Robust Interpretation in Dialogue by Combining Confidence Scores with Contextual Features*

Matthew Purver†, *Florin Ratiu*†, *Lawrence Cavedon*‡

†CSLI, Stanford University, USA ‡National ICT Australia and CS&IT, RMIT University, Australia
{mpurver, fratiu}@stanford.edu lcavedon@cs.rmit.edu.au

Abstract

We present an approach to dialogue management and interpretation that evaluates and selects amongst candidate dialogue moves based on features at multiple levels. Multiple interpretation methods can be combined, multiple speech recognition and parsing hypotheses tested, and multiple candidate dialogue moves considered to choose the highest scoring hypothesis overall. We integrate hypotheses generated from shallow slot-filling methods and from relatively deep parsing, using pragmatic information. We show that this gives more robust performance than using either approach alone, allowing n-best list reordering to correct errors in speech recognition or parsing.

Index Terms: dialogue management, robust interpretation

1. Introduction

CSLI has been developing a multi-domain spoken-language dialogue management system for a number of years, and applied this system to a range of applications, such as control of robotic devices [1], intelligent tutoring [2], and interactive control of in-car devices [3]. A primary concern is leveraging dialogue context to improve the robustness of speech recognition (ASR) and dialogue interaction [4]. This paper describes recent extensions to further address the issue of robust interpretation. Multiple candidate hypotheses from different sources (e.g. deep syntactic parsing and shallow topic classification) are evaluated and assigned overall confidence scores using features at multiple levels (e.g. acoustic, semantic and context-based). Although initially motivated by the extension to plug-and-play multi-device dialogue management [5], particularly for the in-car environment, the approach allows improved selection from an n-best list of recognition and interpretation hypotheses, and it is this issue that we focus on here.

In our approach, all devices (and in fact, all possible interpretation methods associated with each device) perform shallow processing of the incoming utterance, each producing multiple possible candidate dialogue moves. Potential device-move combinations are then scored against a number of features, including speech-recognition and parse confidence, discourse context, current *device-under-discussion*, and NP argument analysis. The device associated with the highest-scoring dialogue move is given first option to process the utterance. A disambiguation question may be generated if no device is a clear winner, or a confirmation question if the winning bid is not scored high enough.

*Cavedon's contribution was performed while employed at CSLI, Stanford University. This research was supported by the National Institute for Standards and Technology (NIST). We thank Fuliang Weng and our partners at VW America and Robert Bosch Corporation, who (amongst other things) developed the parser and semantic classifier used here.

Device choice, move choice, and selection of best ASR/parser hypothesis are thereby made simultaneously, rather than being treated as independent processes. As well as allowing for principled device identification, this has the benefit of allowing competing interpretation hypotheses to be scored on the basis of multiple information sources, including context. The highest scoring result overall may not correspond to the highest-confidence result from the ASR or parser n-best list alone; instead, n-best lists are effectively re-ordered based on device and dialogue context, allowing parsing errors such as incorrect PP-attachment to be automatically corrected. Confirmation and clarification behaviour can also be governed not only by ASR or parse confidence, but by the overall score. This results in measurable improvement with respect to selecting most appropriate candidate dialogue move.

Related Approaches: [6] combines ASR confidence scores with various intra-utterance linguistic features to re-order n-best hypotheses; [7] also includes move bigram statistics. [8] uses similar feature combination to identify misrecognised utterances. More recently, [9] also include pragmatic information such as NP resolution, and simultaneously choose from an n-best list while identifying misrecognition; they also divide misrecognised utterances into two overall confidence ranges, one for outright rejection and one for confirmation/clarification. Similarly [10] combines acoustic confidences with semantic information, and [11] with bridging reference resolution, in order to allow clarification on an integrated basis. We extend these techniques by considering the multi-device situation, combining multiple interpretation techniques, and using an extended set of features for scoring candidates.

