Generating driving directions for intelligent vehicles interfaces

Barbara Di Eugenio
Computer Science
University of Illinois at Chicago
Chicago, IL 60607, USA
bdieugen@cs.uic.edu

Michael J. Trolio
William J. Hughes Technical Center
Joseph Sheairs Associates
Medford, NJ 08055, USA
michael.ctr.trolio@faa.gov

1. Introduction

Much effort is taking place today to understand the best type of interface between a driver and his/her intelligent vehicle, e.g. the best combination of input / output modalities [1, 2, 3, 5, 6, 13]. In a scenario in which an in-vehicle system answers the driver's queries perhaps with a mixture of speech and graphical displays such as maps, there is an orthogonal issue that needs to be addressed: of all the information collected to answer the query, how much should be communicated and at what level of detail. In this paper, we discuss a system that provides driving directions (in written form) to a user, and its evaluation. Traditional literature provides an abundance of optimal algorithms that can be applied to a representation of a territory to generate text output for navigating between two arbitrary termini within that territory. In addition to the termini, the output generally consists of the minimal chain of intermediate landmarks and interconnecting roadways. By using simple templates, the algorithm can then realize this "essential" information into text. A prime example is MapQuest, the well-known web-based application (www.mapquest.com) that uses the NavTech geographical database for generating route directions within the United States. The final output, however, lacks the type of cueing that occurs in discourse between two people exchanging route information [4, 16]. In this paper, we show that providing those additional cues improves wayfiding performance: simulated drivers had fewer incorrect turns when following enhanced directions than those following essential directions.

We will describe the software system we developed, that can generate both essential and enhanced directios. It consists mainly of 1) a route generator that uses factual map data to generate routes, and 2) a natural language front-end. Using this software system and two sample groups, a single-factor experiment was conducted comparing the effectiveness of essential vs. enhanced form directions in wayfinding. Most notable in the experiment was the difference in incorrect turns, a measure of wayfinding; it approached significance in favor of the experimental group, namely, of the group that read enhanced directions. There were no other significant or marginally significant differences in the other measures we collected. More details on the software system and the experiment described in this paper can be found in [14].

2. Related work

There are two main bodies of work relevant to our work: the first concerns in-car information system, the second the linguistic aspects of directions and their computational modeling.

As regards in-car information systems, some work investigates the best combination of input / output modality to improve efficiency and effectiveness, and avoid driver distraction. Car driving is a safety-critical and hands-occupied activity in which the driver cannot spend extensive attention resources on screens or hand-held remote controllers. For example, [1] examines when a spoken language dialogue system should (not) listen to the speech and non-speech acoustics in the car; how to couple the in-car display with the dialogue system; and how to adapt to different drivers. [2] describes an in-vehicle Command&Control dialogue system, Linguatronic, used as an interface to the driver's cell phone. [3, 5] describe the VICO project, aimed at developing an in-car assistance system for accessing tourist information databases and obtaining driving assistance - the latter is of course very close to our own application. Finally, [6, 13] explore the cognitive models underlying human multitasking, specifically driving: driving can be decomposed into interleaved subtasks, but the primary task of driving is itself interleaved with secondary tasks such as radio tuning and phone dialing.

Works that focus on route directions can be categorized into two areas. One is descriptive and analyzes the overall structure of route directions during face-to-face discourse. For example, building on the notion of *deictic space* [8], [16] divides route descriptions into three stages, initial, in-

termediate and final. The output of this model is a generic set of schemas, where each set member represents a class of descriptive route text instances.

As regards computational work on directions, it focuses mainly on the development of route description generators. After analyzing a corpus of more than forty subway directions, Fraczak et al. [4] classify them into schemas containing varying amounts of essential and non-essential information. By using a route generator developed in Prolog, a single instance of directions could then be generated based on each schema type. [4] attempts at showing that salience (i.e., the relevance of each object to be mentioned in the directions) must be defined as a gradient of relative importance to the essential information as opposed to a simple inclusion/exclusion mechanism. Furthermore, this work demonstrates how the relative importance of non-essential route information maps onto linguistic form. [9] discusses the software engineering aspects of developing a route description generator, such as the coupling between the natural language generator and the domain-level task of collecting the essential information.

