Generating Navigation Information Based on the Driver's Route Knowledge

Hua Cheng, Lawrence Cavedon
CSLI, Stanford University
Stanford, CA 94305, USA
huac,lcavedon@csli.stanford.edu

Robert Dale
CLT, Macquarie University
Sydney, NSW 2109, Australia
Robert.Dale@mq.edu.au

Abstract

This paper targets the content selection problem in generating appropriate information in the domain of in-car navigation. It describes an algorithm that models driver's knowledge about roads and routes and uses this knowledge to tailor turn-by-turn instructions from a commercial routing service to those more suitable to the individual driver's background. This content selection component is one part of a domain independent generation system of a general purpose dialogue system toolkit. We claim that this type of adaptive generation facilitates more efficient and driver friendly navigation.

1 Background

With an increasing number of new devices finding their way into the car to address people's needs for communication, entertainment and telematics services, we need to ensure that these devices can be operated without increasing the risk of traffic accidents. Driving is an "eyes busy, hands busy" activity, and so spoken language is an obvious choice for interaction. At the same time, we cannot expect users to learn and remember complicated sets of commands, and so rather than rely on device-specific sets of keywords, we would like to ensure that the interface to these devices is via a relatively natural dialogue.

Our interest is in the design and development of a natural language dialogue interface for operating in-car devices and services that imposes the minimum cognitive load on the driver. We aim to do this by understanding the driver's requests and producing responses based on the driver's knowledge, the conversational context, and the external situation. We believe that highly interactive, robust and situation-aware dialogue enhances usability and safety by reducing cognitive load in the driving situation. The system we are developing is intended to allow the driver to operate in-car equipment, as well as allowing him or her to obtain navigation information (e.g., turn-by-turn instructions), and information about local facilities.

This paper focuses on the generation of navigation instructions. In general, we consider two decision processes particularly important in the driving domain:

- What to say: the system should generate instructions and related information necessary for the success of the navigation task, providing reaffirmation and necessary grounding, customized as appropriate for the current driver.
- When to say it: the system should generate appropriate navigation information at the right time, being cognizant of the potentially stressful situations such as missed turns and driving situations requiring high attentiveness, so that output that is not immediately critical is delayed.

These requirements demand an adaptive generation process, where the dialogue system tailors its output on the basis of characteristics of both the driver and the situation.

Most existing route description generation systems produce generic or at best stereotypical navigation information that only accounts for limited driver specific characteristics. (Höök, 1991) surveys the literature on driver modeling and concludes that an explicit user model is needed. She distinguishes three user prototypes corresponding to tourist, resident and commuter navigators: routes are chunked and presented in different ways to these user groups. (Pattabhiraman and Cercone, 1990) discusses the importance of salience and relevance in the content selection stage of route description generation; however, their notion of relevance is pertinent to the communicative goals of the generator rather than those of the user. (Fraczak et

al., 1998) distinguishes between *known* and *unknown* information, and suggests that the problem concerning *known* information is that of determining whether or not to make it explicit; however, their approach only deals with *unknown* information. Coral (Dale et al., 2003) aims at producing natural route descriptions using general purpose generation techniques. It features a typical generation architecture but emphasizes the role of the micro-planning level in reproducing route descriptions that are linguistically similar to those produced by humans. The system takes GIS data input, segments this to produce a hierarchical structure, and then uses aggregation and referring expression generation techniques to merge the data into coherent multiclausal sentences.

In the present paper, we focus on the "what to say" decision, especially the selection of navigation information to be presented to the driver, using a model of the driver's route knowledge. This content selection process is one component of a domain independent natural language generation system, currently under development as part of a dialogue system toolkit. While the dialogue system has previously been applied to multiple applications (Lemon et al., 2002), the in-car domain raises specific issues for generation, such as situation awareness, user adaptibility, incremental generation, conversation resumption, and time-sensitive generation.

The "when to say it" decision requires a model of the user's cognitive load. This might be estimated from a variety of sources, including instrumentation (e.g. sensors) and human factors (e.g. characteristics of speech (Muller et al., 2001)); developing a solution here remains a topic for future work.

This paper begins by motivating content-based user modeling for in-car applications in Section 2. We then describe how we represent the driver's route knowledge and use this knowledge for selecting driving instructions in Section 3. We also give a brief description of the general purpose NLG system that we are developing in Section 4.

