Cooperative Model Based Language Understanding in Dialogue

Donghui Feng
Information Sciences Institute,
University of Southern California
4676 Admiralty Way
Marina Del Rey, CA 90292-6695
donghui@isi.edu

Abstract

In this paper, we propose a novel Cooperative Model for natural language understanding in a dialogue system. We build this based on both Finite State Model (FSM) and Statistical Learning Model (SLM). FSM provides two strategies for language understanding and have a high accuracy but little robustness and flexibility. Statistical approach is much more robust but less accurate. Cooperative Model incorporates all the three strategies together and thus can suppress all the shortcomings of different strategies and has all the advantages of the three strategies.

1 Introduction

In this paper, we propose a novel language understanding approach, Cooperative Model, for a dialogue system. It combines both Finite State Model and Statistical Learning Model for sentence interpretation.

This approach is implemented in the project MRE (Mission Rehearsal Exercise). The goal of MRE is to provide an immersive learning environment in which army trainees experience the sights, sounds and circumstances they will encounter in real-world scenarios (Swartout et al., 2001). In the whole procedure, language processing part plays the role to support the communication between trainees and computers.

In the language processing pipeline, audio signals are first transformed into natural language sentences by speech recognition. Sentence interpretation part is used to "understand" the sentence and extract an information case frame for future processing such as dialogue management and action planning. We adopt the Cooperative Model as the overall frame of sentence interpretation, which incorporates two mainly used language processing approaches: the Finite State Model and the Statistical Learning Model. Currently there is relatively little work on the cooperation of the two kinds of models for language understanding.

The Cooperative Model has great advantages. It balances the shortcomings of each separate model. It is easy to implement the parsing algorithm and get the exact expected result for finite state model (FSM) but it's difficult and tedious to design the finite state network by hand. Also, the finite state model is not too robust and the failure of matching produces no results. On the other hand, statistical learning model (SLM) can deal with unexpected cases during designing and training by giving a set of candidate results with confidence scores. It is a must to provide some kind of rules to select results needed. However, applying it may not give a completely satisfactory performance.

The rest of this paper is organized as follows: Section 2 describes the case frame as the semantic representation produced by the cooperative model. In section 3, we explain our cooperative language understanding model and discuss two different strategies of the Finite State Model and the Statistical Learning Model. We analyze the experimental results in Section 4. Section 5 concludes with on-going research and future work.

2 Semantic Representation

The goal of automated natural language understanding is to parse natural language string, extract meaningful information and store them for future processing. For our application of training environment, it's impossible to parse sentences syntactically and we here directly produce the nested information frames as output. The topmost level of the information frame is defined as follows:

```
<i-form> := ( ^mood <mood>
              ^sem <semantic-object>)
```

Figure 1. Topmost-Level Information Frame

In the definition, <semantic-object> consists of three types: *question*, *action* and *proposition*. Here, question refers to requests for information, action refers to orders and suggestions except requests, and all the rest falls into the category of proposition.

Each of these types can also be further decomposed as Figure 2 and 3.

```
<question> := (   ^type question
                   ^q-slot <prop-slot-name>
                   ^prop <proposition>)

<action>    := (   ^type action-type
                   ^name <event-name>
                   ^<prop-slot-name> <val>)

<proposition> := <state> | <event> | <relation>
```

Figure 2. Second-Level Information Frame

```
<state> :=    (    ^type state
                   ^object-id ID
                   ^polarity <pol>
                   …)

<event>    := (   ^type event-type
                   ^name <event-name>
                   ^<prop-slot-name> <val>
                   …)

<relation> := (    ^type relation
                   ^relation <rel-name>
                   ^arg1 <semantic-object>
                   ^arg2 <semantic-object>)
```

