ABSTRACT

We describe a procedure for contextual interpretation of spoken sentences within dialogs. Task structure is represented in a graphical form, enabling the interpreter algorithm to be efficient and task-independent. Recognized spoken input may consist either of a single sentence with utterance-verification scores, or of a word lattice with arc weights. A confidence model is used throughout and all inferences are probability-weighted. The interpretation consists of a probability for each class and for each auxiliary information label needed for task completion. Anaphoric references are permitted.

1. INTRODUCTION

We are interested in spoken dialog systems in which the caller responds using fluent natural language to the prompt "How may I help you?" (HMIHY). In previous work we have described the speech recognizer [1], automatic acquisition of salient phrase and grammar fragments, and call-type classification [2,3], the dialog manager [4], and the incorporation of utterance verification [5]. In this paper we consider the issue of spoken language understanding within dialog.

At each turn in the dialog, the speech-recognized response from the caller must be interpreted, see [6-10]. Ambiguities and conflicting information need to be resolved so far as possible using confidence scores and dialog context, and to the extent that this is not possible the dialog manager may initiate a clarification sub-dialog [4]. Auxiliary information that is necessary for completing a service must be extracted. Furthermore, a range of proficiency from novice to power-user must be catered for.

In this paper we describe an approach to context-dependent interpretation in which task structure is represented in a simple graphical form. The block diagram of the understanding system is shown in figure 1. The latest recognized sentence with utterance-verification scores, or alternatively a word lattice with arc weights, is initially passed through a task-dependent preprocessor in order to replace certain classes of words or word-strings (such as pronouns or digit strings) with nonterminal symbols. The understanding module itself is task-independent, and consists of a classifier and an interpreter. The purpose of the classifier is to find the surface meaning of the latest sentence, and that of the interpreter is to follow the implications of this for the dialog context.

The interpreter algorithm performs an inference using the task structure graph, the dialog context and the classification of the sentence, in order to arrive at a result which is conveyed to the dialog manager. The dialog manager makes use of the outcome to determine the status of the dialog after the latest turn, and can then initiate the next turn accordingly.

The task structure graph governs the behaviour of both the dialog manager and the understanding module. It also serves as the basis for the protocol for exchanging information between the two modules. The dialog context is sent from the dialog manager to the understanding module, expressed as a path into the graph, and the result returned to the dialog manager consists of a set of probabilities for the nodes of the graph. In this paper we describe the graph structure, the interpreter algorithm, and application to two dialog tasks.

2. TASK STRUCTURE GRAPH

The task structure is specified using three small data files, which together define the graph. This has a dual rôle in our system: it represents both the semantic structure including the needed secondary attributes (similar to the E-form in [7]), and the sources of ambiguity that need resolution. The graph nodes represent the following:

- call-type labels: the set of services that characterize the application, and
- auxiliary labels: the secondary attributes necessary for completing the service.

The connections within the graph are of three kinds:

- primary arcs: directed from node to node to represent the is-a and has-a relations between labels [10],

Figure 1: Block diagram of spoken language understanding system.
• exclusives: undirected, expressing the incompatibility of certain sets of labels,
• implicit confirmations: directed (possibly bidirectional), expressing the ways in which one label may implicitly confirm another.

The graph should not be interpreted as a Bayesian network: absence of an arc does not indicate independence. The graph for a customer-services application [3] is shown in Figure 2. The chain-dot arc from “forward_number” to “collect” is an example of an implicit confirmation; most of these are omitted for simplicity.

3. INTERPRETER ALGORITHM

3.1 Summary
The interpreter uses a three-pass algorithm over the nodes in the graph. The dialog context is received from the dialog manager and is expressed as a path through the graph to the apex. For example, the following path in Figure 2:

calling_card→billing_method→dial_for_me→HMIHY?

applies in calling_card situations where the dialog manager is seeking confirmation: “Do you want to make a card call?”.
The apex node can be omitted. The lower end of the path (first node) is the focus for the latest turn, and normally any explicit confirmation or denial refers to that focus. The only implicit confirmation arcs that are active terminate at the focus. An important step in the interpretation arrives at a decision for the status of the focus after the latest sentence from the user. During the first pass, information is propagated through the graph so that evidence supporting and opposing the focus is gathered. Opposing evidence may consist of an explicit denial, which may depend upon the outcome at the focus. The only implicit confirmation arcs that are active terminate at the focus. An final probability for the focus is assigned, which may involve reconciling the evidence in favour with that against it. A final probability for all other nodes is arrived at through the second and third passes. The order of visiting the nodes is different for each pass, and is based on a topological sort using all the arcs that are active for that pass.

3.2 Forms of implication
The basic principle for propagation of evidence is the rule

If \( X \rightarrow Y \) then \( P(Y) \geq P(X) \)

This is used in three ways:

• Forward implication (up the graph along primary and implicit confirmation arcs),
• Backward implication (down the graph) in the form: If \( \cdot \rightarrow X \) then \( P(X) \leq P(Y) \),
• Transverse implication (across the graph) in the form: If \( X \perp Y \) then \( P(X) \leq P(\overline{Y}) \).

