An Adaptive Interactive Agent for Route Advice

Seth Rogers
Claude-Nicolas Fiechter
Pat Langley
DaimlerChrysler Research and Technology Center
1510 Page Mill Road, Palo Alto, CA 94304-1135
+1 650 845 2500
{rogers, fiechter, langley}@rtna.daimlerchrysler.com

Abstract

Current route advice systems present a single route to the driver based on static evaluation criteria, with little or no recourse if the driver finds this solution unsatisfactory. In this paper, we propose a more flexible approach and its implementation in the Adaptive Route Advisor. Our system behaves more like a human travel agent, using driver preferences, when known, and working with him or her to find a satisfactory route. The route advisor predicts what route a driver will prefer based on a model of driver preferences, and, if the predicted route is unsatisfactory, it generates additional routes based on interaction with the driver. The route that the driver eventually selects serves as feedback to improve the preference model. We present a pilot study on using these route selections to construct a personalized model. As the preference model becomes more accurate, the need for interaction decreases and the driver receives better route advice.

1 Introduction

The state of the art in computer technology has advanced to the point where systems for generating driving directions between two points are commonplace. There are several web sites offering street-level driving directions, and several in-car systems available as an option on purchased or rented cars. The availability of digital maps and high-capacity, rapid processors enable this technology, but little attention has been paid to ensuring that the interface is flexible enough to deliver satisfactory routes to users who have different preferences.

Current systems for route advice compute solutions using a shortest-path algorithm to find the minimal-cost route from the origin to the destination. Some systems fix the cost as the estimated travel time, while others let the user choose between the shortest path, the quickest, or the "most scenic" one. In all cases, the system then describes the route to the user with little or no recourse if the driver finds the route unsatisfactory. These systems disregard the fact that driving occurs in a rich environment where many factors influence the desirability of a particular route. For example, some drivers may prefer the shortest route as long as it does not have too many turns, or the fastest route as long as it does not go on the highway. The relative importance of these factors varies among individuals, and drivers may not know themselves what they value most in routes.

In this paper, we describe the Adaptive Route Advisor, an adaptive user interface [6] that recommends routes from a source address on a road network to a destination address. We transfer the analogy of a trusted travel agent to our route advice domain. Given a travel task, the travel agent plans an initial solution or two, taking into account the client's preferences if known. After the travel agent presents the initial options, the client may request additional solutions that differ from the initial options along some dimension. For example, if the client is booking a flight, he or she may request a flight with a shorter layover, even if that means an increase in cost. The client and the travel agent continue to work together, generating and evaluating different solutions, until the client is satisfied. During this process, the travel agent can reflect on the choices made by the client and refine his or her model of the client's preferences for the next task. This interaction model is similar to that assumed by the Automated Travel Assistant [7] for airplane flights, except that system involves no personalization and the user must explicitly assign values to preferences.

The Adaptive Route Advisor is designed for in-car use. It is a Java application that functions as a resource-light network client, suitable for mobile environments with a wireless communication infrastructure. The remote servers provide resource-intensive functions such as routing and geolo-

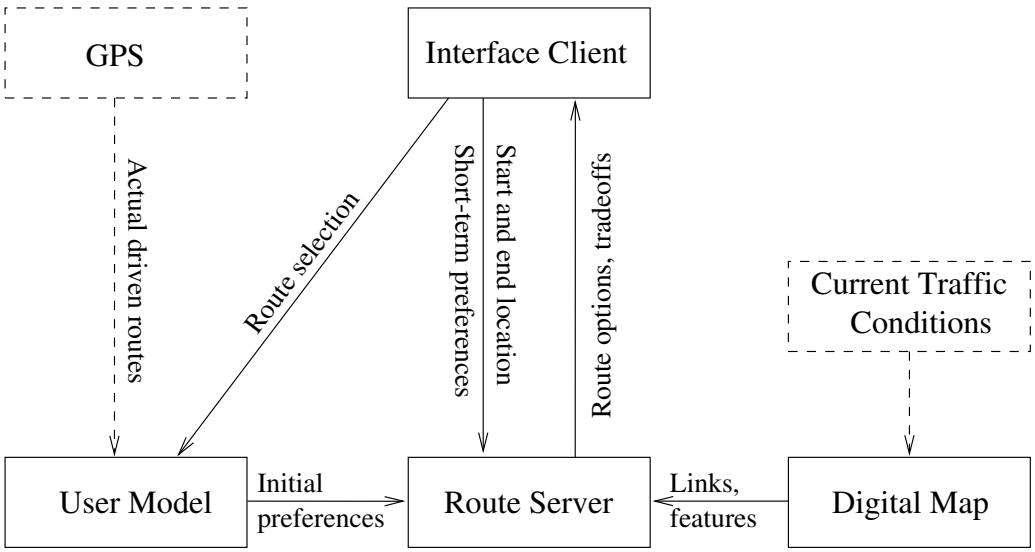


Figure 1: Architecture for the Adaptive Route Advisor. Elements with solid lines are already implemented, whereas elements with dashed lines are under development.

