Limited Enquiry Negotiation Dialogues

Ian Lewin

Netdecisions Ltd
Wellington House, East Road, Cambridge, United Kingdom
ian.lewin@netdecisions.com

Abstract

We define a new dialogue management strategy Limited Enquiry Negotiation Dialogues designed for enabling simple man-machine dialogues in which the parameters (for which the user will supply values) of a query to a database are negotiated. The choice of which query to make next is also not pre-ordained. The strategy is simple and intuitive but permits interestingly complex dialogue behaviour. We propose it as an addition to a dialogue designer’s standard components toolbox along with other well-known ideas such as menu-traversal and slot-filling. We illustrate the strategy by examining how it accounts for interesting but by no means rare data in a Wizard of Oz corpus of business trip planning dialogues. Finally, we discuss some more theoretical issues arising from the model.

1. Introduction

Tremendous interest attaches to the notion of re-usable dialogue structures. They hold out the promise of faster development times, re-use of proven technology and, not least, a greater degree of consistency in user experience. One engineering-oriented approach is to provide a toolbox of components from which a dialogue designer can mix, match and apply to his purposes. Indeed, this is the spirit of the successful widget approach in graphical user interface design. As speech recognizers and synthesizers become mature software products, the same developments can be seen in speech interfaces, witness VoiceXML™[1], SpeechObjects™[2] and DialogModules™[3]. Simple examples of speech widgets include yes-no question dialogues, menu-traversal, list traversal and slot-filling dialogues. In this paper, we will be proposing a new dialogue type based on the concept of query negotiation. The concept can be introduced by briefly reviewing the concept of a slot-filling dialogue.

Slot filling, as realized in the Philips’ train timetable system [4] (the general idea does not originate there; but it is particularly clearly articulated) includes the notion of an action, e.g. a database query, which the system is aiming to carry out. Before it can do so, a number of parameters are required to be supplied with values. Therefore, the system issues a prompt (which needs careful design) and fills in the values of as many parameters as it can from the response. Ideally, all required values will be supplied immediately and then the action can be executed. Otherwise, the system considers the slots not filled and issues another suitable prompt. The cycle continues until all the slots are filled and the intended action is executed. A mechanism – even if it is only “Quit” – may also be provided for escaping the cycle. VoiceXML also includes a somewhat similar construct called a form. The slot-filling idea is a very simple one and it provides a degree of flexibility for the user in filling in the required parameter values. Most importantly, it will often be very easy for a dialogue designer to understand how the intended outcome, a parameterized action, should map onto the underlying non-linguistic capabilities of the system (e.g. carrying out a database query). Nevertheless, the model is of course also very limiting. The user must still the supply values for the required parameters at some time. Also, the dialogue is directed towards a single system goal, that of carrying out the intended parameterized action. If the user asks a question, then this must be treated as a sub-dialogue with an envisaged return to the current dialogue; or else as abandoning the current dialogue.

Limited Enquiry Negotiation Dialogue (LEND) is a model with the following properties:

1. Users need not supply any particular subset of the parameters of an action – the ones that will be used are the subject of a negotiation.
2. The system is not indifferent as to the best set of parameters to be used.
3. Users can ask questions at all times. Questions are not treated as sub-dialogue openings, nor is the current dialogue abandoned. It can be returned to at the user’s discretion.

In the rest of the paper, we define the model and illustrate it with two dialogues from a Swedish Wizard of Oz corpus of business trip planning dialogues [5]. The dialogues particularly highlight the different roles of questions in dialogue. In the first example, the final database query is preceded by a sequence of three consecutive questions and then a statement which does not answer any of them. The whole sequence is perfectly natural and each individual member is entirely within central coverage of the domain. A simple negotiation analysis accounts for the data. The second example illustrates how a more standard ‘nested dialogue’ is also accounted for. Finally, we discuss the relation of LEND to other dialogue modeling techniques including those that invoke concepts of intention or game.

LEND is fully implemented in the TrindiKit environment [6].

2. The Model defined

In LEND, a dialogue is modelled around a set of query-types Q, each consisting of an action A and a set of possible parameters P. In our example scenario, there are just two query-types: ROUTE and PRICE. ROUTEs take parameters of destination, travel-mode (train or plane), date, arrival time and departure time. PRICEs take parameters of destination, travel-mode and cost. In SQL, one might formalize the possible ROUTE queries as

\[ \forall X \in \{ \text{destination} = A, \text{date} = B, \text{travel-mode} = C, \text{arrival-time} = D, \text{departure-time} = E \} \]

SELECT output-fields from ROUTE
WHERE X
We suppose that any subset of parameters can be used in a query. We further suppose a valuation function on possible queries that encodes the worth to the system of that query. Figure 1 shows part of the query space. All single parameter queries are in the top row and any combinations with a non-zero worth are also shown. For example, there are only two non-zero two-parameter queries: destination & date and destination & mode. There are five non-zero three-parameter queries of which destination & date & mode has worth 5.

