MRE: A Study on Evolutionary Language Understanding

Donghui Feng, Eduard Hovy

Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Marina Del Rey, CA 90292, U. S. A
{donghui, hovy}@isi.edu

Abstract. The lack of well-annotated data is always one of the biggest problems for most training-based dialogue systems. Without enough training data, it's almost impossible for a trainable system to work. In this paper, we explore the evolutionary language understanding approach to build a natural language understanding machine in a virtual human training project. We build the initial training data with a finite state machine. The language understanding system is trained based on the automated data first and is improved as more and more real data come in, which is proved by the experimental results.

1 Introduction

The lack of well-annotated data is always one of the biggest problems for most training-based dialogue systems. Typically a successful trainable system requires lots of annotated data for training. Without enough training data, it's almost impossible for a trainable system to work. However, most dialogue systems suffer from the lack of enough well-annotated data. In the dialogue system, one of most challenging problems is: how can we build a language understanding machine to handle unexpected new sentences as input while starting with very little or even no real data?

In this paper, we explore the evolutionary approach to build a spoken language understanding system in a virtual human training project to overcome the problem of data sparseness. The approach presented in this paper has been realized in the research project, Mission Rehearsal Exercise (MRE). The goal of MRE is to provide an immersive learning environment in which trainees experience the sights, sounds and circumstances they will encounter in real-world scenarios [1]. Figure 1 gives the pipeline of the whole procedure. The language processing part plays the role to support the communication between trainees and computers. Audio signals are first transformed into natural language sentences by speech recognition. Sentence interpretation is used to "understand" the plain text string recognized by ASR (Automatic Speech Recognition) and extract semantic information for subsequent processing such as dialogue management and action planning. This paper focuses on the construction of the language understanding part.

For the evolutionary approach, we start with only a story script and based on this we build the initial training data with a finite state machine. The language understanding system is trained with the automated data first and is improved as more and more data come in, which is proved by the experimental results.

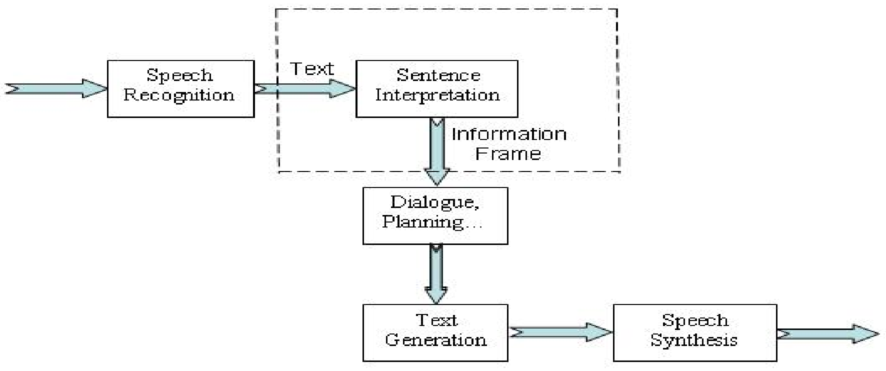


Figure 1. MRE pipeline

The rest of this paper is organized as follows: In Section 2, we discuss the related work. Section 3 describes the semantic representation for natural language understanding. The details of the language understanding model are given in Section 4. Section 5 explores the evolution experimental results of our system. And the paper is concluded with Section 6.

2 Related Work

For natural language understanding, traditional systems first perform syntactic parsing before semantic analysis. Charniak [2] and Collins [3] applied statistical parsers for syntactic parsing to get the dependence tree of the sentence. Miller et al. [4] gave a hidden understanding model for finite state concept network and meaning trees. The system reported by Schwartz et al. [5] divided the procedure of language understanding into three stages, namely, semantic parsing, semantic classification and discourse modeling. Some understanding tasks were transformed into the problems of machine translation [6] [7]. Our approach is to use the power of the statistical approach to obtain automatic adaptability.

