マルチメディア通信と分散処理 69−13
（1995. 3. 3）

自然言語インタフェースに基づいた電子秘書システムの構成

谷垣宏一* チャクラボルティ・ゴウタム** 白鳥則郎**

*東北大学電気通信研究所／情報科学研究科 **東北大学電気通信研究所

概 要

近年のコンピュータ環境の発展に伴い，「エージェント」と呼ばれる，人間の仕事の一部を代行するシステムが可能になってきた．これらのエージェントの機能を，全ての人間にとって利用しやすくする方法の一つとして，エージェントと人間のインタラクションに自然言語を用いる方法が考えられる． 我々は，エージェントに付加する自然言語インタフェースを，ニューラルネットワークに基づいて設計した．このインタフェースは，エージェントのタスク・ドメインに関する自然言語サブセットを学習させることにより，そのタスク用の自然言語インタフェースとして構築されるものである． また，本システムでは，ユーザが入力として複数の文を用いることを許しているため，全ての意味構造を表現するためにニューラルネットワークに必要とされる入力素子の数は，膨大なものとなる．本稿では，こうした構造を取り扱うことが可能なニューラルネットワークを提案する．さらに，実験の結果をもとに，乱れた入力表現に対する本システムの頑健さ等についても述べる．

キーワード： 自然言語インタフェース，ニューラルネットワーク，エージェント，混成学習機構

Trainable Natural Language Interface for Intelligent Agent

Koichi TANIGAKI*, Goutam CHAKRABORTY**, and Norio SHIRATORI**

*Department of Information Science / Research Institute of Electrical Communication, Tohoku University

**Research Institute of Electrical Communication, Tohoku University

abstract

With the recent development of Computer Systems, various human-like tasks could now be accomplished by simulated Agents. However people to using such an agent could be of various levels of expertise, and it is scarcely possible for everyone to fully exploit the facilities available from an agent.

In this paper, we propose a trainable Natural Language User Interface (NLUI) for such agents. The proposed model of NLUI is a general one which is based on Neural Network techniques. After being trained with proper Natural Language subset, it becomes a task-specific NLUI and acts correctly in the domain for which it is trained.

The network is designed such that it can work with large input sentences. In fact the module has no constraint as regards to the length of the input sentences. Also, it is very robust, and during experiment showed very high recognition rate even with unseen and broken inputs.

Keywords: Natural Language Interface, Neural Network, Agent, Hybrid Learning Schemes

1. Introduction

With the recent developments of Computer Systems, many tasks which was previously done by human, could now be accomplished by computer. At the same time computer networks are reaching almost everyone. This background has made possible a situation, where an agent, simulated by computer system, could deliver human oriented tasks of different kinds. However, the users of these agents, we call 'user' from now, could be of various levels of intelligence and expertise. The best way to exploit facilities available with an agent is to have a natural language interface wrapped around the core tasks of the agent. More so, if the NL Interface could cope with broken and conversational forms of inputs. It would be sufficient if the Natural Language Processor(NLP) could execute well only in the domain concerned with the task of that particular agent, and reject irrelevant phrases.

As the agents are of various types, the domain of language used by them could also be different. Instead of creating individual NLP, we look for one which, when initialized and trained properly, would deliver proper NL Interface to the agent. Also, when the task of the agent changes or is extended, the NLP should be able to adapt to the new domain, after proper training.

To achieve the required features, as mentioned, the Rule-based NLP would be inadequate for our purpose. Considering a generalized model of the training aspects, a Neural Network(NN) approach was considered to be suitable.

Till now, we already have some works with NN-NLP [1] [2] [3] [4]. In one of the recent works, PARSEC[5] parses sentences based on deep case representation. It succeeded to demonstrate the effectiveness of Connectionist Model compared to ordinary algorithmic parser, especially for noisy input sentences, which are common in conversational form.

In this present context, only parsing is not sufficient. We need a complete Natural Language Interface. In our proposed NN-NLP model, we further need

(1) Some *Internal Rules* to decide proper target functionalities from semantic structure of input sentences, using task-specific knowledge e.g. what information in which slot of case-frame, what inference on that, and which functiononality etc.), which is difficult to formalize in general, and has been a big load to system designers.

(2) *NL Generator* which generates NL sentences based on the semantics of the input and the context.

The main contribution of this paper is our proposed network, which can infer target function from semantic structure of user's input. The input, in the form of ASCII characters, is directed to the agent, and is usually consists of a few sentences. We proposed a network model which is able to cope with very long sentences. In fact, there is no limit in the length of input sentence. This would be impossible with existing networks of NLP, which would grow in size with the sentence length. Even during experiments we used sentences as long as having 6 clauses.

The natural language generation part was done simply by using some reserved template strategy. They are selected deterministically from the semantic structure of the user's input command and its subsequent classification by our proposed network. Further required variables e.g. time, date, place etc. are extracted properly from the Knowledge Base of the agent.

The proposed model is simulated in an experiment, where our NLP is used as NL interface for secretary agent. It responded nearly perfectly even on unseen and broken input phrases.

In the next section, we give an over-all description of the whole model. In Section 3, we focus on Function Deciding module and explain it in detail. In Section 4, we discuss about the experimental results, where our network is being simulated to perform the task of a secretary. Results by testing with unseen and broken phrases show a good level of generalization by our NN-NL interface. Finally in conclusion, we summarize our work, indicating the drawbacks and the future extensions.

2. Model

2.1 Formal Description of the Model

The flow diagram of the proposed model is shown in Figure 2.1. Definitions regarding the dynamic behavior are given in Definition A1 and A2. Data definitions are given in Definition B1 to B3. Meta-symbols are used with their usual meanings.