2 User Modeling for Content Selection

Two main user modeling techniques have been used to adapt the behavior of systems in contexts such as user-specific webpage recommendation (Zukerman and Albrecht, 2001):

- *Content-based modeling* assumes that users exhibit particular behaviors under given circumstances, and that this behavior is repeated under similar circumstances. This is most useful when a user's past behavior is a reliable indicator of future behavior.
- *Collaborative modeling* assumes that people with related characteristics tend to behave similarly under the same circumstances. This is most useful when a user's behavior is similar to that of other (like-minded) users.

Existing generation components of dialogue systems have previously used collaborative modeling to predict a user behavior based on those exhibited by the group to which the user belongs. (Zukerman and Litman, 2001) identifies a number of user features often considered in the content planning module of existing NLG systems, including expertise or interests, preferences, user prototypes, and emotional state.

While it is easier to collect data from a large number of people, individual variations often reduce the predictability of the collaborative approach. In contrast, the content-based approach is more accurate for modeling individual user behaviors, but the lack of sufficient training data often affects its performance. Hence these two approaches can be combined to achieve better predictions (Zukerman and Albrecht, 2001): collaborative modeling is used for a new user because there is not enough evidence to support a reliable prediction, but as more evidence is collected, the system may switch to the content-based approach. In Section 2.2, we examine the special features of the navigation domain which motivate taking a content-based approach to modeling the driver's route knowledge.

2.1 Domain Specific User Knowledge for Dialogue

The hand-crafted hypothetical dialogue in Table 1, from our in-car application domain, illustrates some user modeling issues that arise.

Here, the user's background knowledge includes local route knowledge and driving preferences. This knowledge is important for tailoring the output to maximize its utility for the user. For example, knowing driver characteristics can affect the generated instructions in quite drastic ways. A description for the route from home to the freeway may just be reduced to something like that in S1 above if the system knows that this is a route already known to the driver. The

Situation	Dialogue
User driving from home	U1: How can I get to the Orbit Cafe in San Francisco? S1: Take Highway 101 and exit at Duboce Ave. It will take about 30 minutes to get there.
When approaching SF	S2: You'll be taking the Duboce Ave. exit, so stay left.
When approaching the exit	S3: The highway will end very soon and you will be on Duboce Ave.
System waits while user gets off the highway.	S4: Do you still remember how to get to Market St.? U2: Not really.
System gives turn-by-turn instruction.	S5: Follow this road, and turn right at Guerrero St. S6: Turn left here at Market St, and you have arrived at the Orbit Cafe.

Table 1: Hand-crafted sample navigation dialogue

system could also refer to previous driving experiences, as in *Take the freeway south; you'll take the same exit you took to get to Bob's house last week.* Such adaptive instructions are preferable to the detailed turn-by-turn instructions provided by most navigation systems in use today.

2.2 Content-based Modeling

A useful property of our application scenario is that the navigation system in any given car will typically be used by the same driver many times. This provides the system with the opportunity to collect a large amount of specific user data through both implicit observations and explicit interactions, in a way that is not true of many other dialogue scenarios. For example, a GPS-based navigation system can maintain a history of all the places the driver has been to, and use this to generate hypotheses about the driver's route knowledge. Based on this knowledge, the system can make reasonable assumptions about what the driver knows and predictions about what she needs to know. The following sources of evidence can be used to generate hypotheses:

Implicit observation: The driver's daily driving activities and use of clarification questions provide useful information. If the driver has driven to a place without asking for help, we can assume that she is familiar with the route between the origin and destination. If the driver queries how to implement a specific navigation instruction—for example, asking how to get to Highway 101 in reply to utterance S1 in the sample dialogue above—we can assume that she is not familiar with that route.

Explicit enquiry: If the system knows that the driver has been to a particular place, but is not sure if she still needs help, it can explicitly query the driver. For example, the system might say: *You should first drive to Highway 101. Do you know how to get there?* Utterance S4 in the sample dialogue above gives another example. Based on the driver's reply, the system can update its representation of the driver's route knowledge.