A route description generator is an instance of a Natural Language generation (NLG)system. A broadly used architecture for an NLG system is that of a pipeline which includes text planning, sentence planning, and sentence realization [11]. Text planning involves the determination of the basic content and structure of the text to be generated. More precisely, the output of the text planner is often a tree where the leaf nodes are basic propositions, the interior nodes are higher-order groupings of propositions, and the edges are relationships between the propositions. Sentence planning involves aggregation, i.e. how to appropriately group the information to be communicated, and lexicalization, i.e. word selection, of the text planning output in order to form sentences. Sentence realization transforms the output of the sentence planning module into grammatical sentences.

3. Generating directions

The geographic domain of the software system we developed is Napa County, California. The raw data comes from the TigerLine95 database maintained by the US Census bureau, which represents, at street-level detail, all charted areas of the United States. Information for each county is contained in seventeen interrelated text files that use various geometric techniques to represent features. Features can be represented as points, e.g. intersections of streets, as lines, e.g. roadways., or as polygons, e.g. parks and lakes.

The user can enter the route for which he needs directions via a simple interface, consisting of two windows (this is the interface we developed for our experiment, and we make no claims as regards its appropriateness for in-vehicle

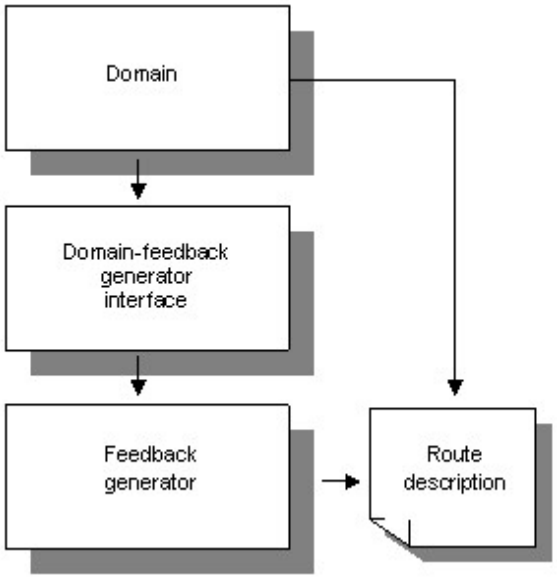

Figure 1: High level structure of the system

deployment – see Section 5 for some speculations on how this system could be deployed in a real time situation). The interface contains a main window, where the user specifies the route termini, and where the route is displayed; and a secondary window, where some parameters pertaining to how the route is rendered in language can be set (these parameters will be explained in Sec. 3.2.2). Figures 2 and 3 show the interface – Figure 2 shows an example of essential directions as generated from our system, and Figure 3 shows the same directions in enhanced form.

The design of our system is object-oriented with all entities being objects of classes. In general, every object belongs to one of three modules: the domain, the domain-feedback generator interface (dfg-interface), and the feedback generator. Figure 1 depicts the high-level relationship between these modules. A route description can be generated directly by the domain[1] (essential) or by the pipeline (enhanced). To compute the route the system uses the A* algorithm [12], on a map graph we built based on the Tiger-Line files.

3.1. Essential route descriptions

The domain can produce the essential route text by invoking an internal task that filters the information into the necessary nodes and transitions. The filtering step is necessary because the output produced by A* in the domain module is a superset of the information needed for the descrip-

1 Clearly the domain per se does not generate anything. We use this wording to refer to a few lines of code that verbalize the output of A*, based only on the domain.