**Figure 1: Partial Query Space (Route Queries)**

```
Dest Mode Date DepT ArrT
1 0 0 0 0
2 0 0 0 0
3 0 0 0 0
4 0 0 0 0
5 0 0 0 0
```

The example is hand-coded and designed to represent the general principles that some parameters are better than others and that more parameters are generally better than fewer. Queries with too many parameters may be non-optimal however. They may lead to no results at all and therefore, in our scenario, provide nothing for the user to choose from.

LEND maintains four runtime structures

1. **Semantic Value**: the semantic value of the last user input - a triple of type (question, assertion), parameter (mode, arrival-time,...), and value (train, 14:35...).
2. **Deal**: an association set of parameters and their statuses each being notRaised, proposed, accepted or rejected.
3. **Context**: an association set of parameters and their values (e.g. mode and train, arrival-time and 14:35...)
4. **Query Stack**: a stack of query-types in which no query-type may appear more than once. The top of the stack is the currently intended query.

The runtime algorithm consists of a cycle over four stages

1. **Selection**: a function from the current Deal and the currently intended query to the next parameter to offer.
2. **Output**: an utterance generator which offers the next parameter and updates Deal with its status (e.g. a notRaised parameter now becomes proposed)
3. **Input**: obtains a Semantic Value from the user.
4. **Interpretation**: a function that updates Deal, Context and Query Stack given the user’s latest Semantic Value.

The selection function is a simple maximum utility calculation over each offer that extends the current offer by proposing one more parameter. The utility of each offer is its value minus its cost, where the cost is simply some constant $k$ if the new parameter is already marked as rejected in the current Deal. If no extension can improve the current utility, then the current query is executed.