On the other hand, some of the state-of-the-art dialogue systems reduce this to the problem of type classification like HMIHY from AT&T [8]. In these systems the final category is represented by a single value. However, in our domain, the result is a cascading semantic frame, which makes the problem of understanding much more complex and difficult.

For the problem of data collection, most dialogue systems suffer from the lack of enough well-annotated training data, especially for frame semantics of language understanding [9]. Some previous applications relied on making annotations of the training data manually, which is both time and effort-consuming. Gildea and Jurafsky [10] reported their work in the project, FrameNet, to build a statistically based semantic classifier. However, their database has not yet approach the magnitude of resource available for other NLP tasks. Fleischman et al. [11] in their work used maximum entropy models to overcome the problem of data sparsity. Some other works have also tried to overcome this with feature-based understanding [7]. Sampson [12] explained the idea of evolutionary language understanding in his book.

In our work, for the same reason and ease of the annotation of training set, we explore the evolutionary language understanding approach. First, we propose to use finite state machine to generate and annotate the training set efficiently. A finite state machine is built to generate and annotate all the anticipated cases as training materials and a traversing algorithm can produce all the training cases. Later the language understanding model is gradually improved as more and more real data come in. We describe the shallow semantic representation first in next section.

3 Semantic Representation

The task of spoken language understanding in our project is to interpret a recognized English sentence into a shallow semantic frame. In our domain our topmost semantic information frame is defined as follows:

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

Figure 2. Topmost information frame

Here <semantic-object> may be one of three types: *question*, *action*, or *proposition*. Question refers to requests for information; action refers to orders and suggestions except requests, and all the rest fall into the category of proposition. The definitions of the second-level and third-level semantic frame are given in Figure 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>
```

```
<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. Second-level and third-level information frame

In Figure 4, we give an example of information frame for the English sentence "who is not critically hurt?". However, we can not directly learn nested knowledge from input sentences and cascading frames. 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. Each of them is called a meaning item and is identified by the statistical classifier separately. Reversely the set of flattened meaning items can be composed and restored to a normal cascading frame.

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)	<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 4. Example of interpretation and re-representation to handle nesting

4 Language Understanding Model

We adopt Naïve Bayes classification as our learning mechanism. To prepare the initial training data, we use finite state machine to obtain result case frames in the domain and transform to flattened meaning item set.

4.1 Naïve Bayes Classifier

Given a string of English words, say, an English sentence, our goal is to extract all of their most possible meanings as represented in the frames. We express this probability as P(M|W). Here, W refers to the words and M refers to the meanings. With Bayes' law, we have Formula 4.1.

$$\arg\max_{M} P(M \mid W) = \arg\max_{M} \frac{P(W \mid M) * P(M)}{P(W)} \qquad (4.1)$$

In this domain, P(W) can be viewed as a constant. Thus (4.1) changes to (4.2) as follows:

$$\arg\max_{M} P(M \mid W) = \arg\max_{M} P(W \mid M) * P(M) \qquad (4.2)$$

Formula 4.2 is composed of two meaningful parts, with P(M) to determine what meanings to express and P(W|M) to determine what words to use in order to express the specific meaning [4]. We name P(M) the *meaning model* and P(W|M) the *word model* and discuss them in the following sections.

4.2 Meaning Model

The meaning model, P(M), refers to the probability that each meaning occurs in the corpus. Here, meanings are represented by meaning items including both slot-value pair and hierarchy information.

Let $C(m_i)$ be the number of times meaning item m_i appears in the training set, P(M) is computed as follows:

$$P(m_i) = \frac{C(m_i)}{\sum_{j=1}^{n} C(m_j)} \quad (4.3)$$

All the values can be acquired by counting the meaning items of all the case frames in the training set. Table 1 shows some example entries of the trained meaning model.