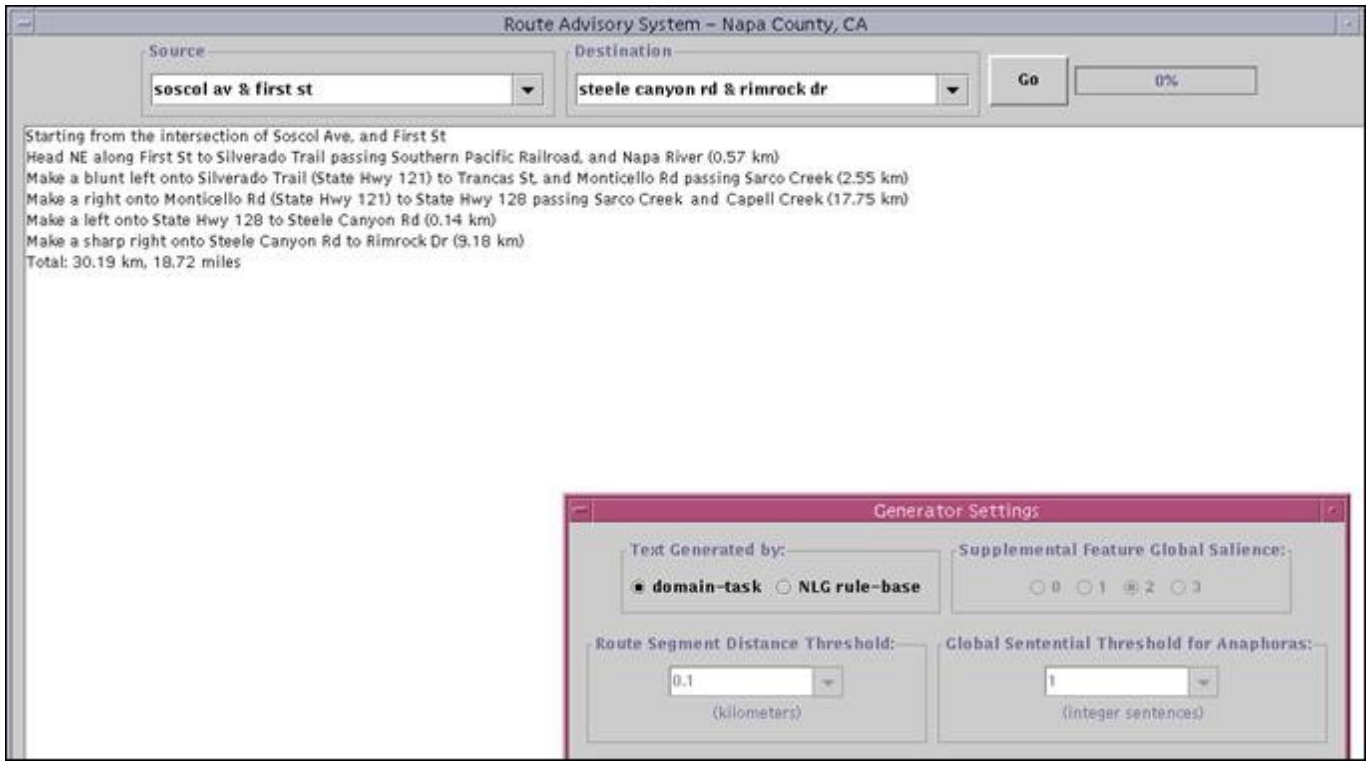

Figure 2: Essential route directions

tion of the route at hand. Then, using simple templates, the relevant information can be realized into text. Essential directions consist of main clauses realized in the following form:

```
turn <direction> onto <Node1-name>
until <Node2-name>
```

Additionally, all supplemental features collected along a transition are realized in a subordinate clause using the template

```
passing <feature1>, <feature2>,...
and <featureN>
```

Using these templates, the domain can create route text that conveys essential information (see Figure 2).

3.2. Enhanced route descriptions

Clearly there are many ways in which basic, Mapquest-style directions can be enhanced. The enhancements we chose to implement fall into two categories: aggregation and inclusion of supplemental information.

