On-line user modelling in a mobile spoken dialogue system

Niels Ole Bernsen

Natural Interactive Systems Laboratory
University of Southern Denmark
nob@nis.sdu.dk

Abstract

The paper presents research on user modelling for an in-car spoken dialogue system, including the implementation of a generic user modelling module applied to the modelling of drivers' task objectives.

1. Introduction

In general, the concept of user modelling addresses issues of understanding users in order to make a system useful and make user-system interaction user friendly and universal. We may distinguish between at least the following concepts of user modelling.

In *design-time adaptation to users*, we employ various methods to anticipate users' needs, goals, interactive behaviour, etc., in order to proactively fit the system to its future users. The system may be programmed at design time to dynamically respond in different ways depending on the user's behaviour, but it does not offer the user any other ways of modifying the system's behaviour. Secondly, the system may be *customisable* by users who can modify the system's behaviour in various ways, such as functionally or with respect to its style of interaction, by selecting among customisation options. Thirdly, in *user model-based adaptation*, the system may itself observe the user's behaviour and adaptively modify its interactive behaviour on-line as a function of the data gathered. Thus, user model-based adaptation is: on-line, made by system, uses collected and stored user information. Two forms may be distinguished: (a) *historical adaptation:* uses implemented user model for each user and (b) *instant adaptation:* uses short-term buffered user information.

In the prototypical form of user model-based adaptation, *historical adaptation*, the system collects and stores information on the individual user's behaviour for later use on-line. In *instant adaptation*, the point is to apply the information gathered as soon as possible, i.e. when the system knows that the user's behaviour has a particular property which requires adaptation. Systems which do instant adaptation may treat the user anonymously. Hybrids between historical adaptation and instant adaptation are possible.

The proliferation of mobile systems in the future offers new challenges for user model-based adaptation technologies. Mobile systems will communicate with their users in an increasing number of modalities [1], such as input/output speech, in addition to the customary GUI input/output modalities. Mobile systems may offer location-based and situation-based information. Coupled with the user-awareness provided by user model-based adaptation, opportunities are huge indeed [2].

This paper describes work on user model-based adaptation in a large mobile system called VICO or Virtual Intelligent CO-driver. Section 2 describes the system's functionality. Section 3 presents findings on generic system tasks for user modelling, user identification, user modelling information, and criteria for selecting the information to model. Section 4 describes the VICO user modelling module. Section 5 describes next steps in our work.

2. The VICO system

Spoken language dialogue systems (SLDSs) are migrating into mobile environments, such as the car. VICO is such a system which has been developed in the European HLT VICO project 2001–2003, addressing some next-step challenges in the context of supporting car drivers whilst driving [3]. In brief, the challenges include: (1) ease of use by a large and heterogeneous user population; (2) processing of fully spontaneous spoken input, English, German, and Italian; (3) multiple-task assistance: with navigation to addresses and points of interest in Italy, hotel reservation, restaurant reservation, and VICO system information; (4) confidence-score adaptive dialogue using speech recogniser and natural language processing confidence scores; (5) GPS-based location-awareness; (6) multimodal interaction through push-to-talk button and spoken input, and spoken and display (text) output; and (7) integration of adaptive user models built on-line.

3. General findings

At the start of VICO, we analysed the scope of user modelling for in-car use, taking into account the particular tasks of VICO (Sect. 2).

3.1. Generic system tasks

VICO's *generic* UM-related tasks are [4]: (1) identify the present driver; (2) retrieve the present driver's user model; (3) optionally: create a new user model UM(Dx) for a new driver, Dx; (4) make appropriate on-line use of the present driver's user model during the driver's dialogue with VICO; (5) collect new information on the driver during the driver's dialogue with VICO; (6) update the present driver's user model with the new information gathered; and (7) store the user model whenever it has been updated with new information. This generic task list assumes the *historical adaptation* approach (Sect. 1). *Instant adaptation* requires that the system solve fewer generic tasks, i.e.: (1) collect information on the present driver during the driver's dialogue with VICO; (2) store the collected information temporarily; and (3) make appropriate on-line use of the information.