**Figure 2: Dialogue 1 - Query Negotiation**

```
S1 Välkommen till reserbokningstjänsten. Hur kan jag hjälpa dig? [Welcome to the travel bureau. How can I help you]
U1 Jag vill ha en resa till Göteborg från Stockholm [I want to travel to Gothenburg from Stockholm]
S1 SV=assertion,destination,göteborg
U2 Vilket datum vill du resa? [On what date will you travel?]
U3 SV=assertion,date,11/08/01
S1 Vilken tid kommer dom fram? [What time do they get there] U4 SV=query,arrival-time,X
S2 Vilken tid vill du resa ungefär? [What time will you leave] U5 SV=query,arrival-time,0800
S4 Jag vill vara framme i Göteborg före klockan åtta. [I want to be in Gothenburg for eight o'clock] U6 SV=assertion,arrival-time,11/08/01
S3 Det finns ett flyg som avgår klockan sju noll noll och ankommer klockan åtta noll noll det är det tidigaste flygret. [There is a flight that leaves at seven zero zero and arrives at eight zero zero. That is the earliest flight]

The Interpretation rules for updating Deal include (applied in parallel): For input SV = <assertion, p, v>

11. Deal(p) = notRaised or rejected → Deal(p) :- proposed
12. Deal(p) = proposed → Deal(p) :- accepted
13. ∀q,∀p (Deal(q) = proposed → Deal(q) :- rejected)
14. ∀q (Deal(q) = proposed → Deal(q) :- rejected)
For any input SV with parameter p
15. The first member of QueryStack to which p is relevant is promoted to the top of the stack; if there is none, then a relevant query is pushed onto QueryStack.

The Output rules for updating Deal are (for <question,p,X>)

O1. Deal(p) = notRaised or rejected → Deal(p) :- proposed
O2. ∀q,∀p (Deal(q) = proposed → Deal(q) :- accepted)

3. Corpus Analysis

We illustrate LEND on two dialogues from our Swedish trip planning corpus. For each Ut, we show the user utterance, a translation and its Semantic Value. All the dialogues generally extend beyond presentation of the initial travel possibility. They also include discussion of alternatives and other features such as taxi and hotel booking. Our concern is just the initial segment resulting in a route query. In our examples we omit the concept of start-point (which is not in the Query Space of Figure 1) from the Semantic Values.

Consider Dialogue 1 (Figure 2). Initially, all parameters are notRaised, the Context is empty and the Query Stack contains one item: route. We do not model production of the opening phrase S1 but assume it is entirely formulaic. The response U1 is interpreted (I1) as proposing destination and sets its Context value to göteborg. A query on only destination has a utility of 1. Selection finds the superior two-parameter query destination & date with a utility of 3.
destination & mode is also superior but only has a utility of 2. The system therefore queries with S2 thereby proposing date (O1) and accepting destination (O2). The response U2 is interpreted (I2) as accepting date and setting its Context value to 11/08/01. destination & date & mode has a utility of 5 which exceeds all other current possibilities. Therefore the system proposes mode through S3 (O1). (Again, we assume the precise format of S3 is just formulaic.) The response U3 is interpreted (I4) as a rejection of mode. It is not a proposal of arrival-time, even though the concept does appear in U3, precisely because the user is not proposing to query on arrival-time but to include it in the query results. Selection now finds that there is still a superior Deal: extending destination & date & mode & arrival-time but to include it in the query results. Selection thus rejects the current Deal with a utility of 4; extending it with mode only gives 2 (because of the cost involved in overriding the rejection); sticking with the current Deal only gives 3. Therefore the system issues S5 and proposes departure-time (O1). The response U5 is interpreted as a rejection (I4) of departure-time and a proposal (I1) of arrival-time. Finally, Selection can find no superior Deal therefore the system makes a route query with parameters:

\[ \text{destination = göteborg, date = 11/08/01, arrival-time = 0800} \]

and chooses an answer to suggest to the user.

Consider Dialogue 2 (Figure 3). Progress is similar until U3. cost is relevant not to route queries but price queries. Therefore, price is pushed on top of the Query Stack. Utility calculation will now switch to price worth assignments (not shown here) but still use the relevant parts of the current Deal and Context. Since there is no better price query than destination & mode, the query is made and the results given to the user. price is popped off Query Stack. On the user’s next turn, mode is re-proposed (I1). arrival-time is also proposed by the user although the current algorithm does not support simultaneous input of two semantic values. The result is the route query destination & mode & date & arrival-time with a utility of 6 that cannot be bettered. The query is therefore made and a selection shown to the user.

4. Discussion: Dialogue Structure in Theory

4.1. Negotiation Structures, Games and Intentions

How much structure (and of what sort) is there in dialogue? How much, and how reliably, can it be tracked? Negotiation dialogues, at least in classical game-theoretical accounts [7], possess minimal structure. They just consist in a sequence of offers, each being a counter by one party to the offer just made by the other party. In an interesting variation [8] stipulates a sequence of simultaneous offers by all parties. Linguistic analysis of ‘Task Oriented Domains’ (and our trip planning domain is as clear an instance as you will find) traditionally uncovers much more complex, usually tree-like, structures over utterances. For example, they may reflect relations between the intentions that motivated them [9] or perhaps nesting game structures [10,11]. It is of course quite natural to view a slot-filling form as a higher-level game aimed at undertaking a system action. The nested games are exchanges aimed at finding values for the parameters. On this story, Dialogue 2 displays a complete ‘trip game’ containing several nested games, eg S2-U2, and one S3-U4 contains yet another U3-S4.