cation.[1] Although the current version does not yet take advantage of information available from mobile deployment (primarily current and past locations from the Global Positioning System) and the interface is not fully optimized for limited input and output resources common in vehicles, future work will embed the Adaptive Route Advisor in a mobile environment.

The pages that follow describe our approach in more detail and present the results of an experiment in personalizing the user model from rankings of alternative routes. First we present the overall system architecture, including the route generation component, the adaptation method that constructs the user preference model, and the user interface that presents route options to the user and gathers preference feedback. We then report on an experiment adapting a preference model to human subjects and its results. Finally, we outline planned improvements to the agent and consider the Adaptive Route Advisor's relevance to other advisory systems.

2 System Architecture

The Adaptive Route Advisor requires heavy memory resources to store the digital map and significant processing resources to compute an optimal route. However, we assume computational resources in vehicles will be limited in the near future. The client/server architecture shown in Figure 1 resolves this difficulty by offloading resource-intensive processes onto a remote server. This architecture also lets the routing system use information about the current traffic conditions, which would be available as a centralized service, as in the Intelligent Traffic Guidance System in Tokyo [3].

[1]Geolocation is mapping a plain English street location to its place in a digital map structure.

In the figure, portions in the current implementation are drawn in solid lines and planned extensions in dashed lines. The interface client is a resource-light process suitable for a vehicle's limited computational power that connects to the servers via a wireless TCP/IP connection. The route server receives requests from the client and uses the digital map to compute an optimal route according to preferences in the user model. Route requests may include short-term changes in the preference model to reflect unusual situations or corrections in the model.

The system initializes the agent with a default user model and refines this model with feedback from interaction with the interface. Future versions will also allow feedback from direct sensing of the driver's preferred routes using the Global Positioning System. Section 4 discusses these extensions in more detail.

2.1 The Routing Algorithm

The generative component of the Adaptive Route Advisor is a routing algorithm that plans a path through a digital map from a starting point to a destination. The planner represents the digital map as a graph, where the nodes are intersections and the edges are parts of roads between intersections. Our digital maps provide four attributes for each edge: length, estimated driving time, turn angle to connected edges, and road class (e.g., highway, freeway, arterial road, local road). The planner refers to these digital maps to minimize the weighted sum of the driving time, distance, number of turns, number of intersections, and distance on each road class.

The routing algorithm finds a path from a designated source node, usually the current position, to a designated destination. The cost of an edge is computed as a weighted

sum of its attributes,

$$c = \sum_i (w_i \cdot a_i).$$

The weight vector plays the role of a user preference model that defines the relative importance of the attributes. The system uses an optimized version of Dijkstra's shortest path algorithm [2] to find the path with the minimal sum of the costs for the edges in the path.

2.2 Constructing the User Model

Although weighting each edge attribute creates a flexible cost function for the planner, the space of possible models is a continuum with as many dimensions as there are attributes. It would be difficult and inconvenient for a user to specify his relative preference for each attribute. Instead, our system automatically induces a driver's preferences from his route choices. We have implemented a perceptron-style training algorithm [8], which we call the *differential perceptron*, that processes a sequence of interactions with the planner and produces a weight vector that models the preferences expressed. In this way, as the driver uses the interface, it adapts itself to his preferences. This is reminiscent of the way in which Hermens and Schlimmer's system for filling out repetitive forms [5] adapts itself to a particular usage pattern and predicts defaults values based on those observed in previous interactions.

We define an interaction with the planner to be the presentation of a set of N generated routes and feedback from the user indicating which route is preferable. This is completely unobtrusive to the user, because he or she evaluates a set of routes and selects one as part of the route advice process. For training, we expand the interaction into $N-1$ pairs, representing the fact that the selected route is preferable to each of the presented alternatives. These training pairs can be used to improve the user model in a simple manner. If, out of the two routes in a training pair, the route preferred by the current user model is not the one the user selected, the adaption method increases the weights corresponding to the features in the selected route and decreases those corresponding to the features in the other route.