As will be demonstrated below, both historical adaptation and generic adaptation are relevant to VICO's potential user modelling tasks and, probably, to user modelling in mobile environments more generally.

3.2. Driver identification

VICO must determine the car's current driver since cars often have several different drivers. Driver identification must be made with *near-certainty*. If it is uncertain that VICO has correctly identified the driver, misidentification will happen too often. In such cases, the driver is likely to be "mistreated" because VICO will adapt to the driver based on a wrong user model. Similarly, the modelled behaviour of the misidentified driver will tend to fudge up the misallocated user model with misleading information. Since the driver's user model cannot be invoked before identification, VICO must identify the driver *up front*, i.e. as soon as that driver starts the dialogue. Later identification means less support for the driver, and the updated user model runs the risk of having missed to collect important information on the driver's behaviour.

In SLDSs, driver identification design is a non-trivial problem. We have considered (a) voice identification, (b) a driver's code, however input to VICO, (c) driver's spelled first name, and (d) combinations of (a) through (c). Other solutions are possible, such as car key radio signals, several (other) biometrical solutions, etc. Among the above, we prefer to use voice identification-cum-spelling of their first names by first-time users. This combines the unobtrusive elegance of voice identification with (non-coded) first-name feedback and avoids having to remember and use a code or spell one's name on each occasion of use. This option also allows passengers to speak to the system without being registered in the UM database – passengers just have to avoid giving the system their first names.

3.3. Which type(s) of information to model?

Based on analysis of some 25-30 candidate kinds of information about driver behaviour which VICO might collect and use adaptively, we distinguish between: (T1) information on the driver's *task objectives* due to task goals, preferences, habits, etc.; (T2) information on the driver's *communication* with VICO; and (T3) information on the driver's *experience* of various kinds. This information typology helps generate a structured space of candidates for observation-based adaptive user modelling, each generic type of information subsuming several specific information types, such as the driver's: preferences for hotels, restaurants, points of interest, petrol brands, or address locations (T1); native language, communication difficulties due to strong accent or dialect, speech disorders, extreme talkativeness, elaborate politeness, frequent cross-talk with passengers, unusual speaking style, etc. (T2); and experience in using VICO itself (T3).

We then identified a set of criteria which should be satisfied by the driver information to be modelled. These are: (C1) *universality:* unless other factors advocate in their favour, the chosen information should be top quality in terms of usefulness to all or most drivers; (C2) *quality:* the information should provide genuine driver adaptivity without significant drawbacks; (C3) *feasibility:* the functionality should be implementable without extreme or unpredictable effort, the latter being due to, e.g., a needed research breakthrough; and (C4) *verifiability:* the functionality must be based on clearly verifiable information about the driver.

An example of T1 in the typology is: store the driver's past hotel preferences, such as number of stars, price, location, hotel chain, etc. Even if not presently told about them by the driver, VICO could offer to use those constraints as selection criteria when looking for a suitable hotel. Let us evaluate the hotel preferences functionality using the selection criteria C1 through C4. C4 is met because the driver's hotel preferences become apparent during dialogue with VICO; they do not appear to have any significant drawbacks (C2). The functionality can be implemented without extreme or unpredictable effort (C3). Whether C1 is met depends on, at least, (i) how many users of VICO will need to book hotels, (ii) how many users will do so *en route*, and (iii) how many users have systematic hotel preferences. We do not know the answers at this point.

It is harder to identify suitable T2 information candidates. An example is a system which (instantly) adapts its dialogue to drivers having a strong dialect or accent. A major problem is that any solution may be at risk as long as we do not have efficient ways of diagnosing different *possible causes* of recognition problems. Low confidence scores, many out-of-vocabulary words, or multiple error corrections, for instance, cannot tell if the cause of repeated recognition problems is a strong dialect or accent or something different, such as a driver who regularly talks to passengers during dialogue (C3, C4). T2 solutions might well satisfy C2. And even if not benefiting all or most drivers, they might benefit large fractions of those drivers who have great difficulty using spoken language dialogue systems at all, illustrating the exception clause in C1.