2. Background

We use an *information-state update* (ISU) approach to dialogue management [12, 4], which can handle richer dialogue phenomena than finite-state-based approaches. We maintain a tree-based model of dialogue context (including a structured move history as well as e.g. referents for anaphora resolution) together with a set of update rules defining the effect of dialogue moves on this state. In this *dialogue move tree* (DMT), each dialogue move is represented as a tree node, and incoming moves are interpreted in context by attachment to an appropriate open parent node (for example, `WhAnswer` moves attach to their corresponding `WhQuestion` nodes). Update rules not only define the possible attachments but their effects (e.g. adding new information and referents, triggering new tasks, activities and system responses).

Dialogue Move Scripting: Our system architecture is coded in Java, but in order to facilitate customization to new domains, and specification and addition of new plug-and-play devices together with their dialogue capabilities, we provide a separate dialogue move scripting language (partially illustrated in Listing 1; see [5] for details). Scripts can be added or edited as text without recompilation, and serve a number of purposes:

1. hierarchical definition of dialogue moves, allowing inheritance and re-use of existing moves, while allowing customization to a specific domain;
2. mapping of utterance representations to appropriate dialogue moves, including arguments for device models;
3. definition of attachment rules for information-state update;
4. move-specific specification of output to be generated, for disambiguation or requests for missing information.

Dialogue move scripts provide templates which give (possibly underspecified) definitions of the various interpreted forms which can trigger each type of move. Variables in the script correspond to variables in the *Activity Model* describing the corresponding device (e.g. an MP3 device contains a `play` operation with a required *_playable-object* argument). When an incoming utterance matches the MP3 input template in Listing 1, the *_playable-object* script variable is filled by unification, and resolved to an object from the device's domain which then fills the corresponding slot in the activity; this creates a new move hypothesis. Each move type is also given a declarative specification of possible subsequent dialogue move attachments (for example, reports of success or requests for clarification by the system) plus certain of their update effects (e.g. particular attachments may close tree branches, making nodes no longer available as antecedents). Further details are provided in [5].

3. Multiple Interpretation Methods

The general approach to triggering candidate dialogue moves allows us to specify multiple interpretation methods in parallel. Most systems use a single interpretation mechanism which is best suited to the application at hand, be it e.g. an open-domain statistical parser, a domain-specific constraint-based grammar, or keyword-spotting techniques. We extend this here to allow arbitrary multiple interpretation mechanisms, each producing its own (independent) interpretation hypothesis and associated confidence.

We use a speech recognizer with a statistical language model, a statistical dependency parser, and a probabilistic slot-filling classifier; the implementations are common across devices, although at this stage in development the models themselves are trained on device-specific data. The syntactic parser (a version of [13]) produces a dependency structure specifying head words, predicates, arguments and grammatical relations; the semantic classifier (similar to [14]) uses a maximum entropy approach [15] to produce topic keywords and/or attribute-value pairs for the relevant domains. Both produce scored lists of n-best candidates. Importantly, this approach means that we cannot assume a 1-to-1 mapping between utterance representations and device/move choice, as might be the case with hand-built grammars or device-specific ASR. It also means that the accuracy of a single interpretation component (e.g. the parser) is lower than might be obtained by hand-crafting. However, we can use our general approach to scripting and scoring move hypotheses to combine and re-order the individual hypotheses, increasing performance and robustness.

Dialogue move scripts (see above) can now be used to construct multiple candidate dialogue moves for an utterance. This is governed by the *Input* field (see Listing 1) for each move type, which specifies a set of patterns: if an utterance representation matches, a candidate node of this type is created. These patterns specify an interpretation method as well as the interpreted form itself: `SYN` patterns match the output of the statistical parser, `SEMCAT` patterns match the output of the classifier (either a general topic keyword or a slot-value pair), while `AND` patterns match combinations of the two.[1]

Each pattern is associated with a weight, used in the overall move scoring function described in Section 4 below. This allows moves created from matches against deep structure to be scored highly (e.g. `SYN` patterns in which predicate and arguments are specified and matched against), shallow matches to be scored low (e.g. simple `SEMCAT` topic matches), and combined matches to have intermediate scores (e.g. `SEMCAT` slot-value matches, or combinations of a `SEMCAT` topic classification with a `SYN` parse containing a suitable NP argument). Depending on other elements of the scoring function (e.g. the ASR confidence associated with the hypothesised string being tested) and on competing move hypotheses, low scores may lead to clarification being required (and therefore clarification will be more likely when only low-scoring (shallow) patterns are matched). Behaviour can therefore be made more robust: when deep parsing fails, a shallow hypothesis can be used instead (clarifying/confirming this specific hypothesis if required depending on its confidence) rather than resorting to a rejection or general clarification. Scores are currently set manually and determined by testing on sample dialogues; future work will examine learning them from data.