More precisely, the system represents routes with a vector $\vec{x}$ containing its measurable attributes. Given an initial weight vector $\vec{w}$, it estimates the cost of a route to be the linear product $c = \vec{w} \cdot \vec{x}$. If route $\vec{x}_1$ is rated better than route $\vec{x}_2$ and the cost of $\vec{x}_1$ is lower than that of $\vec{x}_2$, the weights are consistent and do not need modification. If the cost of $\vec{x}_1$ is higher than that of $\vec{x}_2$, the system applies the differential perceptron update rule to $\vec{w}$, which decreases the cost of $\vec{x}_1$ and increases the cost of $\vec{x}_2$ using

$$\Delta\vec{w} = \eta\vec{x}_2 - \eta\vec{x}_1 = \eta(\vec{x}_2 - \vec{x}_1),$$

where η is the learning rate. For each pass through all available training data, the learning algorithm adds $\Delta\vec{w}$ to $\vec{w}$ and continues running through the training data until the weights stop changing or it has performed a maximum number of iterations, which is set to 5,000 in the current system.[2]

Once the differential perceptron algorithm finds a weight vector that best predicts preferable routes as a weighted sum of attributes, the routing algorithm uses this weight vector in its cost function. Since the routing algorithm is optimal on the cost function, the resulting route is guaranteed to have the lowest cost for that user model among all routes between the same two nodes. In other words, the routes computed are always Pareto optimal, in that there can be routes that are better along each of the dimensions (attributes) independently, but none that can be better simultaneously on all dimensions.

2.3 The Interaction Module

When started, the Route Advisor client locates the servers it needs and displays a route request screen, like the one pictured in Figure 2. In the current implementation, the user specifies origin and destination in a postal address style, and identifies him/herself for the purpose of loading the user model. An in-car implementation could simplify this screen by providing the current location as a default starting point and the most frequent car driver as a default user identity. The driver could select a destination from a list of most common destinations.

After requesting a route, the main interaction window appears, as displayed in Figure 3(a), providing a list of current route options and two menus, "Route" and "Modify." The current routes are presented in terms of nine attributes: total time, number of intersections, number of left turns, right turns and U-turns, total distance, and distances on three classes of roads. Initially the agent presents two routes to the user. The first uses the current preference model as the weight vector for the routing cost function. The second route uses novel weights, selected from a small set of prototypical user models, in an attempt to explore new directions in the space of preference models.

Presenting at least two route options forces the user to make a choice and provide some feedback to the agent. The turn directions for the selected route are shown in the field below the route list and the map displays the selected route, as shown in Figure 3(b). Clicking "Select" indicates that the highlighted route is satisfactory and closes the window. The route advisor assumes that the highlighted route is preferable to the alternative routes and updates the user model. Clicking "Cancel" closes the window but does not update the model.

The "Modify" menu lets the user generate a new route that is faster, shorter, has fewer turns, has fewer intersections, or has less or more highway than the selected route.

[2] Although the system can update the perceptron on-line after each new training example, the experiment described in Section 3 trains on a fixed set of examples.