T3 information includes at least one obvious candidate, i.e. the driver's experience with VICO itself. The idea is to offer up-front information on VICO's tasks and how to operate VICO to all new drivers independently of whether or not a new driver asks for it. Provision of this information would seem to rather easily meet C1 through C4. Implementation will be simple because VICO only needs to determine if the current driver is new to the system. It needs not store a record of the driver's behaviour nor does it need UM update algorithms (C3). Contrary to "standard" instant adaptation (Sect. 1), however, the system must be able to uniquely identify the driver.

4. Application description

Guided by the above analysis, we have implemented a general-purpose UM module and applied it to facilitate drivers' hotel selection dialogue through knowledge of their hotel preferences in the past.

4.1. Location-dependent and location-independent adaptivity

The following driver hotel preference behaviour feature-value pairs are collected and used by the UM:

- type [VALUE = HOTEL]
- hotel name [VALUE = NAME]
- hotel address [VALUES = ADDRESS ITEMS]
- number of stars [VALUES = 1, 2, 3, 4, 5]
- hotel chain [VALUE = NAME]
- hotel location [VALUES = TOWN, OUTTOWN, COUNTRYSIDE]
- max. prices for single (S)/double (D) rooms [VALUES = S: X and D: Y Euros]

- restaurant in hotel [VALUE = TRUE, FALSE]
- protected parking [VALUE = TRUE, FALSE]

Analysis showed that these attribute-value pairs are of two very different kinds, i.e. the generic hotel properties 1, 4, 5, 6, 7, 8, and 9, and the specific hotel properties 2 and 3. *Generic* hotel properties may belong to any particular hotel. *Specific* hotel properties imply all properties of a particular hotel: once you choose a *particular* hotel, e.g. by its name, you choose all its properties. The UM thus cannot use specific hotel properties to support the driver's hotel selection task independently of where the driver happens to be. Only generic hotel properties can be used for *location-independent* user model-based driver support. Thus, the hotel selection UM must have two distinct adaptive functionalities. The first, *location-independent*, functionality offers hotels having the generic properties of hotels which the driver has preferred in the past. The second, *location-dependent*, functionality offers the specific hotels which were preferred in the past if and only if the driver is in the area in which those hotels are located.

4.2. User model update and use

A crucial design issue is how to *update* the UM with new information. If, e.g., the update algorithm averages over the past, and if there is a long UM record of staying in inexpensive hotels, then the UM may never fully realise that the driver has changed hotel preferences. As they are, our update algorithm for each *generic* hotel property takes the two latest hotel reservations into account through combination and interpolation. As for the driver's previous choices of *specific* named hotel/location pairs, the UM stores and uses all of them, no matter how long ago it was when the driver stayed in a particular hotel.

The context of use of the hotel selection UM is that the driver asks VICO to help book a hotel, possibly adding some selection constraints, saying, e.g., "VICO, please find a three-star hotel." At this point, the hotel preferences UM is being applied.

VICO UM application raises several design issues likely to strongly affect user acceptance. Firstly, it seems clear that the UM should never override the driver's stated hotel selection constraints. Thus, if the three-star constraint in the example above conflicts with the UM, the former should prevail by applying conflict resolution. If the driver's stated selection constraints suffice for uniquely identifying a hotel through querying the hotels database, the UM should not be applied. Secondly, when UM hotel selection constraints are being applied in querying the database, possibly as complements to the driver-provided selection constraints and following conflict resolution, it is important to inform the driver that the returned hotel suggestions are the results of UM application. Adaptation should be made behind the driver's back.

In *location-based* selection support, the UM provides a list of past selected hotel/location pairs. If the driver provides a location corresponding to a location on the list, s/he is offered the corresponding hotel(s). If no location is provided, VICO assumes that the driver wants a local hotel. If the car's current location matches a location in the hotel/location pairs list, the driver is offered the corresponding hotel(s).

