BASURDE[LITE], A MACHINE-DRIVEN DIALOGUE SYSTEM FOR ACCESSING RAILWAY TIMETABLE INFORMATION

Roger Trias-Sanz* and José B. Mariño

TALP Research Center, Universitat Politècnica de Catalunya
Jordi Giró 1-3, 08034 Barcelona, Spain
roger.trias.sanz@ieee.org, canton@talp.upc.es

ABSTRACT

We present Basurde[lite], a complete human-machine dialogue system for accessing railway information, based on machine-driven dialogues, a small vocabulary and grammar, and word and phrase spotting based on unrestricted grammars. We give preliminary evaluation results.

Although machine-driven, the system accepts however some flexibility: in some cases the user may give a reply not corresponding to the system’s question, and may use out-of-vocabulary words and expressions. The system mimics the user in his choice of time and date formats and language used for place names.

1. INTRODUCTION

The Basurde[lite] system [1] is a complete human-machine dialogue system which provides railway timetable information in Spanish. It is a spin-off of Basurde [2], an ongoing larger project for a dialogue system which provides the same kind of information as Basurde[lite]. However, while Basurde uses a mixed-initiative strategy and understands a relatively large set of words and expressions, Basurde[lite] uses machine-driven dialogues, and a small vocabulary and grammar recognized by word and phrase spotting.

Such restrictions may make interaction less efficient but at the same time may increase the success rate of the speech recognizer. One goal of this project was to have a reference system to which Basurde could be compared when finished. Short development time was a must. We also wanted to have two dialogue systems which would help us compare machine-driven and mixed-initiative strategies. Finally, we were interested in having, earlier than we would with Basurde, a working system for experimentation and demonstration purposes.

We built several system modules, and their application definition files, and combined them with other existing modules to obtain a complete dialogue system. We describe these three modules in [3]. Here we describe the complete dialogue application.

The language model driving the speech recognizer is a multi-layered description using Hidden Markov Models (HMMs) for demiphones, phonemes and whole words, and finite grammars for the sentence level. It supports multi-lingualism in city names. The language model driving the natural language understanding module (NLU) is given as an unrestricted grammar. Only one part of the language model is active at a given time.

The dialogue model describes machine-driven dialogues with several common phases, including an initial phase, with questions which are required for all queries, a negotiation phase, and a results presentation phase. System output is designed so that the user cannot take the initiative.

The oral response generator is driven by several rules translating from output from the dialogue manager to text in Spanish, including SABLE synthesizer control mark-up. For dates and times, we reproduce whenever possible the same form that the speaker used.

This paper is organized as follows: First we describe the architecture of the Basurde[lite] system. Then we present the language models used for this application. In section 4 we describe the structure of a dialogue. Then we comment on system responses. In section 6 we present the results of a preliminary evaluation. We finish with some conclusions.

2. SYSTEM ARCHITECTURE

The system structure is typical for dialogue systems: an audio module controls the telephone hardware; a speech recognizer turns audio input into sequential text; a natural language understanding module (NLU) turns that text into a structured representation of meaning; a dialogue manager decides the replies to user utterances, and takes actions when appropriate; an oral response generator (ORG) converts these replies from a representation of meaning into text; a speech synthesizer turns this text into audio samples which are fed back into the audio driver (figure 1). A com-
munications module centralizes inter-module communication. The system can be spread across several machines.

![System Architecture Diagram](image)

**Fig. 1.** Overall system architecture. Dashed blocks were partially reused from previous projects. Only data (not control) links are shown.

Of these modules, only the NLU module, the dialogue manager, and the ORG module were developed as part of the Basurde[lite]. The other modules were taken from what was being developed for Basurde. VoiceXML could not be used because no recognizer and synthesizer was available for the application languages.

### 3. LANGUAGE MODEL

The language model tells the system about what user input it should expect, and how to interpret it. We divided it into a model for the speech recognizer (3.1, 3.2, 3.3) and one for the NLU module (3.4).