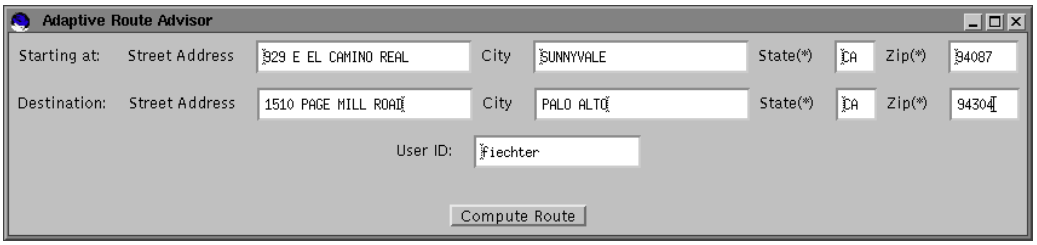
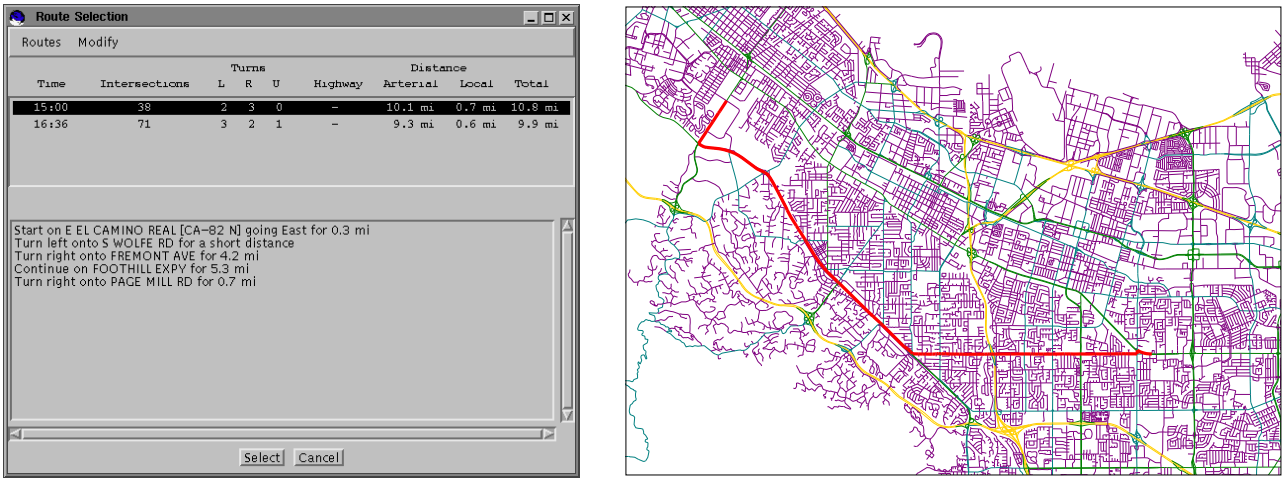
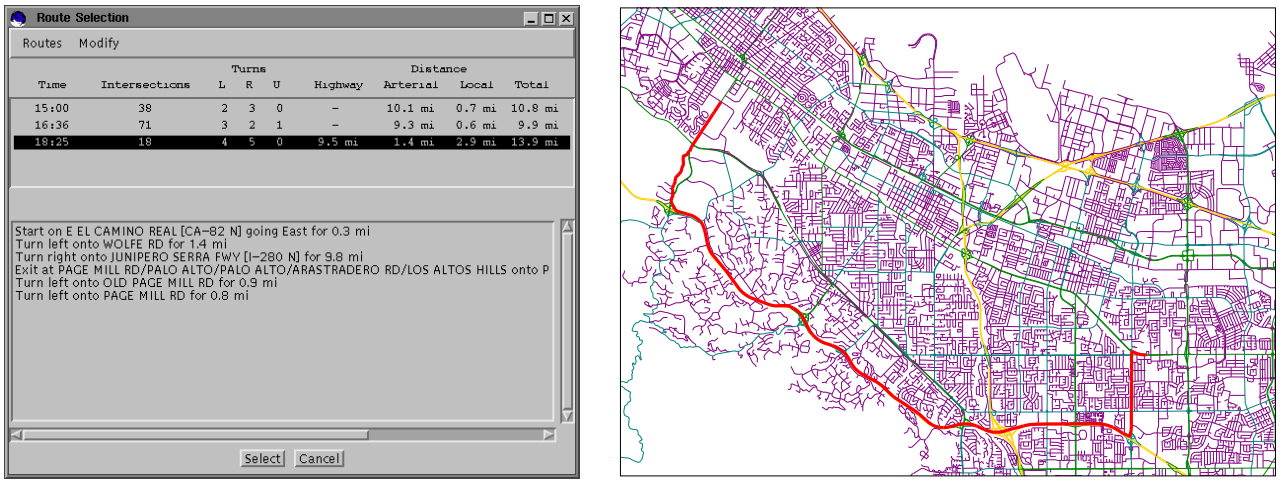


Figure 2: The route request window for the Adaptive Route Advisor.



(a) (b)

Figure 3: Initially, the user is shown two alternative routes, the best route according to the current user model being highlighted in the route selection window (a) and displayed in the map window (b).



(a) (b)

Figure 4: The user can generate additional alternatives by selecting a route in the selection window (a) and choosing a modification from the "Modify" menu; the resulting route also appears in the map window (b). Here the user generated the third route by selecting the first choice in the selection window and choosing "More Highway" from the "Modify" menu.

