Florin Ratiu

CS229 Final Project

Reordering Attachment Candidates in the CSLI Dialogue System's DMT

1. Abstract

This paper describes an approach for selecting the best candidate dialogue move in multi-device dialogue systems based on multiple sources of information, including parsing hypotheses, contextual information, information from topic classifiers and semantic features. At the basis of this approach stands the ability to re-order n-best lists of inputs using the features mentioned above, thus potential errors from speech recognition and parsers can be corrected. Machine learning techniques are compared to the baseline system, which is based on hand-crafted heuristics.

2. The CSLI Dialogue System

The CSLI Dialogue Manager (CDM) implements the Information State Update (ISU) paradigm to dialogue management. This approach provides a powerful representation of the dialogue by collecting information relevant to the dialogue context in a central data structure. This context is used to interpret incoming utterances, do resolution of noun phrases (NPs), construct responses, maintain state, etc. In CDM the ISU approach has been designed to support multi-threaded and multi-topic conversations. The central data structures used to support these are the Dialogue Move Tree (DMT) and the Activity Tree (AT). An overview of the system architecture is presented in Figure1 below.

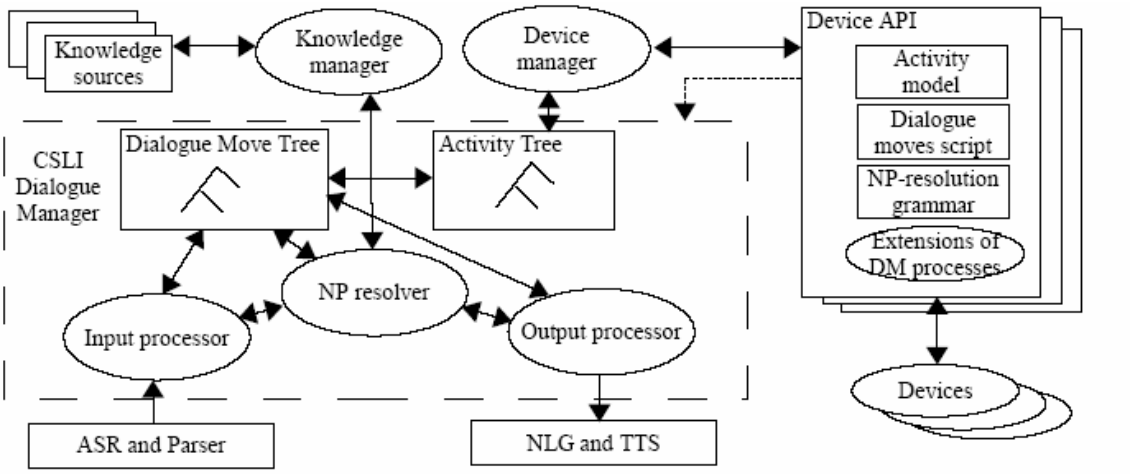

Figure 1: Dialogue System Architecture

The DMT has a tree structure that represents the entire dialogue history. Each subtree represents a thread in the conversation and incoming utterances, interpreted as dialogue moves, are transformed into nodes. Multi-threading and multi-device dialogue is supported by allowing nodes created from utterances corresponding to different topics of conversation or addressing different devices to attach in different subtrees. The root of the tree is a node responsible for managing all the devices the user can address, and the

next level corresponds to the managers of these different devices. All other nodes correspond to user or system utterances.

The Active Node List (ANL) is a list of all the nodes in the DMT that accept attachment of other nodes. They correspond to open topics of conversation, to which new utterances can be added.

The nodes in the DMT express how user's utterances and system's responses are incorporated in the dialogue context. A description of these nodes can be found in [4]. There is a scripting language that defines the conversion from utterances and actions to the DMT nodes, which allows easy customizations of these conversions to new domains and applications. The script can support hierarchies of dialogue moves and provides semantic templates that define valid attachment of new nodes. Examples of the script are also presented in [4].

During the process of interpreting and responding to user's utterances and creating new activities the system is trying to resolve NPs extracted from utterances. This might involve interaction with the knowledge manager and querying of the knowledge base. At the dialogue level, system responses correspond to user nodes in the DMT and some might be grounding points or clarification questions in the dialogue.

3. The Attachment Algorithm

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

- multiple hypotheses from ASR/statistical parser output;
- multiple interpretation methods (deep parsing vs. shallow classification);
- multiple possible move types for a candidate interpretation;
- multiple antecedent nodes (active dialogue threads), including multiple devices, for a particular move type.

