A PDA-Based Personalized Recommender Agent

George S. Almasi and Albert J. Lee
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598
almasi@us.ibm.com, AJL44@columbia.edu

Abstract

As an example of distributed personalization in a pervasive computing environment, this paper describes a PDA-based personal agent "wine guru" that works hand-in-hand with an on-board supermarket shopping program, and also with a server-based datamining program that provides personalized wine lists. The guru ascertains which list of the store's wines from the server best matches the user's tastes in wine, keeps an eye on the user's on-board wine cellar list, and can read the shopping list that the user is preparing and then suggest wines to go with some of the items after asking how they will be prepared. The user may elect to add some of the suggested wines to the shopping list, or choose from the wine cellar. For portability, a subset of Java with a VM small enough to fit onto a PalmPilot was used. It provided all the needed functionality, and acceptable response time.

Keywords: recommender systems, personalization, collaborative filtering, content filtering, data mining, intelligent agents, intelligent assistants, Java, Waba, clustering, pervasive computing.

1 Introduction

We describe a PDA[1]-based recommender agent designed to take advantage of the limited but increasing computational resources of hand-held devices operating as part of a remote server-based system such as Smartpad [1], which was recently developed by IBM and Safeway Stores plc for retail shopping. That system does generate and deliver personalized recommendations [2] to the participants' PDAs, but these recommendations are pre-personalized on the server, and the PDA merely acts as a delivery mechanism, as well as a means of assembling and returning a shopping list – the PDA performs no computation per se. The PDA has several hundred times less computational power than the server for tasks like data mining, but the PDA potentially "knows" a lot about its user that the server doesn't. Would

[1] Personal Digital Assistant

a better recommender system result if some of its parts were carried out on the PDA, by something like an intelligent agent or assistant [3] [4], and does the PDA have the requisite computational capabilities? That was the focus of the project reported in this paper.

Interviews with customers of the remote shopping system indicated a desire for a more interactive recommender, one that could help with menu planning. These interviews also identified table wines as a product group in which recommendations are welcome, a point supported by the items selected from the recommender. For these reasons, our first prototype was a system for matching meals to wines likely to appeal to the customer at hand. Distinct wine lists designed to appeal to different tastes are generated on the server and delivered to the PDA. The server chooses a default active wine list on the basis of the customer's spending history, but the customer can change the active wine list manually or by interacting with a little on-board "wine taste deducer" program. The wine list can also be augmented with wines from the customer's personal catalog and/or private "wine cellar" list. In addition to these on-board lists, the system can read the shopping list that the customer is preparing, and offer suggested wines to go with some of the items that the customer has just put on the list.

The functionality described above was successfully encoded for the PalmPilot using a subset of Java and yielded acceptable response times, with higher performance available by recoding in C++ if needed.

2 Design

The core function performed by the "wine guru" is to recommend an appealing wine to accompany a particular meal. It can be used whenever the user needs such a recommendation, in a restaurant, for example, or during the menu planning phase of preparing a supermarket shopping list. Figure 1 shows this being done in the context of Smartpad [1], a remote supermarket shopping system based on server-supported PDAs. The high-level design is described in this section, while more detail is presented in Section 3.

The key item in the system shown in Figure 1 is a personalized list of wines that match different meal types. The personalized wine list starts out as a set of lists of the store's wines that the server prepares using data mining, collaborative filtering, or some other technique, and may be augmented by wines previously bought by this user or by a "wine cellar" list that the user maintains on the PDA. The starting wine list can be pre-selected by the server based on the user's purchase history, or can be chosen based on user responses to a small on-board "wine taste deducer" program.

The user selects a meal type, the wine-food matcher program selects candidates from the personalized wine list, and the user then has the choice of adding the wine to the shopping list or selecting from the wine cellar.