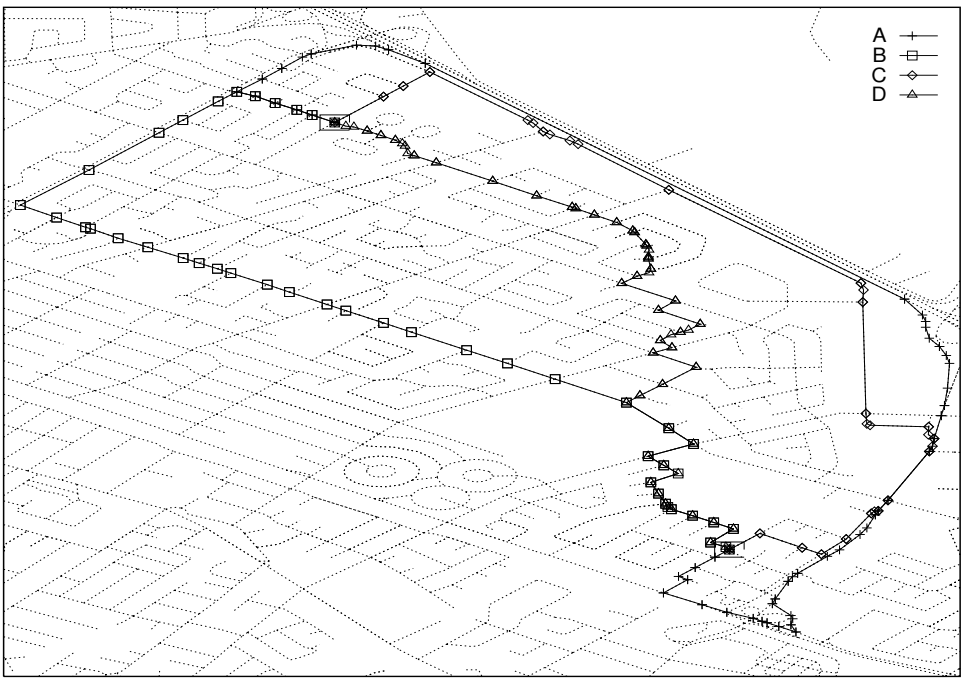


Figure 5: Sample task for the subjects. The starting point is the box at the upper left and the ending point is the box at the lower right. A is the route with fewest turns, B is the fastest route, C is the route with fewest intersections, and D is the shortest route.

The implicit assumption is that the driver is willing to accept routes that are somewhat worse on other attributes if he or she can find one that is better on the selected attribute. This approach to navigating through the space of possible solutions is similar to "tweaking" in Burke et al.'s RENTME system [1]. In that system, the user can navigate a database of apartments for rent by asking for an apartment that is either cheaper, bigger, nicer, or safer than the one currently displayed.

The Adaptive Route Advisor searches for new routes that satisfy the improvement request by modifying the weights it places on attributes, increasing the weight of the selected attribute, and decreasing the other weights. Since slight changes in the weight vector may result in the same route, the system continues modifying the weights until the resulting route is different. For example, Figure 4 shows a route with more highway that has been added to the route list. If the user is unsatisfied with all the alternatives listed, the "Route" menu lets the user generate an entirely new route as different as possible from all those displayed. The route advisor does this by adding a "penalty" in the cost function to all segments used by one of the displayed routes.

The interface described above simultaneously and seamlessly fulfills two functions in the Adaptive Route Advisor. First, it lets the users easily find routes they like by giving them choices and letting them interactively modify the routes proposed. Second, it unobtrusively collects the information that the learning algorithm needs to refine the user model and adapt to a particular driver.

3 Testing the Adaptation Algorithm

In order to test the adaptation algorithm apart from the other functionality of the Adaptive Route Advisor, we simulated a series of interactions on paper with human subject evaluations of planner output. The test consisted of 20 tasks that involved trips between intersections in the Palo Alto area. To compensate for the lack of interactivity, we produced four routes for each task instead of two. Since we had no opportunity to build user models, we used exploratory weight vectors with a unit weight for one attribute and zero for the rest, creating routes optimized for time, distance, number of intersections, and number of turns, respectively. We plotted the four routes, labeled randomly A through D, on a map of Palo Alto. We presented the tasks in a different random order for each subject. Figure 5 shows an example of one of the tasks and its four route choices.

We asked the subjects to evaluate the routes for each task and rank them in preference order, using 1 for best and 4 for worst. Since a ranking of four routes gives six independent binary preferences (A better/worse than B, C, D; B better/worse than C, D; C better/worse than D), each subject provided $6 \cdot 20 = 120$ training instances.

We trained the perceptron for 100,000 epochs ($\eta = 0.001$) for each subject, then looked for some way to compare the resulting user models. Since the cost of a route is a relative measure, the relative values of the weights are more informative than the absolute values. We will refer to the ratio of two weights between two attributes as their *exchange rate*,

because they define how much of one attribute a driver is willing to give up to improve another attribute. For example, if the exchange rate between time and turn weights is 30, the driver is willing to drive up to 30 seconds longer to save one turn, but no more. Figure 6 shows the exchange rates between distance and the other three attributes.