Since a large amount of driver data will be available, the dialogue system should be able to learn from this data and adapt its responses on the basis of the learned driver model. This permits the system to omit information that the driver is familiar with; a dialogue system that keeps on presenting information that the driver already knows is likely to cause irritation.

The wealth of user specific data that a system of this kind can accumulate makes the content-based modeling approach very appropriate; of course, a collaborative modeling approach can provide defaults when user specific data is not available.

3 Design of the Content Selection Component

This section describes a prototype system that can generate both navigation information and queries when it is uncertain. The implementation emphasizes modeling the driver's route knowledge. It uses canned text and templates for surface generation so as to focus on content selection.

3.1 Route Knowledge Representation

Generating personalized navigation instructions requires acquisition and representation of the driver's route knowledge. Existing representations log all intersections and roads via which a driver has driven. For example, the RouteCompiler system (Rogers et al., 1997) represents the routes that a driver has driven as a graph, where nodes are intersections and arcs are the roads between them. A context-free grammar is extracted from the graph to represent the driver's route knowledge at various levels of detail. This knowledge is used to bias a route planing procedure to favor familiar routes, thus minimizing the probability of getting lost. A problem with such representations is that they explode in size, and also use information that is not necessarily salient to the driver (e.g. crossroads driven past but not necessarily noted) to bias routes.

We represent a route by all the *decision points* (DPs) along the route. Decision points are intersections where the driver makes a turn, where street name or road status[1] changes, as well as origin and destination locations. Intersections where the driver simply drives straight through and there is no name change or road status change are not considered to be DPs. We claim that only such salient features to a route should be considered useful in biasing future route descriptions with respect to familiarity.[2] Our representation uniquely identifies routes while reducing the amount of information to be stored because it disregards all intersections that are not decision points and the road information between intersections.

For each DP, we specify the following features:

- *FromStreet*: the street from which the vehicle is to make a turn;
- *IntoStreet*: the street into which the vehicle will turn;
- *TurnDirection*: the direction the vehicle will turn, e.g., east, south;
- *Familiarity*: a measure of how familiar the driver is with this intersection, based on the number of times the driver has driven through this intersection from and into the same road, making the same turn;
- *LastVisit*: the last time that the driver was here.

The first three features uniquely identify a decision point. *Familiarity* is measured on a scale of 1 to 5, with 5 indicating high familiarity. This information determines whether a DP should be presented to the driver. *LastVisit* is used for two purposes: if the driver traveled through a DP a long time ago, we do not assume that she still remembers it; however, if she traveled the DP fairly recently, then this information can be used to remind the driver about this intersection or road by referring to her previous experience.

The following example demonstrates a DP where the driver turns east from Arastradeo Road into Foothill Expressway; she has made this maneuver twice before, and the last time was on January 27, 2004 at 3pm:

```
FromStreet: Arastradero Road, 94304
IntoRoad: Foothill Expressway, 94304
TurnDirection: east (left)
Familiarity: 2
LastVisit: 1/27/2004/15
```

DPs are stored and maintained in a knowledge base through Protégé (Musen et al., 1993), which is a tool for constructing and maintaining knowledge resources. For each routing task, the system examines all the decision points along the route. If a DP is new, it creates an entry in the knowledge base with a Familiarity value of 1; if the DP is already in the knowledge base, its Familiarity value is incremented by 1, until the maximum value of 5 is reached; if the driver asks a question about the DP or gives a negative answer to the system's query about the DP, its Familiarity value is decremented by 1.

For a new driver, the system has no knowledge of her background. As the interactions between the driver and the system increase, our model captures more and more driver specific features, and therefore provides more fine-grained and dynamic user modeling.

3.2 Implementation of Route Description Generation

We use a navigation web service provided by Robert Bosch Corporation to obtain routing information. For each routing request, the server returns a set of instructions as follows:

[1] Roads are assigned a status of 1 to 6, with 6 denoting interstates and freeways, 3 denoting expressways, and 2 being a default status for most city streets.