Dialogue 1 is important because it resists this kind of analysis. If the objective of the game is to make the route query and the objective of sub-dialogues is to pre-establish the parameter values, then we simply note that only one such initiated game S2 is actually completed before the query is made. S3 and S4 are never answered at all. U3 is answered (to some degree, at any rate) but not before the query is made. Of course, one can begin to argue games need be completed by answering the question that initiated it. U4 may not answer S4 but it does answer a different question that is equally salient (in the context or through accommodation). What then of the relation of S4 to U3? Can one game be completed by initiating another on an equally salient theme? Probably such an analysis can be made to work although it may require Ptolemaic wheels of complexity.

Our analysis of Dialogue 1 is simply that possible constraints on a forthcoming query are bated between the two parties: S proposes travel-mode in S3 but U is not keen on this and suggests settling arrival-times first; S does not like this idea (to satisfy it, he’d have to make the very query he is intending) so he offers departure-time; U really doesn’t like this idea and insists on settling arrival-times. Agreement is reached. In fact, it transpires that the main motivating factor behind U’s utterances is his desire to be in Gothenburg by 0800. Quite possibly, he sought arrival times in U3 so that he could pick one that was before 0800 whilst also maintaining a link of conversational relevance to S’s preceding S3. However, it is quite unnecessary for S to attempt to divine this intentional structure. And this is just as it is in negotiation. One can negotiate perfectly well without knowing the intentions of the other party – although if you do know them, this may give you valuable strategic information.
4.2. Returns to open issues

Dialogue 2 is important because it does contain a ‘standard’ example of a nested dialogue (S3-U4 contains U3-S4) but the LEND model does not analyze it so. LEND treats U3 simply as a rejection of mode. U4, the ‘answer’ to S3, is analyzed as a re-proposal of mode. This is perhaps not intuitive. Nevertheless, the strategy works because the user himself maintains the necessary structure and returns to the travel-mode topic. Indeed, why should a system maintain structure that is not deemed necessary (a value for mode is not required, as seen in Dialogue 1) and which the user can maintain if he so desires? One might consider elevating the strategy into a point of principle: a principle of LENDing.

After all, if the user does not return to the travel-mode topic or does but only says I see or something, then probably one ought to consider the issue as just set-aside or lost.

The strategy is also exemplified in the Philips’ slot-filling architecture (though not in VoiceXML forms). The Philips’ system uses questions simply as helpful prompts to encourage supply of useful information. If the user supplies different information than that requested, it is treated just the same. It is just new information to integrate into the current set of slots and values. Nothing in the system notes that a question was unanswered. Nothing notes that a question move was made. If the unfilled slot needs filling, the question is just asked again. It is of course necessary to maintain some state. It is required for resolving context sensitive constructions, such as pronouns and ellipses, into context independent Semantic Values. This point is relevant to cases illustrated in Figure 4.

Figure 4: Two Artificially Constructed Dialogues

<table>
<thead>
<tr>
<th>Dialogue AC1</th>
<th>Dialogue AC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Do you want to take the train?</td>
</tr>
<tr>
<td>U1</td>
<td>How much does the train cost?</td>
</tr>
<tr>
<td>S2</td>
<td>By train, the price is 800 crowns return</td>
</tr>
<tr>
<td>U2</td>
<td>Ok. Well yes then.</td>
</tr>
<tr>
<td>S1</td>
<td>When do you want to leave?</td>
</tr>
<tr>
<td>U1</td>
<td>How long does the journey take?</td>
</tr>
<tr>
<td>S2</td>
<td>By train, the journey takes 4 hours</td>
</tr>
<tr>
<td>U2</td>
<td>Ok. Eight in the morning please</td>
</tr>
</tbody>
</table>

For dialogue AC2, S1 answers U1 in a manner that one might think to be perfectly reasonable. However, this question is not really the question intended. The user in this situation is trying to tell the system what their plans are. It may be argued that knowledge of dialogue structure will help in finding the right context for resolution. Nevertheless, the phenomenon is probably of limited practical importance. First, other resolution methods will often be able to cover the data. If U2 in dialogue AC1 were expressed as Ok, the train, then, in our domain, this could only be an assertion of a value for mode. Knowledge of dialogue structure would not be required to reach this conclusion. Secondly, if the intervening dialogue between When do you want to leave? and Eight in the morning contains further embedded structure or is even just much longer, then even most humans will begin to find resolution problematic. In these cases, it is quite reasonable for the computer to announce that it could not understand the pronoun (if one was used) or the utterance (if it was elliptical) and force the user into a fuller enunciation of his meaning. In cases where the user does fully enunciate, as Dialogue 2 illustrates, recognizing that the utterance answers an earlier question is not necessary.

4.3. Compositional dialogue

The complexity of the phenomena in Dialogues 1 and 2 is clearly not due to any inherent complexity in the individual utterances that make them up. In Dialogue 1 for example, What time do they get there? and I want to be in Gothenburg for eight o’clock are simple within-domain utterances and not exceptional in any way. What is interesting about them is just the dialogue positions that they occupy. To paraphrase [12], what are these familiar utterances doing here?

5. Conclusions

LEND is a new dialogue management model for simple enquiry dialogues which analyzes utterances as moves in a negotiation. In particular, sequences of utterances involving consecutive questions and non-answering statements become simple to interpret as consecutive bids in a negotiative dialogue with a very flat structure. The analysis also permits the processing of dialogues that appear to possess a traditional ‘nested’ dialogue structure.

Acknowledgments

This research was carried out under the SIRIDUS project (EU RTD project IST-1999-10516 while the author was employed at SRI International, Cambridge, UK. We thank Telia Research AB Vitsandsgatan 9, S-123 86 Farsta, Sweden for making the Swedish travel corpus available to us; also, audiences in Oxford and Edinburgh, and Manny Rayner in Cambridge, for criticism and comment.

6. References