The results indicate that route preferences differ widely across people. Some subjects, such as 11 and 16, are apparently willing to go to great distances to improve their route on some other attribute. Other subjects, such as 9 and 17, would sacrifice other attributes to reduce the distance attribute. The most surprising result is that many subjects have *negative* exchange rates. For example, the distance/turns exchange rate for Subject 10 is -1027. This means that, given two routes A and B, if route A has one more turn than route B, it will have a lower cost if it is more than 1027 feet *longer* than B. Besides its intuitive difficulties, it is inconvenient to use these weights directly for planning because it means some edges could have a negative cost. We believe these negative weights come from the bias in the training data toward optimal routes on some attribute. For example, the fact that drivers prefer shorter routes, other factors being equal, is not explicitly represented in the training data. Our future work will include using such background knowledge to eliminate negative exchange rates.

To evaluate the advantage of using a personalized model versus a single fixed model, we also created an aggregate training set of all $120 \cdot 24 = 2880$ instances. Figure 7 compares the accuracy of the personalized model to the aggregate model. As expected, the accuracy of the aggregate model is poor, hovering around chance (50%), even though it was learned from 24 times as much training data as individual models. The personalized model is uniformly better than chance and the aggregate model, but still far from perfect. Some possible sources for this model failure are that people are inherently inconsistent or that our model space does not represent some important attributes in drivers' route preferences. For example, people may dislike a certain road or intersection, which affects the rankings for some tasks but not others. Future studies will include additional information about the routes and measure the subjects' consistency on redundant tasks.

4 Directions for Future Work

The results of our initial experiment indicate that it is possible to learn a cost function that predicts driver preferences with reasonable accuracy. More importantly, this cost function serves as a user model for generating routes that will be satisfactory to the driver. The Adaptive Route Advisor can be made more powerful and useful through additional work in five key areas: use of personalized attributes, better street descriptions, use of direct driving feedback, a more effective interface, and better model induction.

One source of error in the experiment was the limited and impersonal nature of the route descriptors. As Haigh and Veloso [4] note, the descriptor set may not represent all factors relevant to a driver. An in-car navigation system is particularly well situated to use personalized attributes, because it can constantly monitor the driver's behavior using traces from a Global Positioning System. In particular, we can assume that the routes a person drives are desirable by that person's true internal cost function[3] and use information about familiar routes when planning new ones.

The planner could represent familiarity as a binary visited/not visited value for each edge, or it could try to represent the degree of familiarity as a continuous value. However, with an additional assumption that sequences of familiar edges (subroutes) are more desirable than isolated familiar edges, we have developed a familiarity preprocessor [9] that groups sequences of road edges between commonly used intersections into higher-level links, similar to disjunctive macro-operators. A macro link between two intersections represents all distinct routes the driver has used between these intersections. These macro links are hierarchically organized, with some links recursively incorporating smaller macro links. The largest macro links represent entire trips, such as the drive from home to work. Including these macro links affects the planner in three ways: it uses sequences of familiar edges as primitives, it shortens the edge-by-edge description of the route by summarizing familiar sequences, and it biases the route description toward using familiar segments.

We can improve street descriptions by accessing existing geographic databases and by generating new ones. Current databases provide information about the location and types of businesses, as well as demographic information. In future work, we will generate new geographic databases by collecting and analyzing traces of trips from a Global Positioning System. Analyzing the trajectories of many cars along the same edge provides average speed models for different times of day, the location of traffic controls, and number of lanes. An advantage to the client/server architecture is that clients can serve as a distributed sensor network to sample road conditions and provide dynamic updates to the digital map for more accurate routing. Some possible dynamic attributes include transit time, congestion, and road or lane closures.

Besides interacting with the interface, another form of feedback comes from observing the routes actually driven. If the driver does not take the route the user model predicted, the new route is presumably better than the predicted route, and this will generate a new instance for the personalization module. This type of feedback may include more classification noise than direct feedback because there is no direct evidence that the driver liked his route or even that he or she was not lost. However, if the driver usually follows

[3] Situations in which this assumption does not hold include cases where the driver is lost and where he is following directions.