The language model for the speech recognizer is set up as a multi-layer structure, containing both phonetic data (already available at the laboratory) and syntactic information.

#### 3.1. Phonetic elements

In the lowest level, we provide phonetic descriptions (as HMMs) of all significant Spanish and Catalan demi-phones [4] and Spanish phonemes to a multi-lingual speech recognizer [5]. We also give some HMMs for whole words (for “yes”, “no” and the ten digits, in Spanish), which are short and difficult to recognize, yet semantically important.

We wanted the system to recognize place names said, not only in Spanish, but also in the languages used in each place: Basque, Catalan and Galician; the system also accepts a few locations outside Spain.

#### 3.2. Intermediate levels

Higher in the model structure we have “trash” models for capturing out-of-vocabulary words. These are 3-grams of Spanish phonemes. One trash model captures unexpected isolated words: For example, having “I’d like to leave from [trash] city-name” in the language model would make the recognizer accept “I’d like to leave from, err, Barcelona”. Another model captures whole unrecognized sentences, so the recognizer can return that when confronted with an out-of-grammar sentence.

There is a dictionary that maps from words to sequences of phonetic units (vocabulary size is 482 words). Place names in different languages are stored as different words.

#### 3.3. Top-level grammars

At the top level there are multiple grammars which describe all valid input sentences. Each grammar describes a class of sentences, for example, replies to yes/no questions, dates, or times. The dialogue manager module, which knows what kind of user input can be expected at each turn, instructs the speech recognizer to activate the appropriate top-level grammar. In this way we reduce the recognition space. Some expressions, such as those signaling communication problems, can be accepted at any time.

These grammars were finite and non-stochastic (there was no information on the probability of encountering each sentence), with words as terminals. Resource limitations prevented us from using more elaborate grammars, which might have given higher recognition rates.

#### 3.4. Language model for the NLU module

The language model driving the natural language understanding module is an unrestricted grammar described as reduction rules. As in the recognizer language model, the NLU model is divided into several sections, only one of which is active at each turn. This makes the parser work faster and simplifies the design of the language model.

For date and time expressions, we detect and record the form used by the speaker (for example, we record whether the user said “Monday, January 25th”, “next Monday”, or “the 25th”). We try to propagate this information to the ORG module whenever possible, so that system replies resemble user input and we do not get confusing dialogues such as

— I’d like to leave at quarter past nine in the evening.
— Did you say leave at twenty-one fifteen?

or, assuming next Monday is September 15th,
— Next Monday, please.
— On September 15th. At what time?

### 4. DIALOGUE STRUCTURE

The dialogue is completely machine-driven: The machine takes the initiative at the beginning of the conversation, and
its output is designed so that the user cannot retake it. Dialogues have four main parts: initial questions, negotiation, giving results, and restart. The dialogue model is given as a tree whose structure can be modified during the course of a dialogue.

4.1. Initial questions

In the initial questions phase, the machine asks about information which is needed for all queries: the departure place, the arrival place, and the departure date. It accepts however other answers, such as an arrival place when the departure place is being asked. Date intervals and day types (workday or holiday) are also allowed. With this information the system can launch a query to the database system.

4.2. Negotiation

If more than one train is found, we enter a negotiation phase to let the user select one train among those found.

If few trains are found (three or less) we give some basic information about all of them and let the user choose by position (“the first one”). If many trains are found, we let the user specify a departure or arrival time (or time interval), a train type (night, high-speed, . . . ) or an on-board service (restaurant, telephone, . . . ), and restrict the result set to those trains most closely matching the user’s requests.

4.3. Results presentation

When the user has finally given enough details to select only one train in the result set, the system gives basic information for it. Only if the user asks for it does the system give all available information for that train. This phase was usually too long; we need to find a better way of knowing what the user is really interested on.

4.4. Restart

Finally, we ask the user if he wants to make another query for the return trip, for the same route on another date, or for a different trip. In these cases, the dialogue restarts at a suitable point in the initial questions phase.

5. ORAL RESPONSE GENERATION

While the oral response generation module was the easiest to program, and its application-definition file the simplest of them all, it must be stressed that a careful design of system output is crucial in obtaining good performances.