Table 1. Example entries of meaning model

m_i	$P(m_i)$
<i> ^mood interrogative	0.06130325
<i> <t0> ^type question	0.06130325
<i> <t0> <t1> ^type state	0.04613755
<i> <t0> ^q-slot polarity	0.04521792

4.3 Word Model

In Naïve Bayes classification, P(W|M) stands for the probability of words occurring with specific meanings. The specific meanings refer to meaning items including both slot-value pair and level information.

We introduce language model into our system. Let $C(w_j|m_i)$ be the number of times word w_j appears under meaning item m_i; $C(w_{j-1}w_j|m_i)$ the number of times word sequence $w_{j-1}w_j$ appears under meaning item m_i; $C(w_{j-2}w_{j-1}w_j|m_i)$ the number of times word sequence $w_{j-2}w_{j-1}w_j$ appears under meaning item m_i, we can obtain the probability as follows:

$$P(w_j \mid m_i) = \frac{C(w_j \mid m_i)}{\sum C(w_k \mid m_i)} \quad (4.4)$$

$$P(w_j \mid m_i, w_{j-1}) = \frac{C(w_{j-1}w_j \mid m_i)}{\sum C(w_{j-1} \mid m_i)} \quad (4.5)$$

$$P(w_j \mid m_i, w_{j-2}w_{j-1}) = \frac{C(w_{j-2}w_{j-1}w_j \mid m_i)}{\sum C(w_{j-2}w_{j-1} \mid m_i)} \quad (4.6)$$

Formulas 4.4, 4.5, and 4.6 give the probabilities of the word under given meaning items for unigram, bigram and trigram respectively. All the parameters in the formulas can be acquired by counting the mappings between words and meaning items in the training set. Although they are three different language models, they can be stored in a single table. Figure 5 gives some example entries of the trained word model.

```
<i> ^mood interrogative
    who 0.00013071
        is          0.82727273
            not   0.10989011
            badly            0.01098901
    is    0.00654375
        not         0.00181587
            critically        0.2
    not   0.00001188
        critically  0.2
            hurt  0.5
            injured 0.5
```

Figure 5. Example entries of word model

As Figure 5 shows, given the meaning item "<i> ^mood interrogative", the word "who" has a probability of 0.00013071, the word "is" following "who" has a probability of 0.82727273, and the word "not" following "who is" has a probability of 0.10989011.

4.4 Understanding Component

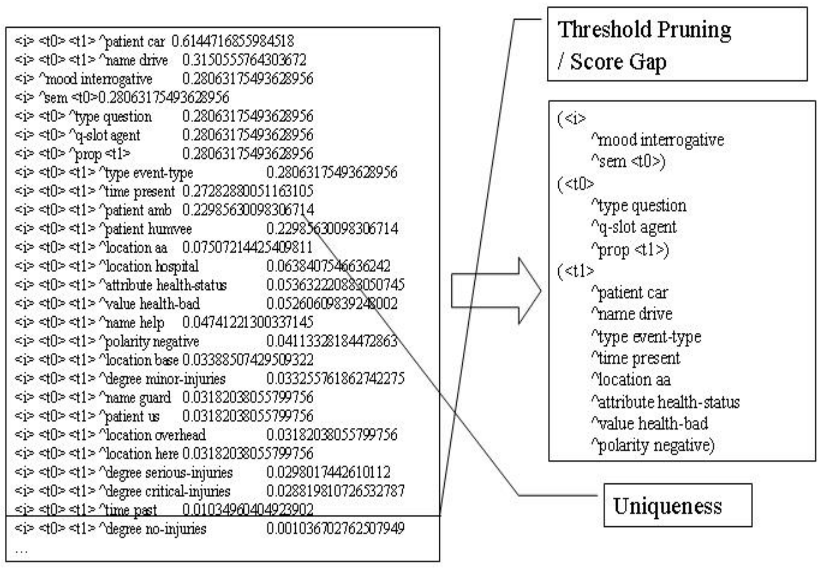
The language understanding component is a key part of our system. Its main task is to interpret an input English sentence with the model acquired in the training procedure and produce a cascading information frame as the result.