Aggregation pertains to how to appropriately group the information to be communicated. This is considered one of the fundamental tasks of an NLG system [11], since providing aggregation cuts down on repetitiveness and abstracts away from unnecessary detail. In our system, aggregation is limited to syntactic form. If the route has many turns of the same type, instead of providing a list such as

> *Turn right on Monticello Road. Turn right on Hudson St. Turn right on Silverado Trail.*

our system groups them as follows:

> *The following three turns are right turns: First, onto Monticello Road; second, onto Hudson St.; finally, onto Silverado Trail.*

More interesting is the inclusion of supplemental information. We derived our model from the one proposed in [4]. They separate essential from supplemental information, and use a notion of salience (importance) derived from their corpus of subway directions to decide which supplemental features to include, and in which linguistic form. In fact, linguistic form, e.g. using a subordinate or a main clause, can impart prominence to a certain feature. In our system, there are three types of supplemental information: 1) transition distances, 2) landmarks, and 3) signposts. A transition is the roadway traveled during one segment of the route. Landmarks are grouped as either large or small, e.g. rivers and railroads are large, and creeks and sloughs are small. Large landmarks are more important than small ones. Signposts are special types of intersections used for improving wayfinding.

In our system, there are four values of salience (0-3), that determine which supplemental information will be in-

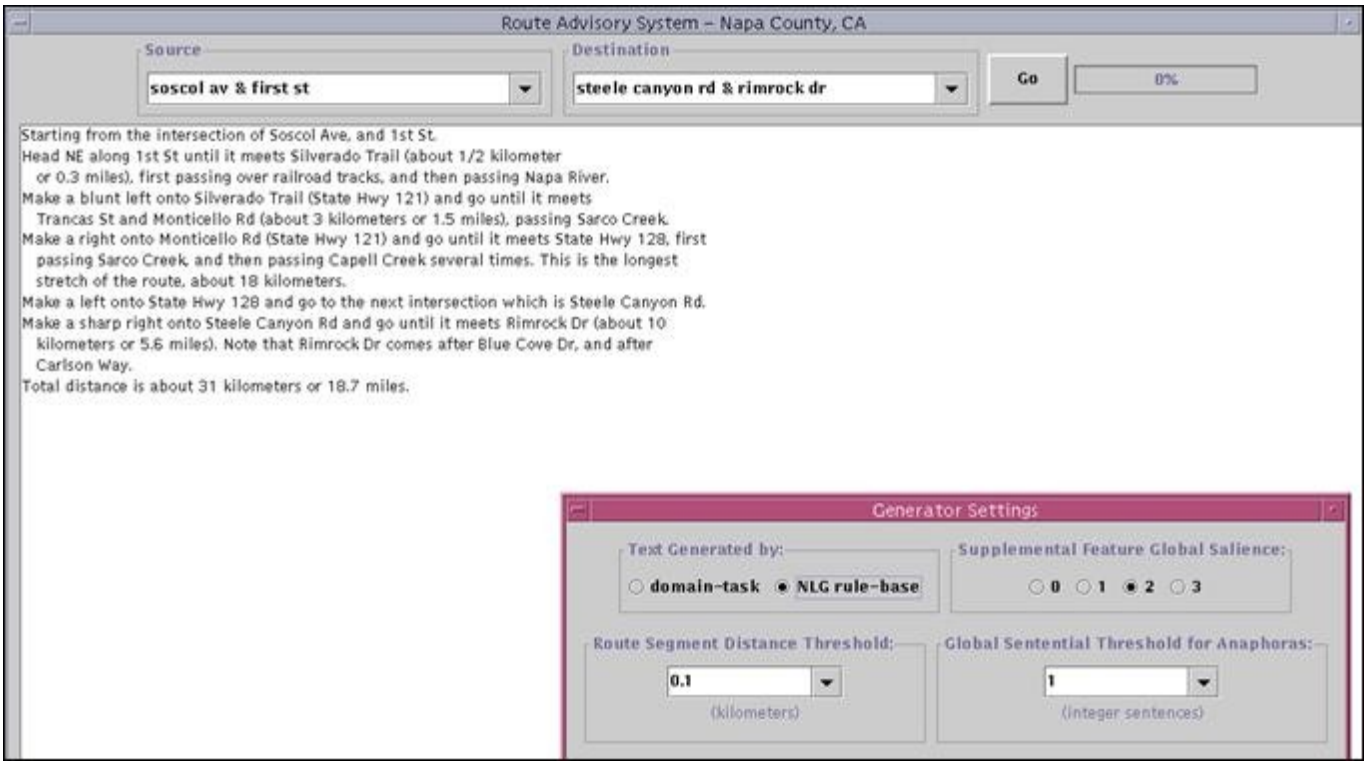

Figure 3: Enhanced route directions

cluded, and in which form. At salience level 2, the level we used in our experiment, we include the transition distance, embedded in the main clause; we mention large landmarks, and if those don't exist, small landmarks, in subordinate clauses (see *passing Sarco Creek* in the example below, which is an excerpt from Figure 3); and signposts, if they exist, in a separate main clause. We also perform referential expression generation, namely, we choose which expression to use to refer to entities in the domain – note that *Sarco Creek* is designated as *the creek* in the second sentence:

> *Make a left onto Silverado Trail and go until it meets Trancas St and Monticello Rd (about 3 kilometers of 1.5 miles), passing Sarco Creek.*
> *Make a right onto Redwood Rd and go to the next intersection which is State Highway 29, again passing the creek.*

3.2.1. The domain-feedback generator interface. The A* algorithm is not interfaced directly to our Natural Language generator, EXEMPLARS [15]. An intermediate module, the dfg-interface, performs tasks such as computing components of aggregated sequences of turns.

The dfg-interface converts the chain of nodes and edges in the A* output into a new, non-isomorphic chain of nodes and transitions. This interface uses transition and node classes, designed specifically to make the information more relevant to the linguistic contexts that will ultimately be generated.

Another dfg-interface task involves aggregation and the construction of aggregated structures. Though aggregation, in the general case, is a very difficult problem, the task in this system is straightforward. The type of aggregation that is realized does not appear in the literature but comes closest to the definition of predicate-based, syntactic aggregation as described in [10]. An aggregation task is as follows: The dfg-interface, as it traverses the A* chain, searches for two or more consecutive transition nodes that contain identical turns. If such a series is found, the creation of an external reference list marks these transition nodes. The list is then inserted into the final dfg-interface output chain as an alternate path. Ultimately, the text planner (in the feedback generator) has the choice of realizing these reference lists as aggregated linguistic structures. For low levels of salience, i.e. 0 and 1, the planner would use the unaggregated path of the dfg-interface. Aggregated structures are the only linguistic structures that require special preparation in advance of the text planner. All other linguistic structures are determined during text planning.

3.2.2. The feedback generator The feedback generator uses EXEMPLARS, a NL generator distributed by CoGenTex, Inc., and written in Java [15]. It provides a framework for developing a text planning and sentence planning NLG rule base. The rules (called *exemplars*) are meant to cap-

```
exemplar AggTurns(AggCluster ac,
         Vector params,
         TransitionGroup tg,
         int salience)
{
 void apply();
   describeTurn();
   describeSuppFeatures();
   describeTerminus();}

 void  describeTurn(){
 }

 void describeSuppFeatures{
 //inherited method}

 void describeTerminus(){
 //inherited method}
}
```

Figure 4: An exemplar from our rule base

ture an exemplary way of achieving a communicative goal in a given context. The NLG system developer will write the specific exemplars needed by the specific application.

The EXEMPLARS rule base is a schema-based, hybrid NLG system. A schema is a pattern used by the text planner that specifies how a particular text sequence should be constructed [11]. The pattern can be composed of other schemas or base messages, along with the relationships between these constituents. A hybrid NLG system mixes various NLG techniques in order to achieve a practical, applied generation system. Although hybrid systems are a departure from traditional NLG tools, they have several advantages including efficiency, simplified architectures, and deterministic output [15].

EXEMPLARS uses classes of exemplars to classify and define schemas. These classes are object-oriented and internally use methods and class data for defining the explicit structuring pattern for the text sequence they generate. The rule base is, therefore, a singular inheritance-based tree hierarchy of communicative actions, which is used by the text planner. Figure 4 shows one of the exemplars in our rule base, in a very simplified form. This is the most generic exemplars used to generate an aggregate description, other exemplars in our rule base specialize it.

The EXEMPLARS text planner takes as input an exemplar requested by the feedback generator, which is able to determine the most general exemplar for a specific context. The text planner then searches the rule base, starting with the requested exemplar as the most general goal, and evaluates applicable conditions in order to determine if any descendents of the requested exemplar more strongly satisfy the goal. No action is taken until the most specific rule is found.