Figure 1 shows two sets of information on the PDA that the server doesn't have access to, at least not yet: "my wine cellar", a personal inventory of wines acquired from various sources, and the shopping list currently under preparation. There are many possible scenarios for text mining such information to provide useful services to the customer. Section 3 describes a simple example that we implemented, namely, a shopping list reader that selects

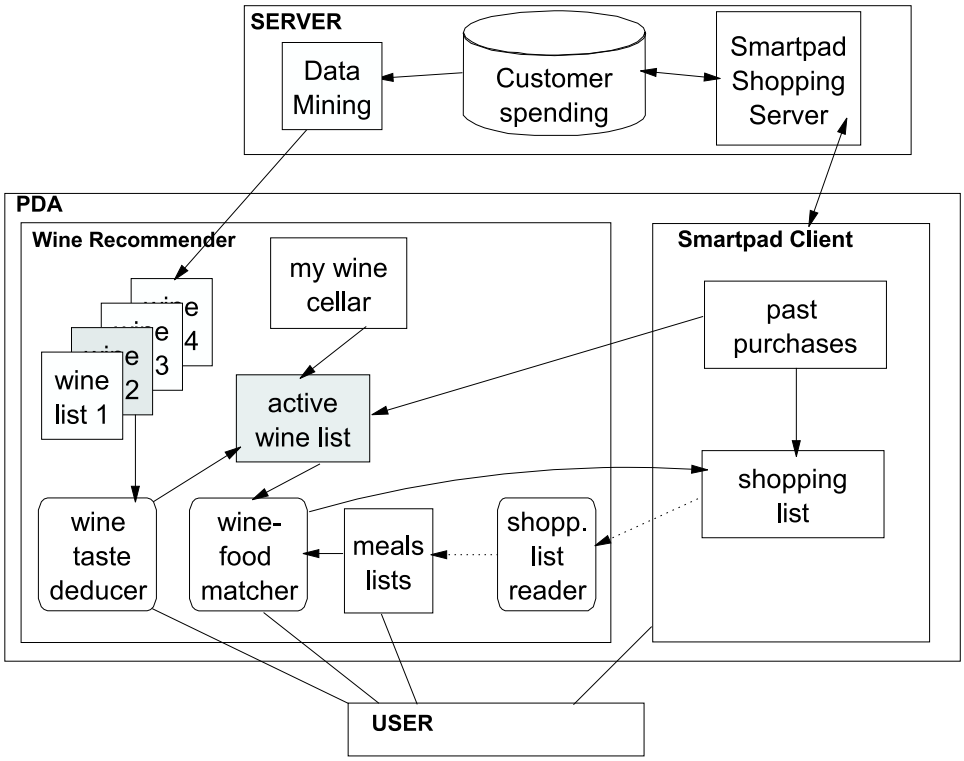

Figure 1: Wine recommender system diagram and usage scenarios. The server and the PDA share in performing the personalization. A wine-food matcher on the PDA matches meals of interest to wine lists supplied by the server, as well as performing other functions (see text).

candidates for a matching wine recommendation and, on request, takes the user to the appropriate meal preparation page, after which the operation is the same as above.

The user can also access the personalized wine list directly, in case matching to a meal is not needed.

The next section describes our initial implementation.

3 Implementation

This section describes the implementation details of our prototype wine recommender's chief components (see Figure 1). For convenience and portability, the prototype was written in Waba [5], a subset of Java with a Virtual Machine small enough to fit onto a Palm Pilot. Waba does not support threads, exceptions, or 64-bit arithmetic, limitations that we deemed acceptable in exchange for the ease of generating code that would run under PalmOS, Windows CE, in any Java environment, or as an applet in a web browser (readers can try the prototype on one of the authors' web pages [6]). Applications in Waba run about 3-5X slower than if written in C++, and our fallback plan was to re-code in C++ if performance became an issue.

3.1 Personalized wine list generation

The user's personalized wine list contains four wines in each of 10 "weight" categories, for a total of 40 wines. The food-to-wine matcher (see below) uses the assigned wine weights (1-4 for whites, 5-8 for reds, and 9-10 for dessert wines) to recommend wines to go with a selected meal. At present, the four wines in each weight category are simply the four in that category that are most popular with the user's "peers", although other business strategies are also possible.