With the word model and the meaning model, we can interpret each sentence using the Naïve Bayes classifier. Given an English sentence, all the words are divided into separate units. With the specific language model P(W|M) (say, the trigram model) we obtain a set of candidate meaning items for each word patterns in the sentence, each associated with a probability. For each candidate meaning item, we take the product of this probability and P(M) as the meaning item's final probability. Thus each word pattern is classified with a set of candidate meaning items.

We normalize each classified set of meaning items and use a weighted sum voting scheme to compose all the classification results. In the learned tables, each English word pattern is associated with a set of probable meaning items. A candidate meaning item may receive probabilities from several word patterns in the input. The accumulation of all the probabilities represents the total score for each meaning item. In the final result, the meaning items are ranked based on their total probability scores, and are transformed into a cascading information frame as the output.

The procedure inevitably produces some noisy results. Some meaning items may contradict with others. We adopt two pruning strategies to eliminate noise. The first is to prune unsatisfactory meaning items based on a gap in the probability values. Where there is a large enough jump between the probabilities of two adjacent meaning items, the lower and everything below are removed. The degree of jump can be defined with a threshold value. The second concerns ambiguity. If a slot has more than one value, the values are grouped together and only the top value is selected. Figure 6 gives an example of the understanding procedure.

With these strategies, the system has the ability to learn and can deal with unexpected sentence patterns. It can extract as much information as possible even if only parts of the input sentences are recognized, and will never die even if new words appear in the sentences. However, the result information frame may also carry some noisy results for future processing. The evolutionary performances are investigated in Section 5.



```
<i> <t0> <t1> ^patient car 0.6144716855984518
<i> <t0> <t1> ^name drive    0.3150555764303672
<i> ^mood interrogative      0.2806317549362856
<i> ^sem <t0>0.2806317549362856
<i> <t0> ^type question      0.2806317549362856
<i> <t0> ^q-slot agent       0.2806317549362856
<i> <t0> ^prop <t1>          0.2806317549362856
<i> <t0> <t1> ^type event-type               0.2806317549362856
<i> <t0> <t1> ^time present  0.2728288005116310
<i> <t0> <t1> ^patient amb   0.2298563009830671
<i> <t0> <t1> ^patient humvee                0.2298563009830671
<i> <t0> <t1> ^location aa   0.0750721442540981
<i> <t0> <t1> ^location hospital             0.063840754663624
<i> <t0> <t1> ^attribute health-status       0.0536322208830507
<i> <t0> <t1> ^value health-bad              0.0526060983924800
<i> <t0> <t1> ^name help     0.0474122130033714
<i> <t0> <t1> ^polarity negative             0.0411332818447286
<i> <t0> <t1> ^location base 0.0338850742950932
<i> <t0> <t1> ^degree minor-injuries         0.0332557618627422
<i> <t0> <t1> ^name guard    0.0318203805579975
<i> <t0> <t1> ^patient us    0.0318203805579975
<i> <t0> <t1> ^location overhead             0.0318203805579975
<i> <t0> <t1> ^location here 0.0318203805579975
<i> <t0> <t1> ^degree serious-injuries       0.0298017442610112
<i> <t0> <t1> ^degree critical-injuries      0.0288198107265327
<i> <t0> <t1> ^time past     0.0103496040492390
<i> <t0> <t1> ^degree no-injuries            0.0010367027625079
...
```

```
(<i>
    ^mood interrogative
    ^sem <t0>)
(<t0>
    ^type question
    ^q-slot agent
    ^prop <t1>)
(<t1>
    ^patient car
    ^name drive
    ^type event-type
    ^time present
    ^location aa
    ^attribute health-status
    ^value health-bad
    ^polarity negative)
```

Figure 6. Example of understanding procedure

5 Evolutionary Experimental Results

5.1 Initial Training Set with FSM

The initial training set requires all the sentences to be provided with correct mappings to their information frames. Here we propose to ease the construction of a well-annotated training set with finite state machine.