[2] This does not preclude using other features in the route descriptions themselves.

```
Input: Route instructions from the navigation web service
 1 Foreach DP on the route
 2     Retrieve driver's knowledge about the DP
 3     If Familiarity <= 2 (has little knowledge) or the DP leads to a
         major road (state highway or freeway)
 4         Mark it as fully describing
 5     Else if Familiarity <= 3 (has some knowledge)
 6         Mark it as briefly describing
 7     Else (very familiar)
 8         Mark it as disregarding
 9     End if
10 End foreach

11 Foreach DP on the route
12     If fully describing
13         If there are disregarded DPs before it
14             Describe the immediately previous DP first
15     Else if briefly describing
16         Find in the sequence of such DPs the last one
17         Give the instruction for that DP
18         If there are disregarded or skipped briefly described DPs before it
19             Randomly decide whether or not to query the driver
20             If driver gives positive answer or does not ask for details
21                 Continue with loop
22     Else
23         Continue with loop
24     End if
25     If driver gives negative answer or asks for details
26         Give instructions from the last described DP to the current one,
             skipping disregarded DPs
27     Else
28         Give the instruction for the current DP
29     End if
30 End foreach
```

Figure 1: Content selection algorithm

```
Driving Distance:  5.8 mile(s).
Driving Time:  9 minute(s).
Depart.  Go East on Miranda Ave.
Drive 0.2 mile(s), < 1 minute.
Turn left on Arastradero Rd.
Drive 1.0 mile(s), 1 minute(s).
...
Straight on Hospital Dr.  Drive
0.1 mile(s), < 1 minute.  Arrive.
```

The first line indicates the total travel distance and time. Each subsequent line provides a turn-by-turn instruction. The instructions are supplied as strings, and the application program is responsible for extracting the information it needs from these messages.

Given this detailed input, our system extracts information such as street names and turning directions from the strings and represents these as internal objects. It then selects which instructions to present to the driver based on its modeling of her route knowledge; it may aggregate multiple instructions as appropriate (see below). Finally, the driver's route knowledge base is also updated as navigation information is presented.

The content selection algorithm is shown in Figure 1. The first loop marks how a DP should be presented based on the driver's route knowledge, and the second loop presents the DPs to the driver.

The algorithm gives turn-by-turn instructions for a driver with little route knowledge, but for a local driver, it skips consecutive familiar DPs until encountering an unfamiliar or significant DP (such as a highway entrance, lines 3 and 4 in Figure 1). It summarizes the route segments

up to this DP (line 14), and then provides detailed instructions from that point on (lines 25 - 29). When there is doubt about the driver's familarity, the algorithm will skip consecutive DPs that do not need to be fully described, and only mention the last one (lines 16 and 17). It can then choose to either explicitly query the driver or say nothing unless the driver asks for more details. Drivers' route knowledge typically includes some local areas and several highways branching out from these areas. Therefore we expect route knowledge learned by our approach to consist of clusters of decision points in these local areas. These clusters plus the road classes facilitate natural route chunking, similar to that obtained by the chunking algorithm of (Höök, 1991). The numbers used in the algorithm are rather arbitary at present; we aim to substantiate appropriate values with data collected in our simulated driving environment in the future.

Aggregation is based on situation: we use the distance between decision points to simulate this. Two instructions on DPs whose distance is smaller than 0.1 miles are aggregated (for example, *Turn right here, stay in the left lane and turn immediately left*), as are adjacent instructions about the same DP (for example, *Turn right onto El Camino Real and drive east for about 3.5 miles*).

3.3 Sample Results

For a driver who has no knowledge of the area, our system generates the following output for directions:

```
Drive east on Miranda Ave.,
stay in the left lane and turn
immediately left onto Arastradero
Road.
Turn right onto El Camino Real and
drive east for about 3.5 miles.
Turn right onto Grand Road.
Turn right onto North Dr., and it
soon turns into Hospital Dr.
```

The distance information is only given when the distance between two DPs is greater than 2 miles. For a driver who knows how to get onto El Camino Real, which is a state highway, the system disregards instructions for the known sub-route and generates the following output:

```
Get onto El Camino Real east and
drive for about 3.5 miles.
Turn right onto Grand Road.
Turn right onto North Dr., and it
soon turns into Hospital Dr.
```

Two types of aggregation are featured in the sample output: conjunction is used to combine descriptions of the same DP or adjacent DPs (as in the first and last utterances of the sample output), and subordinated prepositional phrases are used to realize distance information.