(The notation \( X \perp Y \) denotes that \( X \) and \( Y \) are incompatible, see below for definition).

In order to propagate evidence up the graph from a node, a probability distribution is assigned over the destination nodes for the arcs leaving that node. Nodes are considered in topological order and evidence is gathered at each node \( Z \) using a simple disjunctive rule

\[
P_{\text{forward}}(Z) = \max_{U \cup Z} \left\{ P_{\text{direct}}(Z), \max_{U} P(Z \mid U) P_{\text{forward}}(U) \right\}
\]

Alternative disjunctive combination rules could be used, but have not yet been tried. The direct evidence \( P_{\text{direct}}(Z) \) arrives from the classifier. For the inner maximization, all propagation probabilities are zero except for the child nodes \( U \) of \( Z \), and any node \( U \) for which all paths up the graph pass through \( Z \) in which case the propagation probability is unity. The distribution from \( Z \) over its parent nodes is computed after this rule is applied, because it may depend upon the outcome at \( Z \).

Transverse implication involves a conjunctive rule:

\[
P_{\text{transverse}}(Z) = \min_{U \cup Z} \left\{ P_{\text{forward}}(Z), \min_{U \cup Z} P_{\text{forward}}(U) \right\}
\]

For any node \( Z \), the set of nodes \( U \) that are incompatible with it is in general larger than the initial set of exclusives given as part of the graph specification (section 2). For any two nodes \( U, V \), if there exist disjoint subsets \( S_U, S_V \) of nodes such that all paths from \( U \) pass through \( S_U \), all paths from \( V \) pass through \( S_V \), and each element of \( S_U \) is exclusive of each element of \( S_V \), then \( U \) and \( V \) are incompatible: \( U \perp V \). The set of all incompatible pairs of nodes is computed in advance, from the graph specification tables.

Backward implication also involves a conjunctive rule:

\[
P_{\text{backward}}(Z) = \min \left\{ P_{\text{forward}}(Z), P_{\text{backward}}(\overline{P(Z)}) \right\}
\]
The results show that the algorithm described in this paper out-

3.4 Description of algorithm

During the first pass, all evidence that may be interpreted as

3.3 Propagation probability assignment

A node may have several parent nodes, representing a situation

4. APPLICATIONS

4.1 Customer services

To illustrate the operation of this procedure, consider the

4.2 Call processing

4.3 Billing

Figure 3 shows ROC curves for a test set of 1841 spoken

4.4 Applications

4.4.1 Billing credit

The inference rules are consistent with various logic systems

4.4.2 Billing credit

including the many-valued logic of Fakasiewicz [11].

The distribution assignment is designed to

4.4.3 Billing credit

minimize this.

4.4.4 Billing credit

In fact we use a combination of factors 3 to 5 to over-ride the

4.4.5 Billing credit

default, with factor 5 having highest priority, followed by 3

4.4.6 Billing credit

and it then becomes the "dialed_number" for a billing credit

4.4.7 Billing credit

node but is another intermediate result for a node which is

4.4.8 Billing credit

on the path, but is another intermediate result for a node which is

4.4.9 Billing credit

important for both forward and backward implication.

4.4.10 Billing credit

where \( \frac{P(z)}{P(z)} \) is the first node reached from \( Z \) (in topological order) and that all paths from \( Z \) pass through it. Every node in

4.4.11 Billing credit

different from that in the first pass.

4.4.12 Billing credit

During the second pass, supporting evidence for each node

4.4.13 Billing credit

is pulled in from its child nodes (other than the dialog focus) is pulled in from its child nodes

4.4.14 Billing credit

and against any contrary evidence either from explicit denial or from incompatible nodes

4.4.15 Billing credit

by transitive and conjunctive combination, and finding a good

4.4.16 Billing credit

model is largely heuristic matter.

4.4.17 Billing credit

The result is a final probability for a node which is on

4.4.18 Billing credit

the path, but is another intermediate result for a node which is

4.4.19 Billing credit

on the path, but is another intermediate result for a node which is

4.4.20 Billing credit

important for both forward and backward implication.
performs an earlier version which was deployed for that trial, which incorporated a much simpler task structure and inference model.

4.2 VPQ: a directory information service

For a second application see Figure 4: the task structure graph for VPQ, a voice-activated directory information service [12]. Through this structure the user may call or page individuals identified by name, or engage in a dialog in order to search for some information. The dialog may continue through a series of services within a single session. The implicit confirmation arc from “action” to “continue” allows a user to respond affirmatively to the question of whether they require a further service by actually initiating one. Anaphoric references to individuals are also permitted.

5. CONCLUSIONS

We have described a procedure for contextual interpretation of spoken sentences within dialogs, in which a probability-weighted inference is performed using a graphical model of task structure. Evidence is propagated between the nodes of the graph in a way controlled by the dialog context, and is combined at the nodes using logical rules. The procedure delivers a result to the dialog manager from which the future direction of the dialog can be derived. The procedure has so far been applied to two applications.

6. REFERENCES


Figure 3: ROC curves for “billing method” response sentences.

Figure 4: Task structure graph for VPQ.