The main idea is to design a finite state network and place all related information on the arcs as the input and the output. Every sentence string starts from the finite state network's "START" state, and any successful matching of pre-specified patterns or words will move the system forward to another state. The string to be parsed must match exactly with the input on the arcs and go through from one state to another. Any matching procedure arriving at the "END" state means a successful interpretation of the whole sentence. The composition of all the output along the parsing path gives the cascading information frame for the input sentence. Otherwise, the interpreter will die and return failure.

A complete finite state interpreter requires all the target sentence patterns be available during designing. In our domain the initial training set includes 65 sentence patterns and 23 word classes. Figure 7 gives some examples of target sentence patterns and word classes. The implemented finite state network consists of 128 states totally. To make this sensible in more cases, the point is to put variables on the arcs for both input and output.

Using the finite state machine and lexicon exhaustively, we obtain all the possible sentences as the training corpus by the Cartesian product. The total number of sentences is 20,677. After manually removing unpragmatical or odd sentences, we have 16469 sentences remained. This approach is much more efficient than simply building the mapping of a sentence and information frame one by one manually.

```
$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

5.2 Data Collection

With the system trained on the initial training set, we are in a position ready to interpret any new sentence coming in. In our evolutionary experimental test, we have trainees test our system and collect the useful sentences in the conversation. In each test, several trainees use our system in the virtual training environment. Here, we use I to represent the initial training set, Ni to represent the newly collected data. Every time, the new knowledge collected from the test is added into the training set and the system is re-trained with the new training data. In our study, totally we have 3 new data sets coming in, and we annotate with N1, N2, and N3. Therefore the training sets are I, I+N1, I+N1+N2, and I+N1+N2+N3 respectively. Table 2 gives the size of each data set.

Table 2. Data size in terms of number of sentences

I	N_1	N_2	N_3
16469	210	135	122

Each time the conversations are recorded and annotated manually. Only those meaningful sentences in this domain are kept. The sizes of the new data in terms of the number of sentences are 210, 135, and 122 respectively. The system's evolutionary results are analyzed in Section 5.3.

5.3 Result Analysis

As discussed above, we get the training set bigger and bigger. We use the new training data set to get an improved model and test on the new data. Taking the frames we manually build as the real answers, we define precision, recall, and F-score to measure the system's performance. Since our case frames are nested and have different levels, the corresponding metrics are described as follows:

$\text{sub\_precision} = \frac{\#\text{ of correct slot - value pairs}}{\#\text{ of slot - value pairs from learning model}}$	(5.1)
$\text{sub\_recall} = \frac{\#\text{ of correct slot - value pairs}}{\#\text{ of slot - value pairs from real answer}}$	(5.2)

For the whole case frame we calculate the average precision and recall.

$$\text{precision} = \frac{\sum_{level} \text{sub\_precision}_{\text{level}}}{\#\text{ of levels}} \qquad (5.3)$$

$$\text{recall} = \frac{\sum_{level} \text{sub\_recall}_{\text{level}}}{\#\text{ of levels}} \qquad (5.4)$$

We adopt F-Score to incorporate the two metrics, which is defined by Formula 5.5.