The server can find a group of "peers" for the customer in several different ways. For example, one of the collaborative filtering schemes [7] used in many web-based recommenders could be adapted to treat the customer's past spending as product ratings, and provide recommendations with a high degree of personalization. We chose a somewhat more modest approach to begin with, which was to cluster customers by their wine spending. The datamining technique that we used is described in section 4. As can be seen in Figure 2, even a 4-way clustering produces four groups of customers with quite different tastes in wine.

In this example, the server would then create a list of the four most popular wines in each of the 10 categories for each cluster and then send the 4 lists to the PDA, along with the PDA owner's cluster assignment if known. If not, the PDA owner uses the on-board wine taste deducer (next section) to arrive at a cluster assignment. The cluster assignment decides which of the 4 lists serves as the starting point for the user's personalized wine list, which can be augmented by wines from an on-board "wine cellar" list or with wines from the shopping history maintained by the supermarket shopping program, at the user's discretion.

3.2 Wine taste deducer

The wine taste deducer performs a quick survey of the user's tastes in wines upon request. The results are used to determine a customer's cluster assignment. In the example shown in Figure 3 b, users are asked to rate some or all of the 10 most widely bought wines. A modified version of the neural cluster scoring algorithm is applied to the ratings to determine cluster assignment.

An alternative approach to cluster assignment is to extract rules for cluster membership (the Intelligent Miner tree classifier can be used for this) and then have a small on-board expert system use these rules to translate the survey results into a cluster assignment. This latter approach offers greater versatility for implementing various business strategies.

3.3 Food-to-wine matcher

There are many ways to match foods and wines to each other. Our implementation uses a simple scheme similar to that used on the Wine Spectator's web page [11]. The basic idea is that meals and wines each have a "weight", and a light meal like a delicate fish in a light sauce is well accompanied by a light white wine like Orvieto. Each wine is assigned a weight between 1 and 10; whites range from 1 to 4 in order of increasing weight, reds similarly range from 5 to 8, light dessert wines are assigned 9 and heavy ones 10. Meal types are divided into 15 categories (fish, poultry, beef, cheese, dessert, etc), and each category is further divided according to ingredient and preparation method (delicate fish broiled, medium fish with hearty sauce, etc) and assigned a weight (or range of weights) between 1 and 10. When a user selects one of these meals, the food-to-wine matcher accesses the user's personalized wine list and selects and displays several wines whose weight matches that of the meal. The user may then either select one of the wines to add to the shopping list, or mark it for withdrawal from the wine cellar list.

More complex matching schemes based on multiple attributes are of course possible, but the simple "content filtering" (in the language of recommender systems) that we described above gives surprisingly reasonable results.

3.4 Shopping list reader

As we mentioned above, the shopping list reader is a simple example of a useful content-filtering operation performed on the PDA. As a convenience option, the user can ask the wine guru to read the shopping list under preparation and find items for which it knows how to recommend wines. This operation tests the string-parsing capabilities of the Java subset being used on the PDA. The filtering is done in 2 stages. Each item on the shopping list contains a byte indicating which of 16 product categories the item belongs to, and we check for two of these categories (meat/fish and dairy). If the byte matches, we parse the rest of the string for words like beef, chicken, etc, and present a list of the found items to the user. Selecting an item on this list takes the user to the appropriate meal types menu to select a preparation method, after which the wine recommendation is made as usual. This simple example only scratches the surface of possible useful text mining operations on the PDA.

4 Results

Figure 3 shows an example of the wine recommender in action. We assume that this user is new to the store and has selected the "ask my preferences" option shown in part (a) of the figure. The mini-survey presented to the user is shown in (b). Assume the user indicates liking Lambrusco and disliking Cabernet Sauvignon, and is consequently assigned to cluster 1, the "white wine lovers" cluster in Figure 2. Next assume the user is planning fish for dinner, and selects that category from the main menu in (c). Now assume the user is planning something like broiled filet of sole, and selects "Delicate broiled (sole, flounder)" in (d). Part (e) shows the list of recommended wines. Had the user expressed a strong liking for Bulgarian Cabernet Sauvignon in (b), an assignment to cluster 2 ("Bulgarian Cabernet Sauvignon fans") would have resulted, and a different list of white wines would have appeared (f).