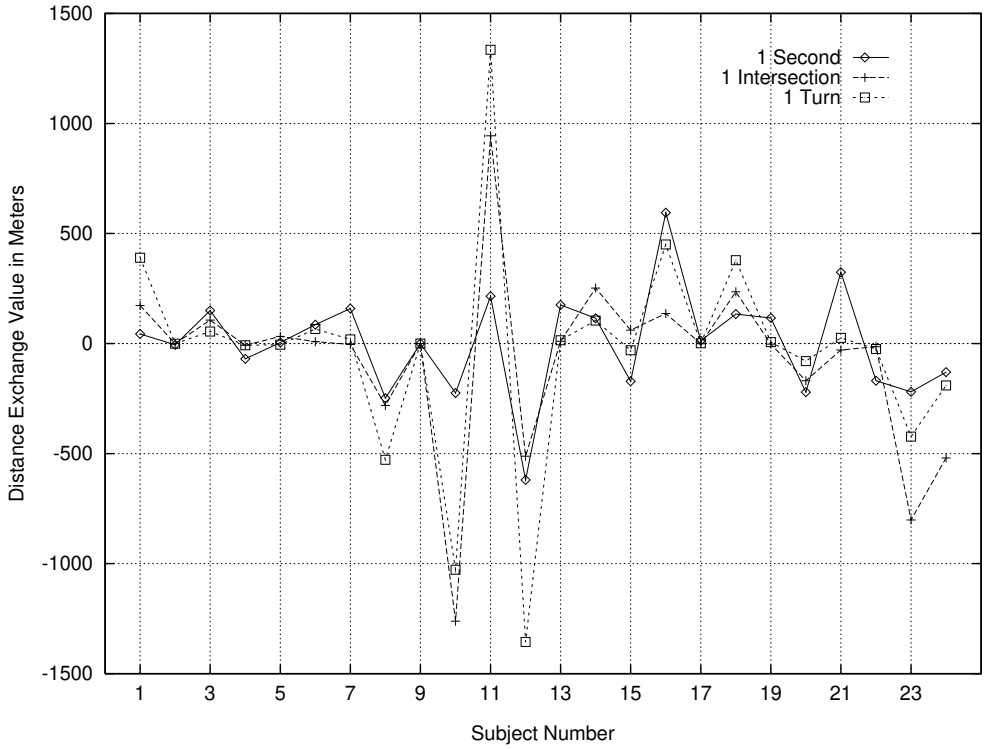


Figure 6: Exchange rates for three of the attributes with respect to distance, computed from all the data for each subject. High positive values for an attribute indicate that shorter distance is less important than reducing that attribute, near zero values indicate that shorter distance is more important, and high negative values indicate that longer distance is more preferable.

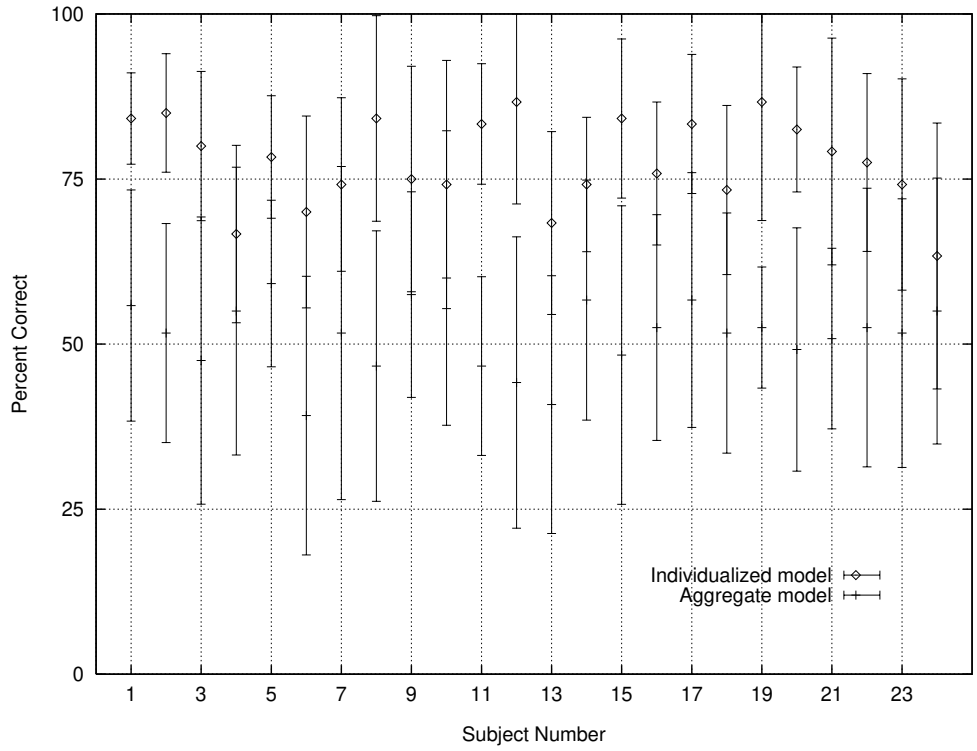


Figure 7: Comparison between the accuracy of the personalized models and that of the aggregated model. The accuracy was computed using a ten-fold cross validation. The error bars mark one standard deviation.

routes because of his own preferences, the noise should cancel out after sufficient training data. These indirect forms of feedback are less intrusive than that required by our current system, where the user must explicitly select the route he prefers.

