

Learning-based Semantic Interpreter in a Dialog System

Junling Hu, Fabrizio Morbini, Feng Lin and Fuliang Weng

Robert Bosch Research and Technology Center

4009 Miranda Ave.

Palo Alto, CA 94304

{Junling.hu, Fabrizio.Morbini, feng.lin, Fuliang.weng}@us.bosch.com

Abstract

This paper investigated a learning-based approach to semantic interpretation. We replaced a script-based interpreter with an adaptive design. The new design involves a Maximum Entropy classifier for classifying dialog moves and an NP extraction component to get the semantic slots. We compare the performance of our classifier with two baseline approaches, one is the original script-based approach, and the other adapted from information-retrieval method. We found significant performance improvement with the learning-based method. We also show good performance by our NP extractor. Our research demonstrates the feasibility of creating a scalable and adaptive semantic interpreter for dialog systems.

1 Introduction

Semantic interpretation in a dialog system involves two key tasks: identifying dialog acts and extracting semantic information from user utterances. These tasks are normally broken down in two steps: A semantic parser first identifies semantic slots, and then the utterance is mapped to a dialog act (He and Young, 2003). Such approaches are suitable for template-driven dialogs, but not sufficient for more sophisticated dialogs, where we also need to resolve anaphora, numeric constraints and so on.

A related work (Ang et al, 2005; Ji and Bilmes, 2006) has attempted to automatically classify sentences into dialog acts. However, such ap-

proaches are not commonly found in practical dialog systems.

We are interested in building an integrated solution that not only automates dialog act classification but also extracts semantic information including anaphora and other things. We implemented our design in a practical dialog system described in (Weng et al, 2006). Our work involves replacing a script-based dialog move classifier with a maximum entropy (ME) classifier, and creating a noun-phrases (NP) extractor that extracts key semantic slots and resolves anaphora.

We show that our learning component performs significantly better than the original script-based system. In addition, we compare our performance to another baseline approach based on information-retrieval method. We show that the ME classifier performs much better than the information-retrieval method too.

Our NP extractor uses results from parser and the dialog-move classifier to extract the key slots. We showed that it achieved 94% accuracy of identifying the right entities (when compared to the current script based extraction).

Our work is motivated by the fact that today's practical dialog systems still rely on hand-written scripts for identifying user moves. Such an approach limits the coverage of the system, as the designer cannot foresee all possible ways a user might say about things. An integrated solution that combines learning-based classifier for dialog acts and semantic-slot extraction will remove this limitation. Eventually our learning-based classifier can be re-trained with new data, without much human

intervention. This paves the way to automatic coverage extension for dialog systems.

2 Background

In this section, we give an overview of our dialog system.

2.1 System Overview

Our work is based on the CHAT (Conversational Helper for Automotive Task) dialog system. It is an in-car spoken dialog system that supports users on tasks such as finding a local restaurant, getting driving directions or playing songs. The basic architecture of this system is shown in Figure 1.

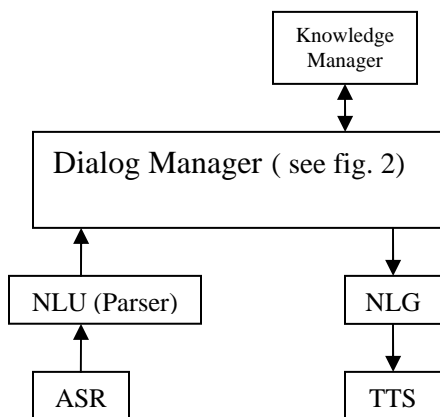


Figure 1: Overview of the Dialog System

The dialog manager of this system is based on information-state update approach (Larsson & Traum, 2000). It uses a tree-based representation of dialogue context (Lemon & Gruenstein, 2004), which includes anaphoric referents and propositional information as well as a structured dialogue move history. Moves are interpreted in context by attachment to an appropriate open node, with corresponding rules updating the context. The detail of the dialog manager is shown in Figure 2.

A Dialog Move Tree (Lemon & Gruenstein, 2004, Mirkovic and Cavedon, 2005) is used to track dialogue states. It remembers previous conversations, controls the next move, and generates the appropriate output. The dialog manager actively maintains a dialog move tree by adding the current nodes and pruning nodes on finished conversation. This dialog move tree persists through different rounds of communication, and is created when the

system starts. The initial tree has a root and its expected nodes. A root is associated with a domain. For example, if we are in a restaurant domain, the dialog move tree will start with a root node that expects restaurant-domain nodes.