Figure 3. Third-Level Information Frame

These information frames can be further extended and nested as necessary. In our application, most of the information frames obtained contain at most three levels. In Figure 4, we give an example of information frame for the English sentence "who is not critically hurt?". All the target information frames in our domain are similar to that format.

```
Input Sentence:    who is not critically hurt?
Output Information Frame:
(<i>        ^mood interrogative
            ^sem <t0>)
(<t0>       ^type question
            ^q-slot agent
            ^prop <t1>)
(<t1>       ^type event-type
            ^time present
            ^polarity negative
            ^degree critical-injuries
            ^attribute health-status
            ^value health-bad)
```

Figure 4. Example Nested Information Frame

Since the information frames are nested, for the statistical learning model to be addressed, ideally both the semantic information and structural information should be represented correctly. Therefore we use prefix strings to represent the cascading level of each slot-value pair. The case frame in Figure 4 can be re-represented as shown in Figure 5. Here we assume that the slots in the information frame are independent of each other. Reversely the set of meaning items can be restored to a normal nested information frame.

```
<i> ^mood interrogative
<i> ^sem <t0>
<i> <t0> ^type question
<i> <t0> ^q-slot agent
<i> <t0> ^prop <t1>
<i> <t0> <t1> ^type event-type
<i> <t0> <t1> ^time present
<i> <t0> <t1> ^polarity negative
<i> <t0> <t1> ^degree critical-injuries
<i> <t0> <t1> ^attribute health-status
<i> <t0> <t1> ^value health-bad
```

Figure 5. Re-representation to handle cascading

We introduce the cooperative model in the following section to extract meaningful information frames for all the English sentences in our domain.

3 Cooperative Model

The Cooperative Model (CM) combines two commonly-used methods in natural language processing, Finite State Model (FSM) and Statistical Learning Model (SLM). We discuss them in section 3.1 and 3.2 respectively.

3.1 Finite State Model

The main idea of finite state model is to put all the possible input word sequences and their related output information on the arcs.

For our application, the input is a string composed of a sequence of words, and the output should be a correctly structured information frame. We apply two strategies of FSM. The Series Mode refers to build a series of finite state machine with each corresponding to a single slot. The Single Model builds only one complex Finite State Machine that incorporates all the sentence patterns and slot-value pairs.

3.1.1 Strategy I: Series Model of Finite State Machine

For this strategy, we analyze our domain to obtain a list of all possible slots. From the perspective of linguistics, a slot can be viewed as characterized by some specific words, say, a set of feature words. We therefore can make a separate semantic filter for each slot. Each sentence passes through a series of filters and as soon as we find the "feature" words, we extract their corresponding slot-value pairs. All the slot-value pairs extracted produce the final nested case frame.

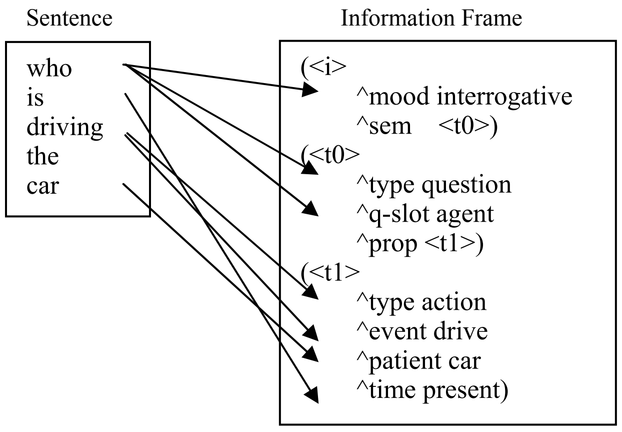

Figure 6. An Example from Series Model of FSM

Figure 6 is an example of the way that series model of finite state machine works. For example, three slot-value pairs are extracted from the word "who". Practically, we identified 27 contexts and built 27 finite state machines as semantic filters, with each one associated with a set of feature words. The number of arcs for each finite state machine ranges from 4 to 70 and the size of the feature word set varies from 10 to 50.