The rule base is responsible for incorporating the following linguistic components into the planned text:

- Aggregated structures
- Signposts / Longest segment clauses
- Landmarks
- Referring expressions

As previously mentioned, the dfg interface computes aggregated structures, data structures that represent two or more consecutive nodes along a route that contain identical turns. Upon encountering an aggregated structure, the text planner will apply syntactic aggregation [10] to realize the structure into the following form, using the exemplar shown in Figure 4, or one of its descendants:

```
The following <n> turns are
<turntype>:
    First, onto <roadway name>
    until <terminus name>
    + <dependent clause>
    Second, onto <roadway name>
     until <terminus name>
    + <dependent clause>
    ...
    Finally, onto <roadway name>
    until <terminus name>
    + < dependent clause >
```

The purpose of linguistic aggregation is to eliminate the redundant utterances of consecutive, identical turns and to group the aggregated route segments under a common turntype. Cognitively, this allows the reader to treat this part of the route as a single, distinctive chunk.

Landmarks, which are attributes of transitions, are categorized into large and small with large landmarks having higher salience. For example, creeks and sloughs (small) both possess low but equal salience. All other landmarks are large. If a route segment contains landmarks, the EXEMPLARS rule base will incorporate them based on their own salience and on the salience setting SAL. For example, for SAL=0 no landmarks are included; for SAL=1, large landmarks are included, if they exist; for SAL=2, large landmarks are included, if they exist, but if they don't, small landmarks are included; for SAL=3, all landmarks are included. Landmarks are included via subordinate clauses, e.g. *passing Sarco Creek*.

A signpost is a separate main clause in the text that notifies the reader that the terminus for the current segment is soon approaching. An example is shown in lines 11-12 of the text in Figure 3: *Note that Rimrock Dr comes after Blue Cove Dr, and after Carlson Way*. Signposts are included for SAL values of 2 and 3. Signposts are realized

only for the last route segment within a given route, therefore, the terminus for the segment is the final destination. The rule base determines if either or both of the two intersections immediately preceding the terminus are within some distance threshold (one mile in the current implementation) from the terminus.

For the longest segment within the route, the rule base will insert a main clause indicating the length, as shown in lines 7-8 of Figure 3. This clause is intended to allow the reader to approximate the distance to the terminus along a lengthy segment and, thereby, lessen the burden on the reader of having to visually scan for the terminus on the map. The inclusion of signposts and length of long segments is based on the (admittedly informal) observation that on a busy urban route the driver needs to prepare for a turn in advance of the actual road on which s/he has to turn, e.g. by changing lanes. If the driver is not aware of the turn coming up, s/he may actually miss it.

Finally, entities in the domain need to be referred to in some way. Referential expression generation is another important task in an NLG system [11]. Again, it is a rather difficult task in general, since there are many possible expressions to choose from: pronouns, proper nouns, definite referring expressions (those that start with the definite article *the*), and others. The choice of referring expression depends on the closeness of the antecedent, on whether there are distractors, i.e. other entities that can be referred to in the same way, and on other factors. For example, if there are one boy and one girl in our domain, we can introduce them with their own first names, and then refer to them with pronouns, e.g.:
Thomas and Jenna are friends. He is 14 and she is 13.
However, if we are talking about two girls, obviously using the pronoun *she* won't distinguish between the two, and we need to use a definite description:
Emory and Jenna are friends. The former is 14 and the latter is 13.

We did not use pronominal referring expressions in this domain, since the antecedents are, generally, too distant. Using pronouns as referring expressions to distant antecedents could result in ambiguous expressions. The usage of definite references is more appropriate. For these route texts, a variation of the GNOME algorithm [7] was used. This algorithm uses a Global Discourse Model (GDM) that contains the entities that can be referred to with a definite referring expression (it also uses a Local Discourse Model, which we did not use). The GDM is updated (i.e., new entities are added in and old entities are deleted) based on the grain size of the update unit. The user interface, through the sentential threshold for anaphoras component (see the secondary window in Figures 2 and 3) allows the grain size of the update to be user defined. For our experiment, we set it to one sentence. Thus, while processing the current sentence, the GDM will contain all the entities mentioned in the previous sentence. We follow this simple algorithm to choose the appropriate referring expression:

```
IF first mention of entity
THEN use proper noun
ELSE IF antecedent in GDM
     THEN use definite referring
      expression
     ELSE use proper noun
```