Each node on the tree is called a dialog move. There are 2 types of dialog moves: user moves and system moves. Examples of user nodes are:

- Meta-question (like “what can I say?”),
- Query on restaurant property (“Is there an Italian restaurant nearby?”),
- Query on how many restaurants,
- Unconfirmed (needs further confirmation),
- Error (system cannot understand).

The patterns of these nodes are defined in dialog-move scripts.

2.2 Dialog-move Scripts

Dialog-move script (Mirkovic and Cavedon, 2005) defines dialog moves, each of which has the following components:

- (1) Input patterns which contains variables corresponding to certain semantic slots
- (2) Output sentences or patterns of sentences
- (3) Expected nodes.

The scripts are different for different domains. Figure 3 shows one entry in dialog-move script in a restaurant domain. For simplicity just one pattern has been left in it. That pattern will match sentences like “What city is this restaurant in?” or “What city is the first one in?”. The system could respond with a disambiguation question like “what do you mean by ‘the first one’” (by producing the system node WHQuestion:disambig:anaphora), or with the final answer “Restaurant1 is in City1” (by

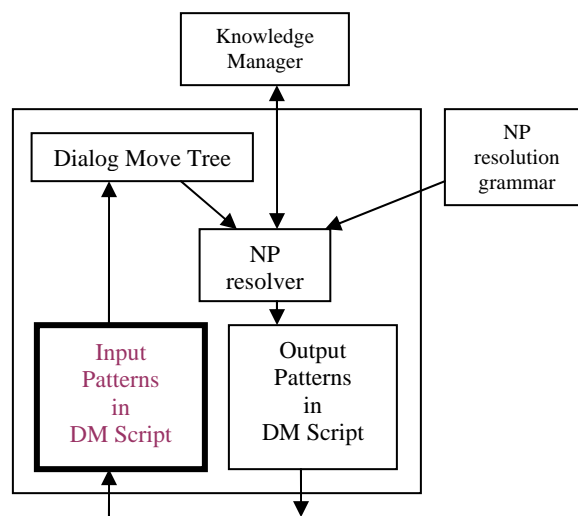


Figure 2: The Dialog Manager

using the directly specified System node ObjectQueryAnswer:property:city.

```

User ObjectQuery:property:city {
  Input {
    "SYN(s{predicate{#is/vbz|#ppps/vbz},
      arglist{subj:OBJECT,
        wcomp{#what/wdt,?nmod{#city/nn}}},
      adjunctlist{*{#in}}})"
    ...
  }
  Producing {
    WHQuestion:disambig:anaphora
    ErrorResponse:anaphora
    System ObjectQueryAnswer:property:city{
      Output {"[object] is [city]"}
    }
  }
  Expecting {Revision:property}
}

```

Figure 3: Example of a node in script.

Dialog-move scripts fulfill the following functions:

- (1) It maps a user input to Dialog move node
- (2) It supports NP grounding
- (3) It defines system output
- (4) It defines next move.

3 Replacing Scripts with Scalable design

Through scripts, the system can capture patterns of sentences. However, these scripts are manually created, and are limited in scope and coverage. In general, scripts have the following advantage:

- (1) Easy to write for small domain.
- (2) Concise and compact representation.

But are also afflicted by the following disadvantages:

- (1) Domain dependent
- (2) Language dependent
- (3) Hard to extend (Relying on experts who understand language patterns).

Our goal is to have a domain-independent and easy-to-extend method to identify user inputs. The best way to do this is removing the need for scripts, and replacing it with a learning-based classifier. Such a classifier can be trained offline and can be re-trained when the system gathers more data.

However, the dialog scripts in the current system fulfill multiple functions. Not only they define input patterns, they also define output patterns. We

need to de-couple these functions before we can replace the input scripts with other ways to process inputs.

3.1 A Scalable Design

We convert the original dialog-move scripts, written in a single file, to a database with several tables. The migration from a file to a database gives us the following advantages:

- 1) The different functions of the scripts are completely separated. In the database, there is an input table for all input pattern, an output table for all outputs, and an expecting table for expecting nodes. This makes it possible to suppress the use of input patterns while we still use the output patterns.
- 2) The new design is much more scalable. This is reflected in both retrieving data and modifying data.
 - a. Retrieving data. When scripts are written in a file, the system has to load the whole file in the beginning, and builds internal data structure to store the entries in the file. This is doable for a small file, but will become infeasible when the size of the file grows. For the database, there is no need to load everything into memory since a query is done in runtime.
 - b. Modifying data. If we want to add a node in a file, we need to write it in special syntax, then load the entire file into memory and overwrite it with the expanded definition. In a database, such modification is simply appending a row to a table, which is a trivial operation. The node structure is made uniform in a database table, thus no more special syntax to learn.

