space. There is in the bibliography a multitude of classification methods that extract classes from a set of points, such as the nearest neighbor or other distance based methods. An example of user model creation through vector classification can be found in [2].

An alternative approach, which does not use attribute grammars during the model creation phase, employs knowledge representation techniques to describe each user navigation, and machine learning to make inferences about these descriptions and create user prototypes (see e.g. [9]).

As soon as the user descriptions are classified and the stereotypes are extracted, the adaptation of the attribute grammar takes place. The adaptation reflects the distribution of users among classes and yields the final attribute grammar that will be used online to determine the user model.

5. Implementation

In a commercial environment, speed of service is very important to keep the customer happy and coming back to the store. This does not change for electronic stores; a slow connection or slow dynamic Web page creation will discourage the shopper.

In the model we propose, speed is particularly important if the user model is built incrementally during the user's visit to the store. Each visit to a page will consist of a recalculation of the user model (a partial one if the grammar is appropriate) and data filtering according to the new model. It is obvious that the recalculation must be as fast as possible, so that the user will be presented with the filtered data quickly; parallel calculation of the model [11] can be an ideal solution in this case.

6. Conclusions

In this paper we have proposed a methodology to extract user models for electronic shop customers with attribute grammars. This methodology parses the sequence of actions the user makes in the e-store's Web site to extract the characteristics of the user. We show that this methodology can be incremental, in order to update user models with time and keep them up to date.

Our plans for future work in this vein is the application of this grammatical methodology in a electronic commerce application and its evaluation, in terms of speed and scalability. We also plan to combine it with an appropriate strategy for initial user model creation, so that we can study the effect that the initial creation has on the extraction grammar development.

Bibliography

[1] L. Ambrosini, V. Cirillo and A. Micarelli (1997). Hybrid Architecture for User-Adapted Information Filtering on the World Wide Web. *Proc. of the 6th Intl. Conf. on User Modeling (UM 97)*, pp. 59–61.

[2] R. Bushey, J. M. Mauney and T. Deelman (1999). The Development of Behavior-Based User Models for a Computer System. *Proc. of the 7th Intl. Conf. on User Modeling (UM 99)*, pp. 109–118 .

[3] eMarketer eBusiness Report^{TM} (1999). *http://www.emarketer.com/estats/sell_ ebiz.html*.

[4] Fletcher Research Press Release (1999). *http://www.fletch.co.uk/content/thinks/pressEurope.html*.

[5] M. Gori, M. Maggini and E. Martinelli (1997). Web-Browser Access Through Voice Input and Page Interest Prediction. *Proc. of the 6th Intl. Conf. on User Modeling (UM 97)*, pp. 17–19.

[6] Alfred Kobsa (1990). User Modeling in Dialog Systems: Potentials and Hazards. *AI & Society* 4:214–240.

[7] Tessa Lau and Eric Horvitz (1999). Patterns of Search: Analyzing and Modeling Web Query Dynamics. *Proc. of the 7th Intl. Conf. on User Modeling (UM 99)*, pp.119–128.

[8] G. Linden, S. Hanks and N. Lesh (1997). Interactive Assessment of User Preference Models: The Automated Travel Assistant. *Proc. of the 6th Intl. Conf. on User Modeling (UM 97)*, pp. 67–78.

[9] Pohl, W. (1997). LaboUr Ũ machine learning for user modeling. In Smith, M. J., Salvendy, G., and Koubek, R. J., eds., *Design of Computing Systems: Social and Ergonomic Considerations (Proceedings of the Seventh International Conference on Human-Computer Interaction)*, B:27-Ũ30.

[10] E. Rich (1983). Users are individuals: individualizing user models. *International Journal of Man-Machine Studies* 18:199–214.

[11] A. M Thanos and G. Papakonstantinou (1999). Facilitating the development of parallel implementations of declarative programming languages using attribute grammars. *Proc. of the 2nd Workshop on Attribute Grammars and their Applications*, pp. 17–36.