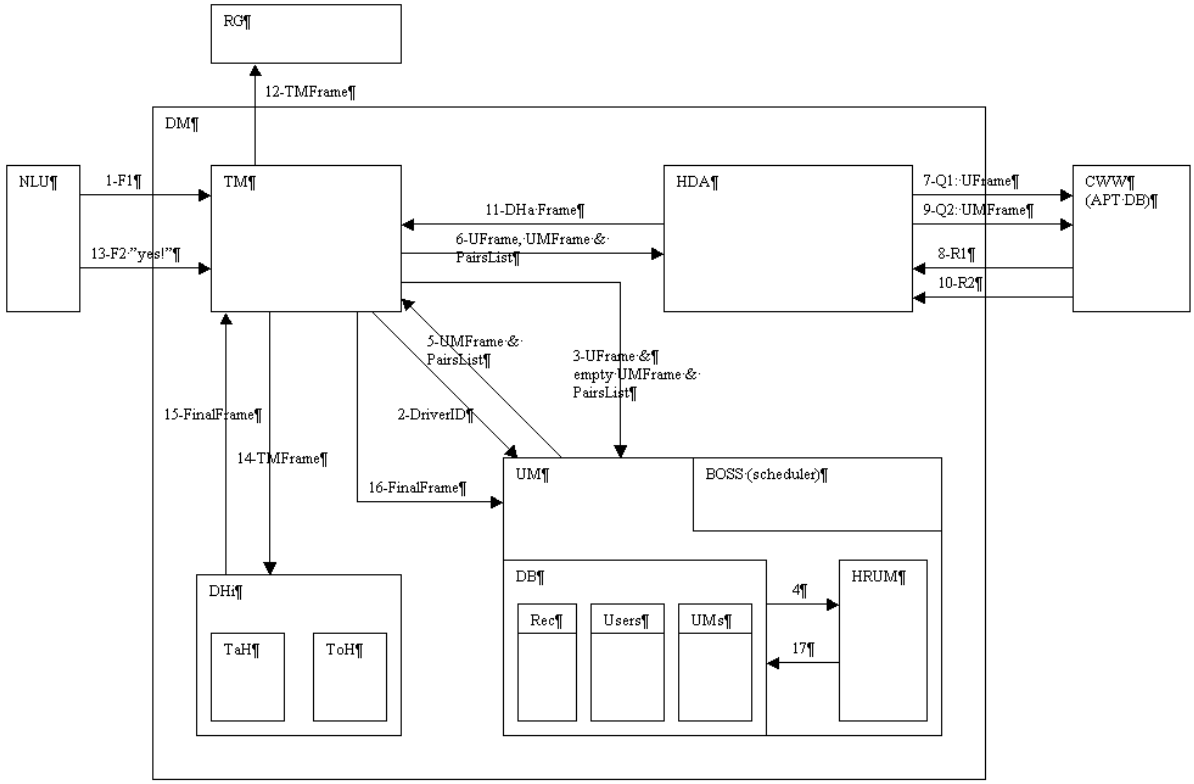

Figure 1: Embedded user model architecture.

4.3. Implementation

Figure 1 shows the generic VICO user modelling module (UM) as embedded in the dialogue manager (DM). External module communication between the DM and three other modules is shown: the DM receives frames from the natural understanding module (NLU), makes database queries over the car-wide web (CWW), and eventually sends output to the user to be turned into a TTS (text-to-speech) string by the response generator (RG). The task manager (TM) controls all intra-DM module communication, using the dialogue histories (DHi) task history (TaH) and topic history (ToH) to store the discourse context. For reasoning about, and making database queries with respect to, each task, the DM is supported by domain agents of which only the hotel task domain agent (HDA) is shown in the figure.

The generic UM includes a communication manager (BOSS), a database (DB), and an extensible set of task-specific user modelling modules of which only the hotel reservation module (HRUM) is shown. The DB includes a list of users known to the system (Users), a series of task-specific records per individual user/task pair (Rec), and a series of models of individual users per task (UMs). UMs are built from Recs by the hotel reservation user module (HRUM) using task-specific update algorithms.