3.2 Two New Components

Given the limitation of scripts for identifying user moves, we replace them with a learning-based classifier. In doing so, we need to fulfill two functions previously provided by input scripts: (1) identifying dialog moves; (2) extracting noun-phrases (NP) from user utterance to fill the required slots.

To identify user moves, we use a statistical classifier. A statistical classifier has the following advantage over scripts:

- (1) It is domain independent. The same kind of features and training model can be used in different domains.
- (2) It is less language dependent. For example, the first two words are important features for many different languages.
- (3) There is no need to write sentence patterns any more. The system can be re-trained with new data automatically. The only human effort in re-training is labeling sentences with certain dialog moves. This effort is significantly smaller than writing input patterns for every sentence together with labeling a dialog move for that sentence.

We adopt Maximum Entropy classifier (Berger et al, 1996) for classifying dialog moves. This method has been shown to work well for a wide range of NLP tasks (Ratnaparkhi, 1998).

The second component we create is an NP extractor that extracts NPs from user input. It extracts key information such as topic information, time indicator and anaphora. These NPs are used to build the constraints that will form the query the knowledge base to retrieve the data needed to answer the user question. The NPs that contain anaphora will go through a resolution module before being used to form constraints.

3.3 Maximum Entropy Classifier

We use the openNLP package¹ for building our maximum entropy classifier. Our training data are recorded from real users. For every user utterance, we label its dialog move by hand. This is the only hand-labeling work we do. We do not need to annotate the sentence as we can get all the features automatically.

The features used for our classifier are shown in Table 1. These features are extracted automatically from the output of our syntactic and semantic parsers. The syntactic parser returns parse tree with part-of-speech and other syntactic information. Our semantic parser returns speech acts and semantic slots. Here “speech act” is a broader category than dialog move (dialog act). For example, a speech act of “Query” can correspond to several dialog

moves: “Query: restaurant”, “Query: Location”, and “Query: Property” etc.

Speech act	whquery, command/request, confirmation, rejection, etc.
Predicate	The top level predicate in the utterance.
Tense	The tense of the top level predicate.
First 2 words	The concatenation of the first 2 words after disfluency elimination.
Semantic slots	A set of 13 binary (i.e. NIL/not-NIL). E.g.: cuisine name, city name, restaurant name, price level.

Table 1: Features used in the ME classifier.

3.4 The NP Extractor

Without expert-written scripts, we designed a way to extract NPs directly from the user utterance. Our method takes advantage of the syntactic structure of the user sentence and the dialog move that this sentence is mapped to.

The NPs we want to extract are:

- (1) Conversational topics, such as “Italian restaurant”, “nearby hotel”, etc.
- (2) Time indicator, such as “today”, “now”, etc.
- (3) Anaphoric phrases such as “this”, “that” and “last one”.

For each sentence, we process its parse tree and extract basic components including subject, object and predicate. Then we look at the dialog move this sentence is classified into. If the dialog move is part of ObjectQuery:property nodes, we extract the subject. We also retain a variable which holds semantic slots given by the semantic parser as a backup.

4 Baseline Methods for Classifying Dialog Moves

We compare the performance of our maximum entropy classifier with two baseline approaches. One is the script-based method, and the other is based on an information-retrieval method.

¹ See <http://maxent.sourceforge.net/>

4.1 Baseline #1: Script-based method

This is the original approach. It has many hand-written patterns associated to dialog moves, and stores them in a script file. An input sentence is processed by the parser, and the parsed result is matched to these patterns. If there are matches (potentially more than one), the dialog moves associated with that pattern will be selected.

For example, if the input sentence is “What city is this restaurant in?” It goes through the process shown in Figure 4. The patterns are written as regular expressions that match particular classes of syntactic trees.

4.2 Baseline #2: Information-Retrieval method

Information retrieval (IR) methods are typically used to compare a user’s query to a large collection of documents, and return a ranked list of documents in order of similarity/relevance with the user query.