4. Dialogue Move Selection

In the general case, multiple possible candidate dialogue moves will be produced for a given utterance, for a number of reasons:

1. multiple hypotheses from ASR/parser output;
2. multiple interpretation methods (e.g. deep vs. shallow);
3. multiple possible move types for a candidate interpretation;
4. multiple antecedent nodes (active dialogue threads), including multiple devices, for a particular move type.

These are not independent: we must consider all factors simultaneously, to allow an integrated scoring function for each candidate and thus consider the best overall. The skeleton algorithm for instantiating and selecting a dialogue move is therefore as follows:[2]

```
foreach open node O
    foreach n-best list entry N
        foreach matching script entry M
            create candidate move
score all candidates
if (score(top) >> score(second))
    select top candidate
else
    generate question to disambiguate
if (score(selected-node) < threshold)
    generate question to confirm
```

The interesting aspect of the above process is the scoring function. Dialogue move candidates are scored using a number of weighted features, ranging from speech-recognizer confidence,

[1]Further general pattern types are available (e.g. `LF` for semantic logical forms, `STRING` for surface string keyword-matching).

[2]Note that we will not create $O \times N \times M$ candidates: only a subset of script entries (if any) will match for each node and n-best entry.

```
User Command:play { // inherits from generic Command dialogue move
  Input {           // templates for triggering move from processed utterance
    // full parse match: ''play/start X''
    1.0   SYN{ s( features(mood(imperative)), predicate(#play/vb|#start/vb),
                ?arglist(obj:_playable-object,?sbj:*)) }
    // topic classifier match
    0.1   SEMCAT{ topic(play_item) }    ... }
  Producing {    // templates for possible matching moves: e.g. confirmation; clarification question
  ... } ... }
```

Listing 1: Sample dialogue move script for a `play` `Command` for an MP3 device

through to pragmatic features such as the "device in focus" and recency of the DMT node the candidate would attach to. The full list of features currently considered is shown in Table 1, although those shown italicized are not currently used, either for reasons of computational overhead (full referent resolution in a large knowledgebase takes time) or lack of domain data (e.g. for move bigram frequencies). Note the inclusion of features at many levels, from acoustic recognition confidences through syntactic parse confidence to semantic and pragmatic features.

Reordering n-best candidates: Choosing the overall highest-scoring candidate therefore allows n-best list re-ordering: while the n-best list rank and confidence are factors in the overall score, other features may outweigh them, resulting in an initially lower-ranked n-best entry becoming the highest-scoring dialogue move.

Initial testing and setting of scoring function weights was performed on a manually constructed set of 400 test utterances for a single device (a restaurant recommendation system), of which 300 were also used to train the statistical parser and 100 were unseen variations. These were provided as manual transcriptions rather than as ASR hypotheses. This leaves multiple parser hypotheses as the main source of multiple dialogue move candidates. Development results on this set were encouraging even with the restricted subset of features from Table 1: looking at n-best parse reordering only, the percentage of sentences for which the correct parse is chosen increased from 90% to 94%, a 41% reduction in error with several common parse errors being corrected. A particular example is incorrect PP-attachment (a notoriously difficult challenge for statistical parsers). The example below (from a restaurant recommendation scenario), shows the top two n-best list entries from the syntactic parser:

```
1. how about [a restaurant [in Grant]] [on Elm]
2. how about [a restaurant [in Grant] [on Elm]]
```