The current user interface is tuned to exhibit the functionality of the agent. To deploy the Adaptive Route Advisor in a car, we will need to partly redesign the interface to take into account the limited input and output facilities. For instance, the menu for modifying the routes might be replaced by a panel of buttons that the user can activate through a touch screen. We will also need to evaluate the in-car user interface with drivers to ensure that the capabilities of the route advisor are easily and intuitively available to drivers.

We are also exploring other inductive methods for adapting the user model, such as regression over the preference rankings, multi-layer neural networks, and principal components analysis. A critical property of prospective methods is that the model be able to generate a numeric cost for partial and complete routes. We are also investigating more flexible model representations, such as adjusting the weight vector based on task characteristics. For example, a driver may always want the fastest route to work but prefer a more leisurely drive home. Results from any method could improve with some background knowledge about the domain and more relevant attributes for the street descriptions. We can improve our evaluation by determining the fraction of modeling errors that are due to driver inconsistency, which we can measure by including some redundancy in our experimental tasks. Our final goal is an agent with a flexible, usable interface that accurately adapts itself to its user over time.

5 Conclusions

Route recommendation for driver is a knowledge-rich problem where the criteria for making decisions (the attributes of the edges) and the relative weight of the attributes (cost function) can be personalized. The Adaptive Route Advisor serves as a intermediary agent between the driver and the complex digital map. The agent and the driver interact to generate multiple route options, giving the driver a more satisfactory route than he or she would receive from a single-option route planner, and providing feedback from the driver that reflects his or her route preferences. The agent encodes these preferences in a user model that the agent uses to predict which route the driver will find most appealing.

Although interaction is in the driver's best interest if he or she wants a satisfactory route, the agent does not require it, and ideally interaction will become less necessary as the agent better approximates the driver's cost function. This low interaction requirement is crucial for in-car decision making where the driver's attention is necessarily focused elsewhere.

In general, our approach to developing advice agents is to automatically and unobtrusively acquire value judgments by observing the user's actions in a domain, and to utilize interaction as an additional source of value judgments. The agent generates a solution using its current user model, receives feedback from the user if its model is inaccurate, and corrects its model in areas relevant to the problem being solved.

Acknowledgments

The authors would like to thank Daniel Russakoff for preparing and running the experiment, and Renée Elio for many helpful comments and discussions.

References

[1] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. Knowledge-based navigation of complex information spaces. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 462–468, Portland, OR, 1996. (Cambridge, MA: AAAI Press/MIT Press).

[2] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[3] Peter Hadfield. Smart cars steer round traffic jams. *New Scientist*, April 26 1997.

[4] Karen Zita Haigh and Manuela M. Veloso. Route planning by analogy. In *Proceedings of the International Conference on Case-Based Reasoning*, pages 169–180, Sesimbra, Portugal, 1995. (Berlin, Germany: Springer-Verlag).

[5] Leonard A. Hermens and Jeffrey C. Schlimmer. A machine-learning apprentice for the completion of repetitive forms. *IEEE Expert*, 9:28–33, 1994.

[6] Pat Langley. Machine learning for adaptive user interfaces. In *Proceedings of the 21st German Annual Conference on Artificial Intelligence*, pages 53–62, Freiburg, Germany, 1997. Springer.

[7] Greg Linden, Steve Hanks, and Neal Lesh. Interactive assesment of user preference models: The Automated Travel Assistant. In *User Modeling: Proceedings of the Sixth International Conference*, pages 67–78, Vienna, New York, 1997. Springer Wien New York.

[8] Nils J. Nilsson. *Learning machines*. McGraw-Hill, New York, 1965.

[9] Seth Rogers, Pat Langley, Bryan Johnson, and Annabel Liu. Personalization of the automotive information environment. In *Proceedings of the workshop on Machine Learning in the real world; Methodological Aspects and Implications*, pages 28–33, Nashville, TN, 1997.