In addition, to avoid ambiguity, the proper noun will be used if the referring expression is contained in a compound clause that in turn contains other landmarks of the same type. Refer to Figure 3, lines 5-7: *Sarco Creek* in line 7 could be referred to as *the creek*, since it was just mentioned in line 5, i.e. the previous sentence. However, since line 7 also mentions *Capell Creek*, the expression *the creek* would be ambiguous.

4. Experiment and results

The experiment was single-factor and involved two independent sample groups. Group A was the control group, and group B the effect group, with each group consisting of 15 subjects all of whom were graduate/undergraduate college students. A group of three routes, plus a sample route, were selected in advance. The three routes consisted of an urban route, an inter-urban route, and an urban-rural route and were used for all subjects. For group A (control), the route text was generated in essential form. In contrast, directions for group B were generated through the domain-interface-feedback generator pipeline.

For each route, the subject was required to outline the route on a map with pen in hand while reading the text (see Figure 5). This phase tested the ability of the subject to wayfind. Elapsed time, with no upper limit imposed, was recorded when the subject found the destination.

Upon completing the wayfinding phase, the subject was given three questions on the route just completed, to be answered without the use of the route text. One question asked the subject to recall the route in as much detail as possible. Measuring subject recollection of the overall route structure was the purpose of this question. We evaluated this questions via the measures of precision and recall. Precision is the ratio of correct answers C to the total number of answers T given by the subject; recall is the ratio of correct answers C to the number of possible correct answers P. For example, if the route has three right turns and two left turns, and the subject remembers one right turn and three left turns, C=3, T=4, P=5, precision=3/4=0.75, and recall=3/5=0.6.

Figure 5: Tracing the route on the map

The other two questions were shorter and more direct, asking the subject to recall specific supplemental features, such as railroads along the route, and the total route distance. The purpose of these was to see if the conceptual salience of supplemental features was related to the enclosing linguistic form. These two questions were graded on a 0 to 3 scale. The two question total was scaled and normalized – this is the number reported in Table 1 under *Two question score*.

Finally, the number of incorrect turns was computed by inspecting the maps the subjects had used, since they were required not to lift the pen from the paper (see Figure 5).

Table 1 reports the means for the various measures, for the two groups. Boldface is used for measures that significantly differ between the two groups.

	Group A	**Group B**
Total Time	22'2"	22'2"
Route precision	.95	.91
Route recall	.39	.45
Two Question score	.79	.77
Wrong turns	4.4	**2.7**

Table 1. Experimental Results

The results show that supplemental information helps to correctly navigate the route. There was a marked difference in the number of incorrect turns, 4.4 for group A and 2.7 for group B. The difference is statistically marginally significant (Mann-Whitney test, U = 94, p = 0.09). There were no other measures that significantly differed between the two groups, Group B was slightly better on route recall, group A on route precision and on answering the other two questions, but none of these differences is statistically significant, not even marginally. A curious effect is that the average time on task is identical across the two groups.

5. Discussion and Conclusions

Our system demonstrates that enhanced instructions improve wayfinding performance of simulated drivers. A system like ours could be straightforwardly integrated into driving direction services that are accessed before driving starts such as MapQuest, or by a passenger in the car who has access to e.g. a PDA.

Generating directions in real time for the driver and conveying them via speech, possibly coupled with a display showing the map, is obviously more complex. We still believe that enhanced directions would be beneficial; however, many issues arise, such as timing and what is the right amount of information to provide so as not to cognitively overload the driver. Clearly, conveying the amount of information provided in either Figure 2 or 3 orally, in one shot, would not be effective - the driver would likely forget most of it right away. Directions would have to be conveyed in smaller chunks, at the appropriate time, which also depends on what the driver is doing. For example, the interface should have the ability to pause and allow the driver to focus on driving for a complex maneuver, by suggesting "Turn left when safe" instead of "Turn left now" (if there are choices of when to turn left). An intriguing possibility is to integrate a cognitive model of driving such as that described in [13], in order to predict the appropriate timing for providing instructions.