4 Architecture

We are in the process of incorporating the capabilities of a domain independent generation architecture and enhancing it to handle the user and situation related issues discussed in the previous sections. Such an architecture typically assumes a pipeline architecture with three major components: a content planner, an utterance planner and an utterance realizer (Reiter, 1994). The content planner concerns the selection of information to be expressed and the organization of this information into a hierarchical structure that represents a globally coherent text plan. The utterance planner and realizer are concerned with the construction of utterance structures that convey the selected information, and the choice of grammatical features to produce locally fluent utterances. The user model affects all levels of generation and has a bearing on the variety and complexity of the text being generated.

Our goal is to produce conversational turns ranging from lengthy multi-clausal contributions to sub-clausal fragments, which we will refer to collectively as turns. Each turn can consist of several utterances, which should vary depending upon characteristics of the user. Figure 2 illustrates the generation architecture we are working on; we explain each major component below.

4.1 The Content Planner

There are, of course, different ways in which multi-utterance dialogue contributions might be generated. Research in text generation has addressed both top-down and bottom-up content planning strategies (Moore and Paris, 1993; Marcu, 1997). In a dialogue context, this decision is often determined by the nature of the dialogue domains. Some domains are highly structured and goal-driven, and the generation system has to come up with a text plan that conforms to the goal structure. In other domains, the goals are only loosely structured; this is the case in our in-car navigation system, where the goal is to effectively navigate the driver to the destination. Different text plans can achieve the same goal, so the generation

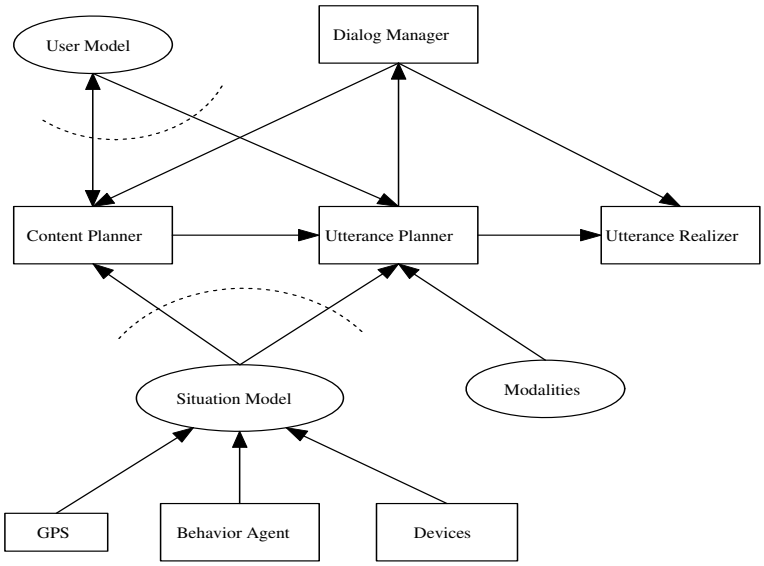

Figure 2: The generation architecture

system can choose a text plan that satisfies as many domain and user preferences as possible.

A generic content planner should be able to accommodate both needs, that is, to take either a communicative goal or a set of domain preferences as well as the semantic contents associated with these goals as the input, and produce a turn specification, which is a tree structure that contains the propositions to be expressed and the rhetorical relationships between them. The content selection component described in Section 3 is the first step toward a bottom-up content planner. This planner is invoked once per conversational turn, and makes reference to a dialogue model that is maintained over the entire dialogue.

4.2 The Utterance Planner and Realizer

The utterance planner takes a turn specification and produces a sequence of one or more utterance specifications, compatible with the realizer input specification language. Two important tasks of the utterance planner are aggregation and referring expression generation. These two tasks encounter new challenges in the in-car dialogue environment because of the time- and position-sensitive nature of the generation task. A referring expression or aggregated utterance can cause confusion if the driver's position changes dramatically during processing. Aggregation can also be used to opportunistically provide additional information (such as landmarks) to help identify domain objects, for example, *Turn left at Mathilda Avenue, where you can see an Arco gas station at the corner.*

The utterance realizer takes an utterance specification and produces marked-up text to send to the Text-to-Speech component. It should have the ability to address the requirements of echoing user language, prosodic markup and variability.

4.3 Interfaces to Other Dialogue Components

The generator draws on a number of other system components to provide the desired functionality.