There are interesting and valid questions about the quality of the food-to-wine matching, the wine taste deduction, the clustering, and even the choice of clustering vs. some other peer-finding mechanism like collaborative filtering, but these questions are beyond the scope of this paper. The point of this paper is that a PDA-based intelligent agent/assistant that can communicate with a server and can also make good use of its standalone computing capability can have the best of both worlds – for example, to perform useful functions like providing on-the-spot wine recommendations. Because the recommendations are personalized and also available anywhere, the service provided is more than either the PDA or server can do alone. Our purpose was to show that a program that performs these tasks reasonably well can already fit onto current PDAs and run fast enough to be practical.

4.1 Data Mining Results and Considerations

The data used for clustering customers on the basis of their wine purchases was extracted from the transaction records of 30,000 customers over an 8-month period. About half of these customers had bought wine from the supermarket at least once, indicating a large potential audience for a wine recommender. For our clustering, we selected the approximately 4000 customers who had bought at least 4 different wines; this 27% of the wine buyers accounted for 75% of the total wine revenue. We then narrowed the list of 600 wines down to the 10 most widely purchased. We reduced the number of wines in order to increase the cluster sharpness, and we chose the most *widely* purchased wines in order to increase chances that participants in the wine taste survey (Figure 3 b) would recognize at least one wine on the list. 75% of the 4000 customers above had bought at leat one of these 10 wines, and it was those 3000 customers that we used for clustering.

We applied Intelligent Miner's neural (Kohonen) clustering method [8], [9], [10] to the wine spending data (we used the built-in normalization option, which scales values below the mean from 0 to 0.5, and values between the mean and the maximum from 0.5 to 1.0). The choice of the number of clusters sets the degree of personalization. The results for 4 clusters are shown in Figure 2. and Table 1.

As we said in one of our other publications [2], we have not found the traditional measures of cluster quality or "validity" such as the Condorcet criterion or Dunn's index [12] to be