It is possible that the enhanced directions would provide too much information in a real driving scenario in which the driver him/herself has to remember the directions. Because the level of additional information provided is a settable parameter in our system, this is testable. We could run an experiment in which we provide spoken directions in a piecemeal fashion, and we compare saliency levels SAL=1 and SAL=2 (SAL=2 was the setting in the experiment described in this paper). This experimental setting would be closer to a real driving situation even if obviously still missing some crucial elements. A more realistic experiment could be run as a collaboration with a research group that has a driving simulator at their disposal.

Acnowledgments

This work is supported by grants N00014-99-1-0930 and N00014-00-1-0640 from the Office of Naval Research. We are grateful to CoGenTex Inc. for making EXEMPLARS available to us.

References

[1] N. O. Bernsen and L. Dybkjær. A multimodal virtual co-driver's problems with the driver. In *Multi-Modal Dialogue in Mobile Environments (IDS-02)*, Kloster Irsee, Germany, June 2002.

[2] D. Buehler, S. Vignier, P. Heisterkamp, and W. Minke. Safety and operating issues for mobile human-machine interfaces. In *Proceedings IUI03*, Miami, FL 2003.

[3] P. Coletti, L. Cristoforetti, M. Matassoni, M. Omologo, P. Svaizer, P. Geutner, and F. Steffens. A speech driven in-car assistance system. In *Proceedings of IEEE Intelligent Vehicles (IV 2003)*, Columbus, OH, June 2003.

[4] L. Fraczak, G. Lapalme, and M. Zock. Automatic generation of subway directions: salience gradation as a factor for determining message and form. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 58–67, Niagara-on-the-Lake, Canada, 1998.

[5] P. Geutner, F. Steffens, and D. Manstetten. Design of the VICO spoken dialogue system: Evaluation of user expectations by Wizard-of-Oz experiments: A speech driven in-car assistance system. In *Proceedings of LREC02*, Canary Islands, Spain, June 2002.

[6] B. E. John, D. Salvucci, P. Centgraf, and K. Prevas. Integrating models and tools in the context of driving and in-vehicle devices. In *Proceedings of the Sixth International Conference on Cognitive Modeling*, 2004.

[7] R. Kibble and R. Power. Nominal generation in GNOME and ICONOCLAST. Technical report, Information Technology Research Institute, University of Brighton, Brighton, UK, 2000.

[8] W. Klein. Local deixis in route directions. In J. Jarvella and W. Klein, editors, *Speech, Place and Action: Studies in Deictic and Related Topics*, pages 161–182. J. Wiley and Sons, Ltd., 1982.

[9] T. Pattabhiraman and N. Cercone. Selection: Salience, relevance and the coupling between domain-level tasks and text planning. In *Proceedings of the Fifth International Workshop on Natural Language Generation, Pittsburgh, Pennsylvania*, pages 79–86, 1990.

[10] M. Reape and C. Mellish. Just what *is* aggregation anyway? In *Proceedings of the European Workshop on Natural Language Generation*, Toulouse, France, 1998.

[11] E. Reiter and R. Dale. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press, 2000.

[12] S. J. Russell and P. Norvig. *Artificial Intelligence — A Modern Approach*. Prentice Hall, 1995.

[13] D. Salvucci. A multitasking general executive for compound continuous tasks. *Cognitive Science*, 2005. (in press).

[14] M. J. Trolio. A natural language approach to generating route directions: An empirical analysis. Master's thesis, University of Illinois (Chicago), 2000.

[15] M. White and T. Caldwell. Exemplars: A practical, extensible framework for dynamic text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 266–275, Niagara-on-the-Lake, Canada, 1998.

[16] D. Wunderlich and R. Reinelt. How to get there from here. In J. Jarvella and W. Klein, editors, *Speech, Place and Action: Studies in Deictic and Related Topics*, pages 183–202. J. Wiley and Sons, Ltd., 1982.