$$\text{F - Score} = \frac{2 * (\text{precision} * \text{recall})}{\text{precision} + \text{recall}} \qquad (5.5)$$

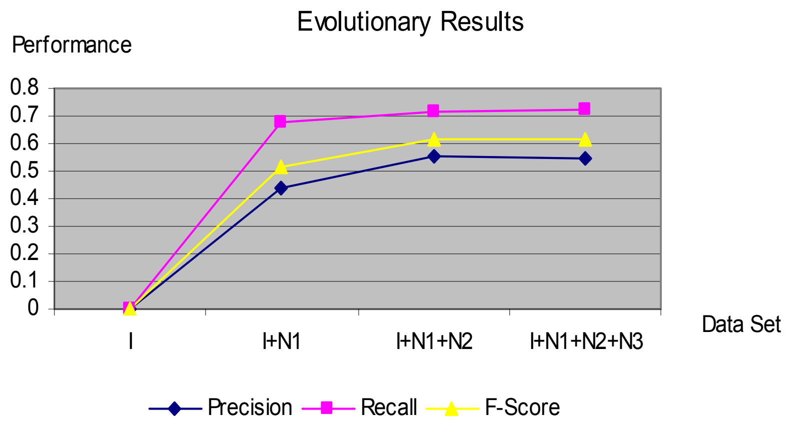


Figure 8. System performance

Figure 8 gives the system's performance curves as more data are introduced in. The results are encouraging, though further refinement and filtering are needed. We set the performances trained with the initial training set as 0. From Figure 8, we can see the system's performance improves as more data come in. Basically, we can understand that later tests with N_2 and N_3 have better performances because the previous set has already incorporated real data whose features and distributions are more similar to later real test data. One of the reasons that the performance with I+N_1 +N_2 +N_3.changes slightly with I+N_1 +N_2 is that the data set N_3 may contain more data with new features from the previous set, but this will benefit the future test results as more and more similar situations occur. That's the way why evolutionary approach always learns more knowledge from its experience and can handle future situations better. In this way, we can make the system more and more robust to handle unknown sentences in the real test.

6 Conclusions

The lack of well-annotated data is always one of the biggest problems for most training-based dialogue systems.

In this paper, we explore the evolutionary language processing approach to build a natural language understanding system for dialogue systems in a virtual human training project. The initial training data are built with a finite state machine. The language understanding machine is trained based on the automated data first and is improved as more and more data come in, which is proved by the experimental results.

The quality and the configuration of the training set affect the ability to process sentences. How to build a balanced training set with single finite state machine will remain one of our important future problems. Ongoing research also includes improving pruning approaches and finding new ways to integrate semantic knowledge to our classifier.

References

1. Swartout, W., et al.: Toward the Holodeck: Integrating Graphics, Sound, Character and Story. Proceedings of 5th International Conference on Autonomous Agents. (2001)
2. Eugene Charniak. Statistical Parsing with a Context-free Grammar and Word Statistics. AAAI-97, (1997) pp. 598-603
3. Michael Collins. Three Generative, Lexicalised Models for Statistical Parsing. Proc. of the 35th ACL, (1997) pp. 16-23
4. S. Miller, R. Bobrow, R. Ingria, and R. Schwartz. Hidden Understanding Models of Natural Language, Proceedings of ACL Meeting, (1994) pp. 25-32
5. Schwartz, R., Miller, S., Stallard, D., and Makhoul, J.: Language Understanding using hidden understanding models. In ICSLP'96 (1996.), pp. 997-1000
6. Klaus Macherey, Franz Josef Och, Hermann Ney. Natural Language Understanding Using Statistical Machine Translation, EUROSPEECH, (2001) pp. 2205-2208, Denmark
7. K. A. Papineni, et al. Feature-based language understanding, Proceedings of EuroSpeech'97, Greece, vol 3, (1997) pp. 1435-1438
8. A.L. Gorin, G. Riccardi and J.H. Wright. How may I help you?, Speech Communication, vol. 23, (1997) pp. 113-127
9. W. Minker, S.K. Bennacef, and J.L. Gauvain. A Stochastic Case Frame Approach for Natural Language Understanding, Proc. ICSLP, (1996) pp. 1013—1016
10. D. Gildea and D. Jurafsky. Automatic Labeling of Semantic Roles, Computational Linguistics, 28(3) (2002) 245-288 14
11. Michael Fleischman, Namhee Kwon, and Eduard Hovy. Maximum Entropy Models for FrameNet Classification. EMNLP, Sapporo, Japan. (2003)
12. G. Sampson, 1996. Evolutionary Language Understanding, Cassell, NY/London (1996)
13. Peter F. Brown, et al. Class-Based n-gram Models of Natural Language, Computational Linguistics, 18 (4), (1992) 467-479