Dialogue Manager: The Dialogue Manager stores the current context of the ongoing conversation, as well as the historical context. Each utterance, whether by user or system, is classified as a dialogue move, updating this context. The dialogue manager also constructs an 'Activity Tree', which represents all activities performed by the behavioral agent and their execution status.

Behavioral Agent: The user effectively converses with the behavioral agent. Requests from the user are communicated to the agent as goals for it to achieve. It executes the user's requests, and put the communicative goals on to the Dialogue Manager's Activity Tree, to become part of the Content Planning process. The Behavioral Agent is also responsible for prioritizing its communicative goals according to urgency and importance, and hence ensuring that critically important information is planned to be

communicated early by the Content Planner.

User and Situation Models: These components are responsible for influencing the generated utterances to account for current situation and user state. The Situation Model draws on multiple information sources, such as GPS, device sensors, etc. The User Model stores information pertinent to the current user, including the driver's route knowledge. It should be capable of adapting over time, by being given explicit preferences by the user and by observation.

5 Future work

The current implementation features the content selection algorithm in Figure 1 and some aggregation strategies, but uses canned text and templates for utterance realization. We are in the process of implementing the above domain independent generation architecture, designed specifically to deal with issues raised by the in-car domain, including situation awareness, user adaptibility, incremental generation, interrupted conversation, and time sensitive generation. We are also collecting data on situated direction-giving by human subjects, some portion of which will be used for evaluation purposes; evaluation will include different familiarity of users with specific routes.

Acknowledgments This project is partially supported by NIST ATP funding, Robert Bosch Corp., and VW of America. The authors would like thank Yao Meng, Karsten Funk and Fuliang Weng for making the Bosch navigation service available for our research.

References

R. Dale, S. Geldof, and J. Prost. 2003. Coral: Using natural language generation for navigational assistance. In M. Oudshoorn, editor, *Proceedings of the 26th Australasian Computer Science Conference*, Adelaide, Australia.

L. Fraczak, G. Lapalme, and M. Zock. 1998. Automatic generation of subway directions: Salience gradation as a factor for determining message and form. In *Proceedings of International Conference on Natural Language Generation*, Niagara-on-the-Lake, Canada.

K. Höök. 1991. An approach to a route guidance interface. Master's thesis, Licentiate Thesis, Dept. of Computer and Systems Sciences, Stockholm University.

O. Lemon, A. Gruenstein, L. Cavedon, and S. Peters. 2002. Collaborative dialogue for controlling autonomous systems. In *proceedings of AAAI Fall Symposium.*

D. Marcu. 1997. *The Rhetorical Parsing, Summarization and Generation of Natural Language Texts.* Ph.D. thesis, Dept. of Computer Science, University of Toronto.

J. Moore and C. Paris. 1993. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–695.

C. Muller, B. Grossmann-Hutter, A. Jameson, R. Rum-mer, and F. Wittig. 2001. Recognizing time pressure and cognitive load on the basis of speech: An experimental study. In *Proceedings of User Modeling*, Sonthofen, Germany.

M. Musen, S. Tu, H. Eriksson, J. Gennari, and A. Puerta. 1993. Protege-II: An environment for reusable problem-solving methods and domain ontologies. In *International Joint Conference on Artificial Intelligence*, Savoie, France.

T. Pattabhiraman and N. Cercone. 1990. Selection: Salience, relevance and the coupling between domain-level tasks and text planning. In K. McKeown, J. Moore, and S. Nirenburg, editors, *Proceedings of the 5th International Workshop on NLG*, Pennsylvania.

E. Reiter. 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the 7th International Workshop on NLG*, pages 163–170, Kennebunkport, Maine.

S. Rogers, P. Langley, J. Bryan, and A. Liu. 1997. Personalization of the automotive information environment. In R. Engels, B. Evans, J. Herrman, and F. Verdenius, editors, *Proceedings of the Workshop on Machine Learning in the Real World: Methodological Aspects and Implications*, Nashville, TN.

I. Zukerman and D. Albrecht. 2001. Predictive statistical models for user modeling. *User Modeling and User-Adaptive Interaction*, 11.

I. Zukerman and D. Litman. 2001. Natural language processing and user modeling: Synergies and limitations. *User Modeling and User-Adapted Interaction*, 11.