Numbers in Figure 1 shows the progression of information exchange during hotel reservation.:

1. The user begins the hotel reservation dialogue.
2. As soon as the driver is identified, the TM sends the driver's id to the UM.
3. The TM sends the UFrame (what the user said), the UMFrame (empty frame which UM has to fill), and the empty CityHotelPairs list to the UM.
4. The UM retrieves HRUM from the database (UMs table).
5. The UM sends the filled UMFrame and the filled CityHotelPairs list to the TM.
6. The TM sends UFrame, UMFrame, and the CityHotelPairsList to the HDA (DA_hotel).
7. The HDA queries the CWW with the UFrame data, so far *not* using the UM data. The purpose is to find out if the driver's own request could yield a satisfactory result without making use of UM information.
8. The CWW responds to the query (R1).
9. If there is any useful information in the UMFrame and if the previous DB return (R1) includes more than three hotels, then the HDA queries the CWW with the additional UMFrame data.
10. The CWW replies to the query (R2).
11. The HDA sends the DHa frame which contains information about hotels, to the TM.
12. The TM sends the TMFrame to the response generator in order to propose one or several hotels to the driver.

In the ensuing dialogue, the driver and VICO eventually agree on a specific hotel. Then:

13. The NLU inputs a frame which expresses the user's agreement.
14. The TM stores this information in the DHI.
15. The TM retrieves the final frame which contains full information on the selected hotel, from the DHI.
16. The TM sends the final frame to the UM.
17. The UM stores the frame in the DB's Rec table, updates the user module (HRUM), and stores it as well.

What happens to the CityHotelPairs list of previous city/hotel pairs visited by the driver, is that the HDA looks for information on the city in which the user wants to book a hotel. If a city is provided by the driver in the first hotel reservation input, the HDA checks if the city is in the city-hotel pairs list. If it is, the corresponding hotels are offered to the driver, overriding any driver generic hotel selection input. If the city is not in the list, the list is disregarded. If the driver did not provide a city, the HDA must wait until city names are returned from the DB in response to the query made. If any of the returned city names match the cities in the city-hotel pairs list, the corresponding hotels are offered to the driver, again overriding any driver-provided generic hotel selection input. If there is no match, the city-hotel pairs list is disregarded by the HDA.

5. Conclusion and future work

The generic VICO UM module has been integrated into the system's dialogue manager. We have implemented the location-independent part of hotel reservation user modelling. Following implementation of the location-dependent part, our next task is to test with real users in order to evaluate the UM's update algorithms and the principles behind its on-line use. We continue to investigate the most useful and usable driver identification solution.

6. Acknowledgements

VICO is supported by the EU HLT Programme. The support is gratefully acknowledged. I would like to thank Aziz Joumady and Dymitro Kupkin who implemented the VICO UM module, and Laila Dybkjær who specified the hotel reservation task.

7. References

[1] Bernsen, N. O., ``Multimodality in Language and Speech Systems: from Theory to Design Support Tool". Granström, B., House, D., Karlsson, I. (eds.), *Multimodality in Language and Speech Systems*, Kluwer Academic Publishers, Dordrecht, 2002a, 93-148.

[2] M. Bauer, P., Gmytrasiewicz, J., Vassileva, J. (eds), *User Modeling 2001: 8th International Conference*, Springer-Verlag, Berlin Heidelberg, 2001.

[3] Bernsen, N. O., Dybkjær, L., ``A Multimodal Virtual Co-Driver's Problems with the Driver". Dybkjær, L., André, E., Minker, W., Heisterkamp, P. (eds.), *CD-ROM Proceedings of the ISCA Tutorial and Research Workshop on Spoken Dialogue in Mobile Environments*, International Speech Communication Association, Bonn 2002.

[4] Bernsen, N. O., ``Report on User Clusters and Characteristics", *VICO Report D10*, NISLab, 2002b.