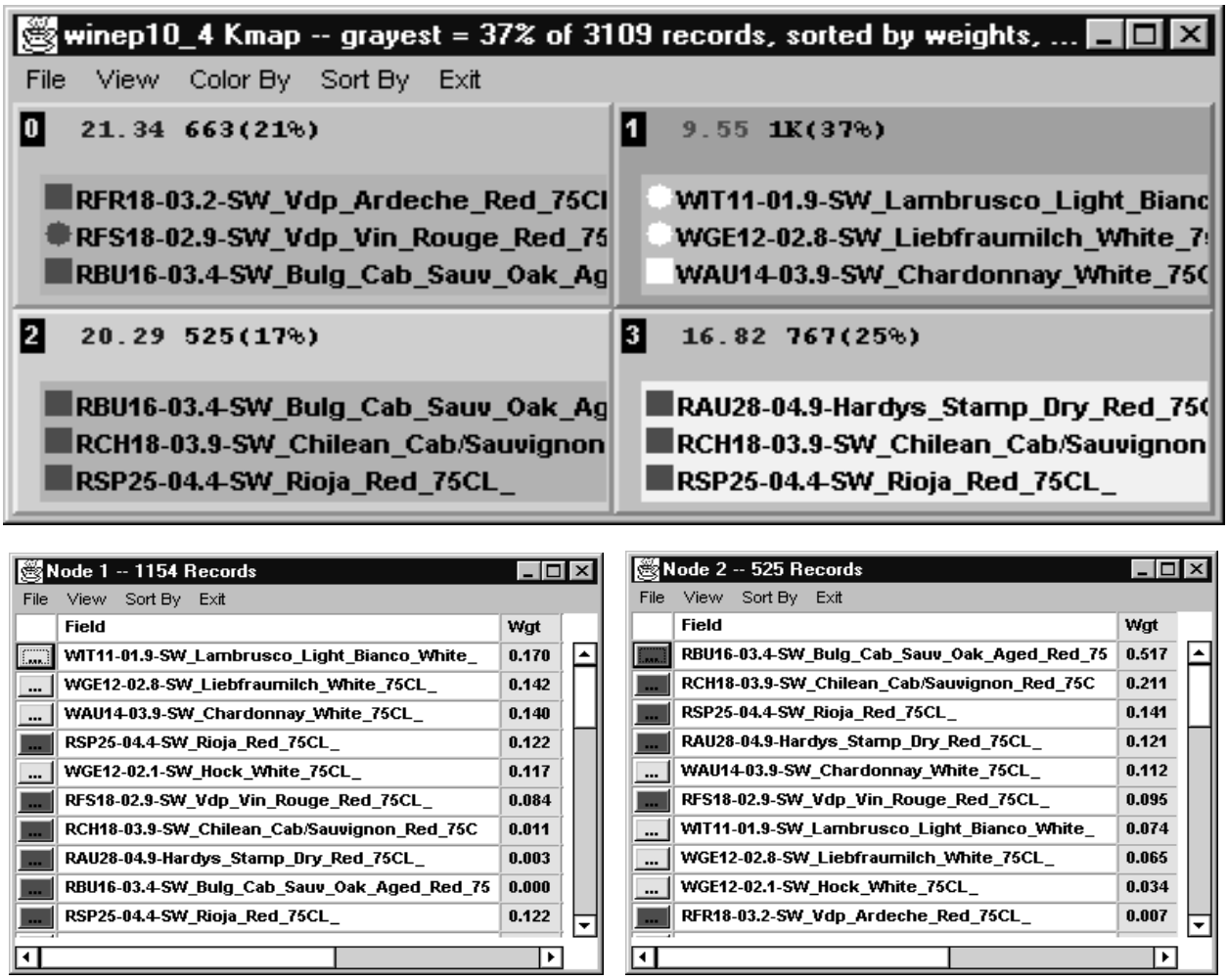

Figure 2: Dividing customers into 4 clusters already reveals rather different tastes in wine. Cluster 1 is dominated by people with a strong preference for white over red wine. People in cluster 2 have a strong preference for Bulgarian over Chilean Cabernet Sauvignon, but people in cluster 3 have an even stronger opposite preference. See the accompanying tables for more detail. (In this figure, "weight" and "Wgt" refer to neural weights, i.e., the mean wine buying attribute strengths of the customers in each cluster.) The alphanumerics preceding the wine names contain information about the wine's color, origin, wine-weight, and price. Dark bullets indicate red wines, light ones whites. The bullet shape in the main map indicates price range – squares cost more than circles. The upper left of each rectangle in the main map lists the cluster's average spending and population.