Consider all sentences corresponding to a particular dialog move as one document. For any new user sentence, we want to see which of the existing groups of sentences best matches this new utterance. The node corresponding to the group that has highest matching score would be the dialog move that is most relevant to that utterance.

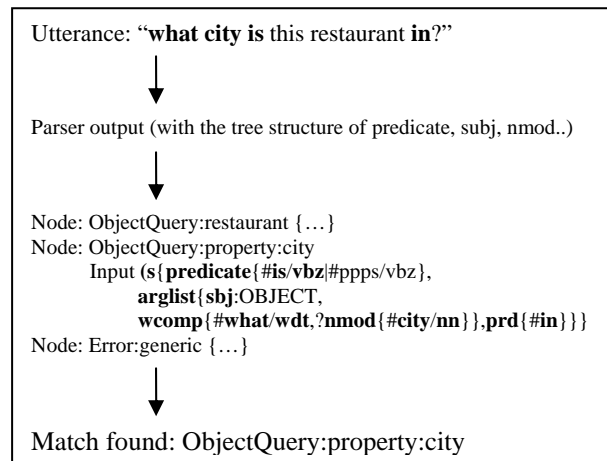


Figure 4: Example of script-based match.

Therefore we treat each new sentence as a query, and each dialog move node as a document. The

content of these documents are the sentences in our training data.

We use cosine similarity (Manning et al, 2007) to measure the similarity between a query (q) and a document (d).

$$sim(q, d) = \cos(\theta) = \frac{V(q) \bullet V(d)}{\|V(q)\| \cdot \|V(d)\|}$$

Where $V(q)$ is the weighted term vector for query q and $V(d)$ is the weighted term vector for document d . $V(q) \bullet V(d)$ is the inner product of these two vectors, and $\|V(q)\| = \sqrt{V(q) \bullet V(q)}$.

The crucial step here is building the term vector. We decide to include every word in the sentences along with its Part-of-Speech (POS) tag from the parser. Furthermore, we decided to include all the features used by the ME method to make the performances of the 2 methods more easily comparable. In summary, the following terms are in the term vector:

- Word-POS (e.g. restaurant-noun)
- All features from ME method (speech act, tense, predicate, semantic slot, ...)

For example, the sentence “Are there any Chinese restaurant” generates the following term vector:

```

<are-vbp, any-determiner, Chinese-Proper Name,
restaurant-noun;
predicate-are;
tense-present,
speechact-Query;
cuisine_name-notnil, service_level-nil, ...
>
  
```

The weight of each term is given by *tf-idf* (term frequency-inverse document frequency) score. The *tf-idf* of term t in document k is $tf_t(k) \cdot idf_t$, where

$$tf_t(k) = \frac{\# \text{ of occurrences of } t \text{ in document } k}{\# \text{ of occurrences of all terms in document } k}$$

$$idf_t = \log \left(\frac{\# \text{ of documents}}{\# \text{ of documents containing } t} \right)$$

Each document has a vector of equal size, with elements corresponding to the *tf-idf* scores of the term vector. We calculate the cosine similarity between the input sentence and the document that represents a dialog move.

5 Experimental Results

5.1 Training Data

We used a total of 1668 sentences collected from real users. These sentences were classified into different nodes by hand.

We found that the sentences are not equally distributed among the various nodes. The most commonly used nodes are ObjectQuery:restaurant and Revision:restaurant. Each has more than 800 sentence instances. These two nodes are used for sentences asking the system to find restaurants with certain properties.

We do not annotate the POS and any other feature of the sentences. All these are generated automatically by our parser. Therefore the hand-labeling work is limited to deciding the set of nodes an utterance could be mapped to.

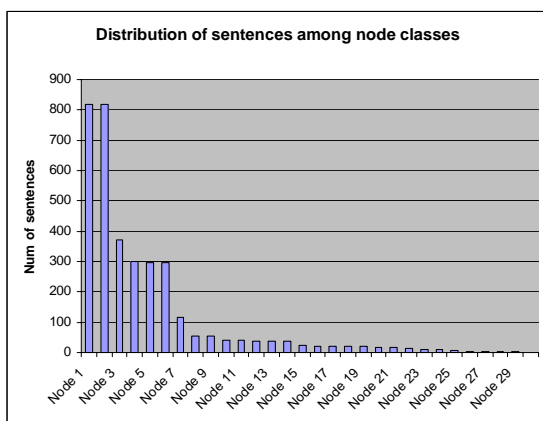


Figure 3. Distribution of sentences among node Classes

5.2 Experimental Results of the Classifier

We compare the performance of our maximum entropy classifier with two baseline methods. We applied 10-fold cross validation to test the performance. The average results of 10 tests are shown in Table 2.