Here, the second is lower-ranked but correct (i.e., both PPs modify *restaurant*). In the corresponding dialogue moves generated from these parses, the second has two database constraints filled (city and street name), while the first has just one, boosting the overall score of the candidate move corresponding to the first parse. Similar improvements are seen with examples involving nominal modifiers:

```
1. how about [a [[cheap] chinese] restaurant]
2. how about [a [cheap] [chinese] restaurant]
3. how about [a [cheap chinese] restaurant]
```

Here the second parse is correct, treating *cheap* and *chinese* as both independently modifying *restaurant*; the first takes *cheap* as modifying *chinese*, and the third takes *cheap chinese* as a single multi-word unit. Again, as the candidate move corresponding to the second parse fills two database-query constraints (price level and cuisine type), it scores highest overall.

Method	Set 1	Set 2
Top parse only	52.4%	30.8%
Classifier only	n/a	60.5%
First good combination	75.3%	67.9%
Best scored combination	77.7%	72.8%

Table 2: Evaluation results

Evaluation: We performed an initial evaluation of the approach on independent test data. This was taken from a system test in which trials were conducted with 20 naive subjects for each of two devices independently (an MP3 player device and a restaurant selection (RS) device). Trials were conducted using ASR; all subjects were native US English speakers. With the MP3 device, subjects performed 11 tasks, providing about 1400 utterances; with the RS device, 9 tasks, producing just over 1000 utterances. Task completion rates were good (98% and 94% for the two devices); semantic interpretation accuracy was reasonable (f-scores calculated over the database constraints associated with each utterance were 82.5% and 82.2%).

To assess the impact of our combined scoring approach, we took the logs from 3 test subjects and 2 similar runs performed after the test (covering the same tasks but adding more variety of different types of query), giving a set of 222 utterances which were manually annotated for parse correctness. Of these, 85 involved simple dialogue moves (e.g. `Yes/No-Answer` or `Confirmation`) which were all covered by the parser (and all correctly parsed); and 56 involved speech recognition failures or bad misrecognitions which only the shallow semantic classifier could process. Hence we had 166 utterances ("Set 1") on which we could evaluate parse re-ordering, and 81 utterances ("Set 2") to evaluate classifier/parser combination. Results are shown in Table 2: the integrated scoring option ("best scored combination (of parse and topic-classifier)") clearly outperforms the 1-best parse and classifier-only versions. More interestingly, it also outperforms a baseline result (labelled "first good combination") achieved by allowing the full n-best parse list, and combination with the classifier output, but searching down the n-best list and stopping as soon as a good combined hypothesis is found (above a score threshold). This improvement corresponds to a 9.8% error reduction on Set 1 (15.4% on Set 2), showing that considering and scoring all hypotheses does give an appreciable improvement.

5. Discussion, Future Work, Conclusions

Move type comparison: The scoring function for feature combination is currently manually defined. When comparing between candidate moves of the same type, this is not trivial; when comparing candidates of different types it becomes even less so, as some move types and some DMT attachment contexts will allow only a subset of the features to have meaningful values. However, com-

Recognition features:	ASR and parse probabilities;
	ASR and parse n-best ranks;

Semantic features:	topic classification for the parse (with score); for dialogue moves spawning activities: - number of input slots filled; *- number of resolved/unresolved/ambiguously resolved slots after NP resolution;* for queries about database objects: - set of constraints sent to the knowledge base; *- cardinality of the set of knowledge base query results;*
Contextual features:	current most active node; current activity; position and recency of the parent node in the active node list; *dialogue move bi-gram frequencies, based on both tree attachment and temporal sequence*

Table 1: Move Scoring Features

parison between move types is essential, as two ASR hypotheses with similar recognition scores may have very different possible move types (compare e.g. the `Command` *"Play a rock song by Cher"* with the `Query` *"What rock songs are there?"*). We are therefore currently investigating the use of machine learning techniques to improve on our current manual definitions, building on the approach of [9] by using a wider feature set. With annotated data the optimal weights of a scoring function that combines all the features can be automatically learned.