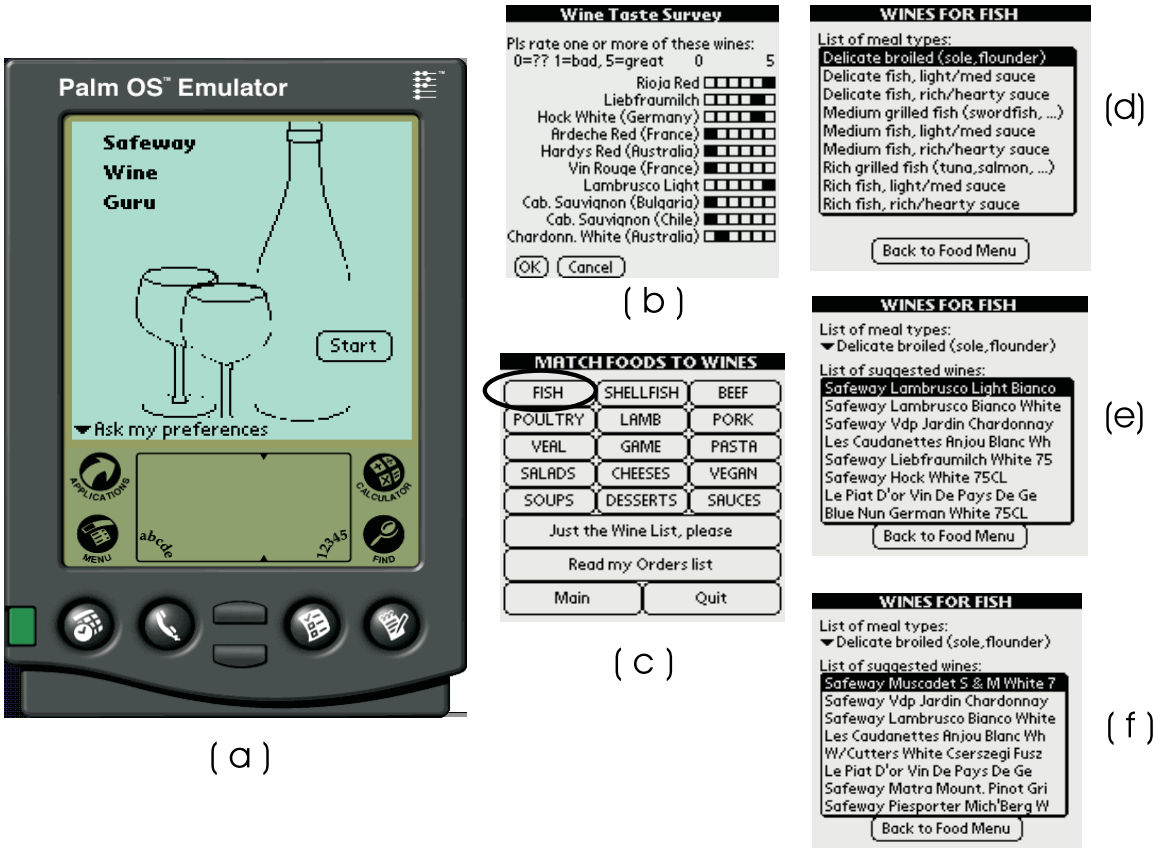

Figure 3: An example of the wine recommender in action (*clockwise from top left*): Customers selecting the "ask my preferences" option on the opening panel (a) are presented a mini-survey (b) to determine which cluster's wine list most closely matches their tastes. Based on the answers shown here, this customer is assigned to cluster 1 (see Figure 1). On the meal type menu (c), the customer selects fish, and then "Delicate broiled (sole, flounder)" from the preparation menu in (d), which leads to the wines recommended in (e). A strong vote for Bulgarian Cabernet Sauvignon in (b) would have resulted in assignment to cluster 2 and a different list of white wines (f).

very useful for this kind of data, so we provide some alternative measures in Table 2. "Wallet share" is the number of customers in the cluster who bought a wine divided by the total number of customers who bought that wine. "Wallet share enrichment" is wallet share divided by the fraction of the total population that the cluster represents. For example, chances of finding an Ardeche Red drinker in cluster 0 are 4.6 times higher than among wine buyers as a whole.

Rather different wine tastes are evident even with this small number of clusters: customers in cluster 1 strongly prefer white wines over reds. Customers in clusters 2 and 3 both like Cabernet Sauvignon, but those in cluster 2 strongly prefer Bulgarian over Chilean, while those in cluster 3 have an even stronger opposite preference. Customers in cluster 1 and 3 are more likely to buy a premium wine (over £9.00) than customers in clusters 0 and 2.

The wine lists that result from this clustering are reasonably distinct. A given pair of clusters has at most 56% overlap in table wines, and at least 25% of the table wines on each cluster's list are unique to that cluster, which we deemed to be satisfactory for current purposes.

4.2 Performance and Memory Requirements

The initial loading of the Java classes and the parsing of a 50-item shopping list each take about 2 seconds on a Palm V; all other operations are completed in one second or less. Our own experience and that of Wabasoft both agree that a 3-5X improvement can be obtained by recoding in C++, if needed. The program memory limit on the Palm V was adequate – we used it as follows: 64KB for the object heap (the maximum allowed), 2KB for the main stack, and 300 bytes for the native stack. The class heap took 30KB of the static memory. The Waba VM takes 100KB, the wine recommender takes 95KB, and the wine files take about 14KB.

Initial reaction from wine enthusiasts of our acquaintance has been encouraging, and we are looking forward to tests with real customers.