Methods	Accuracy (%)	
	1-best result	2-best result
Max Entropy	90	92
Baseline #1 (Script-based)	71	
Baseline #2 (IR)	72	81

Table 2. Performance Results

For maximum entropy (ME) classifier, the “1-best result” refers to the top node with the highest probability that matches to any of the nodes in the hand-label. The “2-best result” refers to the top 2 nodes with the highest probability. For script-based method, there is no well-defined ranking of the returned matches. Therefore, for the IR approach, we count as a successful result one in which the returned nodes intersect the nodes in the hand-label.

As we can see, the classifier outperforms both baseline methods significantly, particularly for 1-best result. It results in 20% difference in accuracy from both IR approach and script-based approach.

5.3 Experimental Results of the NP Extractor

We evaluated our NP extractor on the same set of sentences used to train the ME classifier. For each sentence we run the pattern matcher and keep the matches that are in accordance with the hand label. Each match determined by a pattern instantiates the variables defined in the pattern itself. We then run the NP extraction algorithm as defined above and compare the two assignments. If all the variables assigned by the pattern matcher have been properly assigned also by our NP extraction method we count the match as positive otherwise as negative.

Our experiments show that, this NP extractor correctly identifies 94% of the NPs. In total, the 1668 sentences produced 3181 NPs of which 94% are identified by our NP extractor.

Our NP extractor is heuristically designed. Further work needs to be done to generalize it and make it more robust.

6 Conclusions

This paper investigated a learning-based approach to semantic interpretation. We replaced a script-based interpreter with an adaptive design. The new design involves a Maximum Entropy classifier for classifying dialog moves and an NP extraction component to get the semantic information.

In the process, we created a scalable design that migrate a system from script-based to learning-based method for identifying dialog moves.

We compare the performance of our classifier with two baseline approaches, one is the original script-based approach, and the other derived from information-retrieval methods. We found significant performance improvement with the learning-based method (92% accuracy). We also show good performance (94% accuracy) by our NP extractor. Our research demonstrates the feasibility of creating a scalable and adaptive semantic interpreter for dialog systems.

Acknowledgment

We thank Lenhart Shubert and Zhe Feng for helpful discussions.

References

- Ang, J., Liu, Y., Shriberg, E.: "Automatic Dialog Act Segmentation and Classification in Multiparty Meetings," in *Proc. ICASSP-2005*, Philadelphia, 2005
- Adam L. Berger, Vincent J. Della Pietra and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguist.* 22, 1 (Mar. 1996), 39-71.
- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2007.
- Dan Bohus, B. Langner., A. Raux, Allen Black, M. Eskenazi and Alex Rudnicky, 2006. [Online Supervised Learning of Non-understanding Recovery Policies](#), in *SLT-2006*, Palm Beach, Aruba
- Y. He and Steve Young. (2003). A data-driven spoken language understanding system. *IEEE Workshop on Automatic Speech Recognition and Understanding*. US Virgin Islands.
- Gang Ji and Jeff Bilmes Backoff Model, Training using Partially Observed Data: Application to Dialog Act Tagging in *Proceedings of Human Language Technology Conference of the North American chapter of the Association for Computational Linguistics(HLT-NAACL'06)* , New York, NY. June, 2006
- Staffan Larsson and David Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3-4), 2000.
- O Lemon and A Gruenstein, "Multithreaded context for robust conversational interfaces," *ACM Transactions on Computer-Human Interaction*, vol. 11, no. 3, 2004.
- D Mirkovic and L Cavedon, "Practical plug-and-play dialogue management," in *Proc. PACLING*, Tokyo, 2005.
- Adwait Ratnaparkhi, 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. Thesis, University of Pennsylvania.
- Fuliang Weng, Sebastian Varges, Badri Raghunathan, Florin Ratiu, Heather Pon-Barry, Brian Lathrop, Qi Zhang, Tobias Scheideck, Harry Bratt, Kui Xu, Matthew Purver, Rohit Mishra, Madhuri Raya, Stanley Peters, Yao Meng, Lawrence Cavedon and Liz Shriberg. 2006. *CHAT: A Conversational Helper for Automotive Tasks*. Proceedings of the 9th International Conference on Spoken Language Processing ([Interspeech/ICSLP](#)).