Confirmation/clarification: Use of a score which essentially combines confidence at many levels is also advantageous for confirmation and clarification strategies. We can use overall *absolute* score to decide whether to accept a move, ask for explicit clarification, implicitly confirm in the next system move, or reject it outright [16] by checking against a range of predefined thresholds. We can also compare the *relative* scores of the highest competitors to decide whether to accept the winner unambiguously, or ask a disambiguating clarification question. The threshold values are currently specified manually in dialogue move scripts; a future direction is to automatically learn optimal values.

Conclusions: Our general approach to interpretation and dialogue move selection—creating multiple potential candidate moves and scoring them via the combination of features from multiple sources—has been shown to increase robustness of dialogue performance by re-ordering lists of n-best candidates. Future work includes learning optimal weights for the scoring function, and optimal values for confirmation and error thresholds.

6. References

[1] O Lemon, A Bracy, A Gruenstein, and S Peters, "A multimodal dialogue system for human-robot conversation," in *Proc. 2nd Annual Meeting of the NAACL*, 2001.

[2] B Clark, E. O Bratt, O Lemon, S Peters, H Pon-Barry, Z Thomsen-Gray, and P Treeratpituk, "A general purpose architecture for intelligent tutoring systems," in *International CLASS Workshop on Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems*, 2002.

[3] F Weng, L Cavedon, B Raghunathan, D Mirkovic, H Cheng, H Schmidt, H Bratt, R Mishra, S Peters, L Zhao, S Upson, E Shriberg, and C Bergmann, "A conversational dialogue system for cognitively overloaded users," in *Proc. 8th INTERSPEECH*, Jeju Island, Korea, 2004.

[4] O Lemon and A Gruenstein, "Multithreaded context for robust conversational interfaces," *ACM Transactions on Computer-Human Interaction*, vol. 11, no. 3, 2004.

[5] D Mirkovic and L Cavedon, "Practical plug-and-play dialogue management," in *Proc. PACLING*, Tokyo, 2005.

[6] M Rayner, D Carter, V Digalakis, and P Price, "Combining knowledge sources to reorder n-best speech hypothesis lists," in *Proc. ARPA Human Language Technology Workshop*, Plainsboro, NJ, 1994.

[7] A Chotimongkol and A Rudnicky, "N-best speech hypotheses reordering using linear regression," in *Proc. 7th EUROSPEECH*, 2001, pp. 1829–1832.

[8] M Walker, J Wright, and I Langkilde, "Using natural language processing and discourse features to identify understanding errors in a spoken dialogue system," in *Proc. 17th International Conference on Machine Learning*, 2000.

[9] M Gabsdil and O Lemon, "Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems," in *Proc. 42nd Annual Meeting of the ACL*, Barcelona, 2004, pp. 344–351.

[10] M Gabsdil and J Bos, "Combining acoustic confidence scores with deep semantic analysis for clarification dialogues," in *Proc. 5th International Workshop on Computational Semantics (IWCS-5)*, Tilburg, 2003.

[11] D Schlangen, "Causes and strategies for requesting clarification in dialogue," in *Proc. 5th SIGdial Workshop on Discourse and Dialogue*, Boston, 2004.

[12] S Larsson and D Traum, "Information state and dialogue management in the TRINDI dialogue move engine toolkit," *Natural Language Engineering*, vol. 6, 2000.

[13] F Weng, N Jin, J Meng, and Y Zhu, "A novel probabilistic model for link unification grammar," in *Proc. 7th International Workshop on Parsing Technologies*, Beijing, 2001.

[14] Y He and S Young, "A data-driven spoken language understanding system," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, US Virgin Islands, 2003.

[15] Y Zhou, F Weng, L Wu, and H Schmidt, "A fast algorithm for feature selection in conditional maximum entropy modeling," in *Proc. Empirical Methods in Natural Language Processing*, Sapporo, Japan, 2003, pp. 153–159.

[16] R San-Segundo, J. M Montero, J Ferreiros, R Córdoba, and J. M Pardo, "Designing confirmation mechanisms and error recover techniques in a railway information system for Spanish," in *Proc. 2nd SIGdial Workshop on Discourse and Dialogue*, Aalborg, 2001, pp. 136–139.