This strategy extracts semantic information based on the mapping between words and slots. It is relatively easy to design the finite state machine networks and implement the parsing algorithm. For every input sentence it will provide all possible information using the predefined mappings. Even if the sentence contains no feature words, the system will end gracefully with an empty frame. However, this method doesn't take into account the patterns of word sequences. Single word may have different meanings under different situations. In most cases it is also difficult to put one word into one single class; sometimes a word can even belong to different slots' feature word sets that can contradict each other. On the other hand, the result produced may have some important slot-value pairs missed and the number of slots is fixed.

3.1.2 Strategy II: Single Model of Finite State Machine

In this strategy we only build a big finite state network. When a new sentence goes into the big FSM parser, it starts from "START" state and a successful matching of prespecified patterns or words will move forward to another state. Any matching procedure coming to the "END" state means a successful parsing of the whole sentence. And all the outputs on the arcs along the path compose the final parsing result. If no patterns or words are successfully matched at some point, the parser will die and return failure.

This strategy requires all the patterns to be processed with this finite state model available before designing the finite state network. The target sentence set includes 65 sentence patterns and 23 classes of words and we combine them into a complex finite state network manually. Figure 7 gives some examples of the collected sentence patterns and word classes.

```
$phrase1 = what is $agent doing;
$phrase2 = [and|how about] (you|me|[the]
                $vehicle|$agent);
…

$agent = he|she|$people-name|[the] ($person_civ |
            $person_mil| $squad);
$vehicle = ambulance | car | humvee | helicopter
             |medevac;
…
```

Figure 7. Target Sentence Patterns

Aimed at processing these sentences, we design our finite state network consisting of 128 states. This network covers more than 20k commonly-used sentences in our domain. It will return the exact parsing result without missing any important information. If all of the input sentences in the application belong to the target sentence set of this domain, this approach perfectly produces all of the correct results. However, the design of the network is done totally by hand, which is very tedious and time-consuming. The system is not very flexible or robust and it's difficult to add new sentences into the network before a thorough investigation of the whole finite state network. It is not convenient and efficient for extension and maintenance.

Finite state models can't process any sentence with new sentence patterns. However in reality most systems require more flexibility, robustness, and more powerful processing abilities on unexpected sentences. The statistical machine learning model gives us some light on that. We discuss learning models in Section 3.2.

3.2 Statistical Learning Model

3.2.1 Naïve Bayes Learning

Naïve Bayes learning has been widely used in natural language processing with good results such as statistical syntactic parsing (Collins, 1997; Charniak, 1997), hidden language understanding (Miller et al., 1994).

We represent the mappings between words and their potential associated meanings (meaning items including level information and slot-value pairs) with P(M|W). W refers to words and M refers to meaning items. With Bayes' theorem, we have the formula 3.1.

$$\arg\max P(M \mid W) = \arg\max \frac{P(W \mid M) * P(M)}{P(W)} \qquad (3.1)$$

Here P(W|M) refers to the probability of words given their meanings.

In our domain, we can view P (W) as a constant and transform Formula 3.1 to Formula 3.2 as follows:

$$\arg\max_m P(M \mid W) = \arg\max_m P(W \mid M) * P(M) \qquad (3.2)$$

3.2.2 Training Set and Testing Set

We created the training sentences and case frames by running full range of variation on Finite State Machine described in Section 3.1.2. This gives a set of 20, 677 sentences. We remove ungrammatical sentences and have 16,469 left. Randomly we take 7/8 of that as the training set and 1/8 as the testing set.

3.2.3 Meaning Model

The meaning model P(M) refers to the probability of meanings. In our application, meanings are represented by meaning items. We assume each meaning item is independent of each other at this point. In the meaning model, the meaning item not only includes slot-value pairs but level information. Let $C(m_i)$ be the number of times the meaning item m_i appears the training set, we obtain P(M) as follows:

$$P(m_i) = \frac{C(m_i)}{\sum_{j=1}^{n} C(m_j)} \qquad (3.3)$$

This can be easily obtained by counting all the meaning items of all the information frames in the training set.