5 Summary and Future Work

What we have shown, in essence, is a simple demonstration of a pervasive computing system that uses both collaborative and content filtering to provide personalized wine recommendations, with the content filtering performed entirely on the PDA and the collaborative filtering shared between the server and the PDA. With the computing power of PDAs sure to increase, we believe that there is a great opportunity for such systems of intelligent agents/assistants that combine the information sources and computing capabilities of servers and PDAs in order to provide a variety of personalized services.

6 Acknowledgements

We are grateful to our colleague Richard D. Lawrence for his encouragement of this project, and for many helpful discussions with him and with Liz Robertson of Safeway plc.

Wine		Customers per cluster no.			
Type	Name	0	1	2	3
Red	Rioja Red	162	271	140	200
White	Liebfraumilch	82	*317*	66	34
White	Hock White	70	264	34	23
Red	Ardeche Red	**663**	0	7	18
Red	Hardys Dry Stamp	72	6	118	**467**
Red	Vin Rouge	*231*	191	98	69
White	Lambrusco	111	**378**	74	44
Red	Bulgarian Cabernet	195	1	**525**	1
Red	Chilean Cabernet	150	24	*215*	*465*
White	Chardonnay	97	312	110	174

Table 1: Cluster distribution of customers who bought at least one bottle of the wine indicated. The numbers in **bold** and *italic* are the customer counts corresponding to the wines with the largest and second largest neural weights, respectively, in each cluster.

	Cluster no.			
	0	1	2	3
Fraction of customers	0.21	0.37	0.17	0.25
Most significant wine	Ardeche Red	Lambrusco Light	Bulgarian Cabernet Sauvignon	Hardys Stamp Dry Red
Wallet share	0.96	0.62	0.73	0.70
Wallet share enrichment	4.6	1.7	4.3	2.8
Premium red spending (%)	10	34	23	33
Premium white spending (%)	11	46	13	30

Table 2: Summary of cluster characteristics. The 21% of the total customers who fell into cluster 0 were 96% of the total who bought Ardeche Red, an enrichment over the database average by a factor of 4.6. (Premium wines here are those costing over £9.)

References

[1] Kotlyar V., Viveros M.S., Duri S.S., Lawrence R.D., Almasi G.S. "A Case Study in Information Delivery to Mass Retail Markets" In the Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA), Florence, Italy, August/September 1999. Published by Springer-Verlag as Lecture Notes in Computer Science, vol 1677

[2] R. D. Lawrence, G. S. Almasi, V. Kotlyar, M. S. Viveros, and S. S. Duri , "Personalization of Product Recomendations in Mass Retail Markets", to appear in *International Journal of Data Mining and Knowledge Discovery* special issue on E-Commerce and Data Mining.

[3] S. Franklin and A. Graesser, "Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents", Proceedings of the Third international Workshop on Agent Theories, Architectures, and Languages, Springer-Veerlag, 1996.

[4] J. Hendler, "Making Sense out of Agents", *IEEE Intelligent Systems*, March/April 1999, pp. 32-37.

[5] Waba Development Kit, *www.wabasoft.com*

[6] Wine Guru Demo, *www.ibm.com/people/a/almasi/winegurudemo.html*

[7] Al Borchers, Jon Herlocker, Joseph Konstan, and John Riedl, "Ganging up on Information Overload", *Computer*, Vol. 31, No. 4, April 1998, pp. 106-108.

[8] Intelligent Miner for Data, *www.ibm.com/software/data/iminer/fordata*

[9] T. Kohonen, "Self-Organizing Maps", Springer-Verlag, 1995.

[10] R. D. Lawrence, G. S. Almasi, and H. E. Rushmeier, "A Scalable Parallel Algorithm for Self-Organizing Maps with Applications to Sparse Data Mining Problems", *Data Mining and Knowledge Discovery*, Vol. 3, 1999, pp. 171-195.

[11] The Wine Spectator, *www.winespectator.com*

[12] J. C. Bezdek and N. R. Pal, "Some New Indexes of Cluster Validity", *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics* Vol. 28, No. 3, June 1998, pp. 301 - 315.