They are not independent: it is important to 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:

```
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 ))
then
        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, 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 the table above. Note the inclusion of features at many levels, from acoustic recognition confidences through syntactic parse confidence to semantic and pragmatic features.

Recognition features:	- Parse probabilities and ranks; - NOTE: unfortunately there is only one speech hypothesis available (the project is joint work with Bosch RTC and they only provide a single SR result)
Semantic features:	- Topic classification for the parse; - number of slots filled/(ambiguously) resolved/unresolved by input pattern (that tries to match the parse); - set of constraints (in a frame object) that is sent to the knowledge base; - number of kb results for such a frame;
Contextual features:	- number of kb results for such a frame; - position and recency of the parent node in the DMT; - frequency of DMT attachments - pairs of child and parent node types; - pairs of chronologically consecutive nodes;

This integrated scoring mechanism therefore allows n-best list input to be re-ordered: dialogue-move candidates are potentially instantiated for each n-best list entry and the highest-scoring candidate chosen. 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.

The default attaching algorithm used only a small subset of these features to score attachments and then selects the "best" one. Instantiation of a dialogue move is conditioned on the parse of the user utterance matching a specified pattern (this is just a parse tree) and/or a topic classifier. Once this first step is completed the attaching algorithm uses the statistical parse and topic scores (probabilities and likelihoods) and a "should attach" score (defined arbitrary by developer to indicate how likely is a child user node type to attach to a parent node type) to compare the candidate possible attachments. Experience with this system showed that incorporating a larger set of features is hard, many heuristics are not very intuitive and complexity increases very fast.

All these suggest an alternative approach: apply machine learning for selecting the best dialogue move candidate. For each user utterance in the training set we create a set of candidate dialog moves (pairs of parent – child node, where the child always corresponds to the user utterance), based on the state of DMT and matchings of n-best list of parses with patterns and topics. For each such possible attachment we can compute a vector of the features presented above and manually label it as correct/incorrect attachment.

4. Data Collection

Data used for these experiments was collected during several user sessions in which the performance of this dialogue system was evaluated. The subject utterances were recorded as 8kHZ and 16kHz waveform files and the entire information state and candidate attachements were logged. There were two domains tested (an MP3 application and a Restaurant database selection) for a total of almost 400 user utterances.

For each user utterance only a single speech recognition is available, but this produces an n-best list of parses, and for each one of them multiple dialog move candidates are considered. The objective of the attachment algorithm is to select the best possible dialog move among all the candidates produced by all the parses in the n-best list.

5. Results

The baseline system (B) simply takes the first (most likely) parse and selects the attachment with the most recent parent node in the DMT. The default attachment algorithm (D) is based on hand-crafted techniques (described in section 3).

The machine learning approach (ML-Reorder) uses classification obtained from the memory based learner TiMBL. For each potential candidate we associated a vector of features (the ones from above) and also the supervised classification class (correct/incorrect). So for each user utterance (*U*) we'll have many candidate moves *{M1, M2... Mk}*, all of which are incorrect except for the one we labeled as the correct one. We always have one correct example – the best move based on the list of parses (we allow attachment of error nodes if there is nothing else better), so this is done regardless of correctness of the speech recognition.

	Baseline (B)		Default (D)		ML-Reorder		ML-Reorder'	
	correct	incorrect	Correct	incorrect	Correct	incorrect	Correct	incorrect
MP3	126	84	165	45	101	109	174	36
	59.04%		78.57%		48.09%		82.85%	
Restaurant	111	74	153	32	114	71	166	19
	62.16%		82.70%		61.62%		89.72%	

The performance of ML-Reorder was on average worse than that of the baseline system. This might be explained by the fact that only one candidate move in *{M1, M2... Mk}* is selected, while there might be several "almost optimal" ones, with feature vectors very similar to the features of the one that should be selected, but which have incorrect labels. This problem suggested a different representation of the training data. For each training example (user utterance) we only consider the 5 best candidates produced by the default D system. A vector of features consists of the set of features of the 5 five candidates (simply concatenated) and the labeling class is the number of the instance (among these 5) that is the correct one. In all cases the correct dialog move was among these 5 best candidates and in cases in which we had less than 5 candidates we simply filled the features vector with zeros. We call this system ML-Reorder' and the performance (best among all the systems) is reported in the last column of the table above. Since we have limited data, a leave-one-out crossvaliadation setup was used here.

Performance gain over the baseline comes from the fact that both D and ML-Reorder' manage to solve some parsing problems and re-order the n-best list of parses. Most frequent ones were PP-attachments (a difficult challenge for statistical parses) and nominal modifiers:

PP – attachment:
1. how about [a restaurant [in Grant]] [on Mayfield]
2. how about [a restaurant [in Grant] [on Mayfield]]

Nominal modifier:
1. how about [a [[cheap] chinese] restaurant]
2. how about [a [cheap] [chinese] restaurant]

6. Conclusion and Future Work

The results so far seem quite encouraging. Both D and ML-Reorder' perform significantly better than the baseline, while ML-Reorder' obtains increases of 5.45% and 8.49% over D in the two domains. Performance in the restaurant domain is better because topic classification is only available here and this tends to be an important feature.
The data sets used in these examples were rather small and other learners might be tested. A difficult task so far (both for D and ML-Reorder') was disambiguating between revision dialog-moves (nodes that should attach to other previous user nodes) and nodes that should create a new dialog thread. So far this problem was solved in the broader context of generally selecting the best dialog move from the entire set of candidates. If we restrict to training examples relevant to this task we should be able to apply an ML-Reorder algorithm efficiently, without having to increase the features vector size.

References:

1. Chotimongkol and A. Rudnicky. 2001. Nbest speech hypotheses reordering using linear regression. In Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH).
2. M. Gabsdil and O. Lemon. 2004. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In Proc. 42nd Annual Meeting of the ACL.
3. C. Manning, H. Schutze. 1999. Foundations of Statistical Natural Language Processing.
4. D. Mirkovic and L. Cavedon. 2005. Practical plug-and-play dialogue management. In Proceedings of the Annual Meeting of the Pacific Association of Computational Linguistics (PACLING).
5. L. Cavedon, M. Purver, F. Ratiu. 2005. Combining Confidence Scores with Contextual Features for Robust Multi-Device Dialogue.