3.2.4 Word Model

In the naïve Bayes learning approach, P(W|M) stands for the probability of words appearing under given meanings. And from the linguistic perspective, the patterns of word sequences can imply strong information of meanings. We introduce a language model based on a Hidden Markov Model (HMM). The word model can be described as $P(w_i \mid m_j, w_{i-2}w_{i-1})$, $P(w_i \mid m_j, w_{i-1})$ or $P(w_i \mid m_j)$ for trigram model, bigram model, and unigram model respectively. They can be calculated with the following formulas:

$$P(w_i \mid m_j, w_{i-2}w_{i-1}) = \frac{\#of(m_j, w_{i-2}w_{i-1}w_i)}{\#of(m_j, w_{i-2}w_{i-1})} \qquad (3.4)$$

$$P(w_i \mid m_j, w_{i-1}) = \frac{\#of(m_j, w_{i-1}w_i)}{\#of(m_j, w_{i-1})} \qquad (3.5)$$

$$P(w_i \mid m_j) = \frac{\#of(m_j, w_i)}{\#of(m_j)} \qquad (3.6)$$

3.2.5 Weighted Sum Voting and Pruning

We parse each sentence based on the naïve Bayes learning Formula 3.2. Each word in the sentence can be associated with a set of candidate meaning items. Then we normalize each candidate set of meaning items and use the voting schema to get the final result set with a probability for each meaning item.

However, this inevitably produces noisy results. Sometimes the meanings obtained even contradict other useful meaning items. We employ two cutoff strategies to eliminate such noise. The first is to cut off unsatisfactory meaning items based on a gap in probability. The degree of jump can be defined with an arbitrary threshold value. The second is to group all the slot-value pairs with the same name and take the top one as the result.

3.3 Cooperative Mechanism

In the previous two sections, we discussed two approaches in our natural language understanding system. However, neither is completely satisfactory.

Cooperative Model can combine all three approaches from these two models. The main idea is to run the three parsing models together whenever a new sentence comes into the system. With the statistical learning model, we obtain a set of information frames. For the result we get from single model of finite state machine, if an information frame exists, it means the sentence is stored in the finite state network. We therefore assign a score 1.0. The result should be no worse than any information frame we get from statistical learning model. Otherwise, it means this sentence is not stored in our finite state work, we can ignore this result. In the end, we combine this information frame with the frame set from statistical learning model and rank them according to the confidence scores. Generally we can consider the one with the highest confidence score as our parsing result.

The cooperative model takes all advantages of the three methods and combines them together. The cooperative mechanism also suppresses the disadvantages of those methods. The series model of the finite state machine has the advantage of mapping between word classes and contexts, though it sometimes may lose some information, and it contains real semantic knowledge. The statistical learning model can produce a set of information frames based on the word patterns and its noise can be removed by the result of the series model of the finite state machine. For the single finite state machine model, if it can parse sentence successfully, the result will always be the best one. Therefore through the cooperation of the three methods, it can either produce the exact result for sentences stored in the finite state network or return the most probable result through statistical machine learning method if no sentence matching occurs. Also the noise is reduced by the other finite state machine model. The cooperative model is robust and has the ability to learn in our target domain.

4 Experimental Results

The cooperative model will demonstrate its ability on sentence processing no matter whether the sentence is in the original sentence set. However, currently we only have simple preference rule for the cooperation and haven't obtained the overall performance. In this section, we'll compare the different models' performance to demonstrate the cooperative model's potential ability.

Based on our target sentence patterns and word classes, we built a blind set with 159 completely new sentences. Although all the words used belong to this domain these sentences don't appear in the training set and the testing set. In the evaluation of its performance, we compare the results of the three approaches and get Table 1. As we can see from this table, finite state method is better in the relative processing speed and for processing existing patterns while statistical model is better for processing new sentence patterns, which makes the system very robust.