Users adapt the length and complexity of their responses to that of the system’s questions [6], and we used this both to reduce the complexity of user replies and to restrict user vocabulary. For example, questions are short and formed so that users answer in a certain specific way, yes/no questions are clearly marked as such, and in choice questions we give a list of valid response words in the hope that users will use the same words in their replies.

Finally, to avoid confusing the user and to give the system a better appearance, we try to say dates, times and places in the same way that the user did: for places, we try to use the same language; for dates and times, the same form (see 3.4). However, when we give the final train information, we always use full dates and times.

6. EVALUATION AND RESULTS

We evaluated the system performance by having colleagues from the laboratory (not directly involved in this project) call the system and try to solve some simulated scenarios. In all, 27 calls were analyzed.

We calculated turn success rates for the speech recognizer [SR]1, natural language understanding module [NLU], dialogue manager [DM] and oral response generator [ORG], as well as a “full chain success rate” [full] (turns in which the system’s oral response to the user utterance is correct). For each call we get three transaction success rates: whether users get the requested information [t. real], think they get it [t. user], or think they get a useful part of it [t. part].

For each call we asked the user to grade the following statements (adapted from [7]) between 0 (I disagree) to 5 (I agree): it was easy to get the information I wanted [info], the system understood what I said [s. SR], I found it easy to understand what the system said [s. TTS], I knew what I could say or do at each point in the dialogue [guided], the system worked the way I expected it to [expect], and I would like to use this system regularly [use].

Figure 2 shows these values. Error bars show 95% confidence intervals, rather large because of the small number of calls we were able to make.

By splitting tests in different ways we obtained the following results (the complete results can be found at [1]):

As users get more experienced with the system (each tester made 3 calls), transaction success rates and subjective grades increase. There is a 100% increase in the real transaction success rate from the first call to the third call. Improvements in subjective grades are less impressive. On the other hand, turn success rates do not show any improvement, so users may actually be learning to guide the system to their goals, rather than to speak in a way the system understands better.

Easy scenarios get better subjective grades than complex ones, but transaction success rates do not change significantly. This may be related to the higher degree of user fatigue in difficult scenarios.

1Codes in brackets are used in the results graph.
As expected, ambient noise makes conversations more difficult. Tests made in a low ambient noise situation get up to 1.5-point increases in some subjective grades compared to noisy situations; transaction success rates get up to 30% increases.

The system performs satisfactorily for simple tasks: recognition rates are high and processing delay (since the user stops talking until the system starts responding, including the end-of-speech detection delay) is typically 2 seconds on a 550 MHz Intel Pentium II computer.

However, the system was less comfortable to use for more complex tasks (e.g., those involving multiple trips). We believe that machine-driven, small-vocabulary systems are not flexible enough to handle complex tasks, and that mixed-initiative systems should be used in that case.

We could very easily train the user on how to use the system by means of carefully designed system prompts. We saw that precise wording was indeed crucial in determining the kind of user answers we would get [8]. On the other hand, callers largely ignored explicit aids and help mechanisms. We verified that making dialogues fast-paced and keeping system prompts as short as possible improves system performance [8].

We have presented a machine-driven dialogue system that lets people obtain railway timetable information through telephone. We use a small vocabulary and grammar. The speech recognizer uses HMMs of phonemes, demiphone and whole words, and finite grammars of words, and, for a subset of its vocabulary, can operate multi-lingually. The system tries to follow the way the user gives dates, times and places.

Preliminary evaluations show good performances, and confirm the crucial role that system prompts have in controlling the user’s reply and, more generally, in the overall system performance. Future work should include more extensive evaluation, allowing user barge-in, and using better language models.

8. ACKNOWLEDGEMENTS

This work was supported by a grant from the Spanish government (TIC98-0423-C06-01), and executed from February to July 2001.

The authors would like to thank Antonio Bonafonte at the TALP Research Center at UPC for his help in the development of Basurde[lite].

9. REFERENCES