	Sentences in Domain	Speed	Existing Patterns	New Patterns
Series of FSM	Fixed	Fast	100%	Partial Result
Single FSM	Fixed	Fast	100%	Die
Stat Model	Open	Slow	85%(pre) 95%(rec)	75%(pre) 92%(rec)

Table 1. Results Comparison

On the other hand, we investigate the performance of statistical model in more detail on the blind test. Given the whole blind testing set, the statistical learning model produced 159 partially correct information frames. We manually corrected them one by one. This took us 97 minutes in total. To measure this efficiency, we also built all the real information frames for the blind test set manually, one by one. It took 366 minutes to finish all the 159 information frames. This means it is much more efficient to process a completely new sentence set with the statistical learning model.

We next investigate the precision and recall of this statistical learning model. Taking the result frames we manually built as the real answers, we define precision, recall, and F-score to measure the system's performance.

$$\text{precsion} = \frac{\#\,\text{of correct slot - value pairs}}{\#\,\text{of slot - value pairs from learning model}}$$

$$\text{recall} = \frac{\#\,\text{of correct slot - value pairs}}{\#\,\text{of slot - value pairs from real answer}}$$

$$\text{F\_Score} = \frac{2 * (\text{precision} * \text{recall})}{\text{precision} + \text{recall}}$$

Our testing strategy is to randomly select some portion of the new blind set and add it into the training set. Then we test the system with sentences in the rest of the blind set. As more and more new sentences are added into the training set (1/4, 1/3, 1/2, etc) we can see the performance changing accordingly. We investigate the three models: P(M|W), P(W|M) and P(M)*P(M|W). All of them are tested with same testing strategy.

Portion	0	1/4	1/3	1/2	2/3
Prec	0.7131	0.7240	0.7243	0.7311	0.7370
Rec	0.8758	0.8909	0.8964	0.9133	0.9254
F-Score	0.7815	0.7943	0.7966	0.8073	0.8152

Table 2. Result of P (M|W)

Portion	0	1/4	1/3	1/2	2/3
Prec	0.7218	0.7416	0.7444	0.7429	0.7540
Rec	0.8871	0.9161	0.9276	0.9270	0.9386
F-Score	0.7913	0.8147	0.8208	0.8197	0.8304

Table 3. Result of P (W|M)

Portion	0	1/4	1/3	1/2	2/3
Prec	0.7545	0.7693	0.7704	0.7667	0.7839
Rec	0.8018	0.8296	0.8407	0.8372	0.8323
F-Score	0.7745	0.7950	0.8021	0.7985	0.8035

Table 4. Result of P (W|M) * P (M)

From the three tables, we can see that as new sentences are added into the training set, the performance improves. Comparing Tables 2, 3 and 4, the poor performance of P (W|M)* P (M) is partially due to unbalance in the training set. The higher occurrences of some specific meaning items increase P(M) and affect the result during voting.

5 Conclusions

In this paper we proposed a cooperative model incorporating finite state model and statistical model for language understanding. It takes all of their advantages and suppresses their shortcomings. The successful incorporation of the methods can make our system very robust and scalable for future use.

We notice that the series model of the finite state machine model actually incorporates some semantic knowledge from human beings. Ongoing research work includes finding new ways to integrate semantic knowledge to our system. For the statistical learning model, the quality and the different configurations of training set highly affect the performance of models trained and thus their abilities to process sentences. The balance of training set is also a big issue. How to build a balanced training set with single finite state machine model will remain our important work in the future. For the learning mechanism, Naïve Bayesian learning requires more understanding of different factors' roles and their importance. These problems should be investigated in future work.

Acknowledgements

The author would like to thank Deepak Ravichandran for his invaluable help of the whole work.

References

Eugene Charniak. 1997. Statistical Parsing with a Context-free Grammar and Word Statistics. *Proc. Of AAAI-97*. pp. 598-603.

M. Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. Proc. of the 35th ACL.

S. Miller, R. Bobrow, R. Ingria, and R. Schwartz. 1994. Hidden Understanding Models of Natural Language," Proceedings of the Association of Computational Linguistics, pp. 25-32.

W. Swartout, et al. 2001. Toward the Holodeck: Integrating Graphics, Sound, Character and Story. *Proceedings of 5th International Conference on Autonomous Agents*