Definition A1 defines Intelligent Agent(IA), which takes as its input <USERINPUT> from environment (user), and generate the corresponding output <OUTSENTENCES>.

As we can see in Figure 2.1, IA is divided into 3 parts, NL-Interpreter, NL-Generator, and Task Handler. Task Handler is a module which handles the core task of the agent with simple command-based interface.

Definition A2 defines the three functional units, namely NL-Interpreter, NL-Generator, and Task Handler.

Definition B1-1 defines that <USERINPUT>, which has local structure of <Clause>, <phrase>, and the smallest units are a series of <word>s. The features of these <word>s must have been defined and saved earlier in the Knowledge Base of Feature Translator.

By NL-Interpreter, the <USERINPUT> is parsed, and then classified into proper function according to its meaning. The generalized <FunctScript> is sent to NL-Generator.

NL-Generator send to Task Handler <Command>s described in the <FunctScript> (Definition B1-2), and get

<response>s from Task Handler (Definition B2).

Finally, NL-Generator select proper <template> sentence corresponding to the <response>, from a set defined by the <FunctScript>. Values for the variables are also obtained from the <response> to generate <OUTSENTENCES> to be delivered to the user.

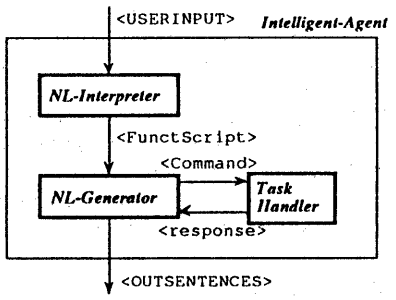

Figure 2.1 The Agent Model

Definition A1

Intelligent-Agent : <USERINPUT>→<OUTSENTENCES>

Definition A2

NL-Interpreter : <USERINPUT>→<FunctScript>

NL-Generator :

<FunctScript>×(*TaskHandler*)*→<OUTSENTENCES>

TaskHandler : <Command>→<response>

Definition B1-1

```
<USERINPUT> ::= (<Sentence>)*
<Sentence> :=
     <Clause>((_<Clause>)|(,_<Clause>))*
     (.|?)
<Clause> ::= <Phrase>(_<Phrase>)*
<Phrase> ::= <word>(_<word>)*
<word> ::= { W | W∈KBofFT }
```

Definition B1-2

```
<FunctScript> ::=
     (<Command>)* +(<template>)*
<Command> ::=
     <commandname>×(<parameter>)*
<commandname> ::=
     { C | C is a predefined.name }
<parameter> ::= { P | P is a parameter }
     e.g. date, time, etc.
<template> ::=
     { T | T is sentences with fixed
          structure and blanks for
          variables }
```

Definition B2

```
<response> ::=
     { R | R is an output from TH
          identifying response }
```

Definition B3

```
<OUTSENTENCES> ::=
     (<template>+(<response>)* )*
```

2.2 Model Description

Figure 2.2.1 shows the block-level structure of the whole model. Task Handler and four internal modules of NL Processor are described in the next subsections.

(i) *Task Handler*

Task Handler(TH) consists of the Knowledge Base related to the task, and TH-Program, which reads from and writes into the KB, depending on the input command. TH is specific for the task.

Linguistic jobs are achieved in NL-Interface. The other actual core tasks are all facilitated by the TH.

(ii) *Feature Translator*

The form of user's input sentences are ASCII character sequences. The first module of NL Interface is Feature Translator, which translate the ASCII characters word by word into internal word representation of Word Feature, as is traditionally done in [1] [5] (see Fig 2.2a).

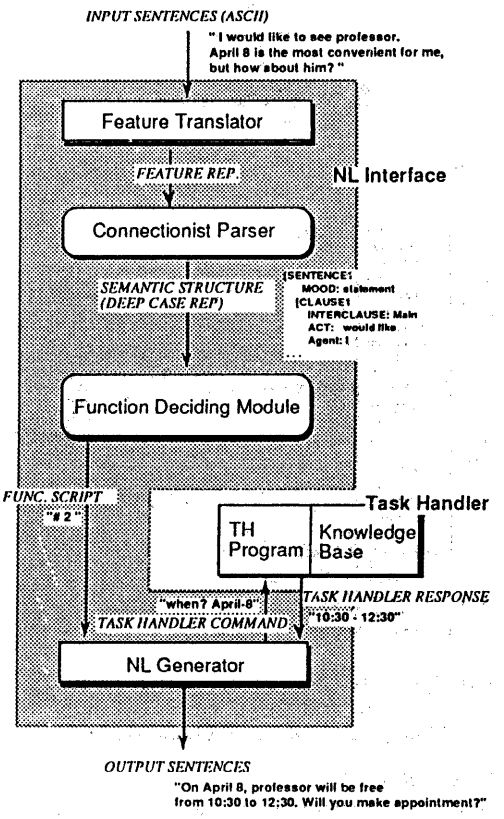

Figure 2.2.1 Internal Modules

Word feature is a set of numbers, which represent the syntactic and task-specific semantic features of that word. The values of this set of numbers are used as activation levels of input units in the networks of subsequent module. They are arbitrarily coded and saved in the Knowledge Base of Feature Translator during

initialization.

Our feature mapping differs from other works in that, its values are continuous. Traditional representations were binary. There, a particular feature is represented by a dimension consisting of several bits. With a particular input word, only one of the bits in that group may be fired, depicting its feature for that dimension. In our feature representation, a dimension is represented with a single unit, and the property of the dimension is expressed with its activation value, which varies from a minimum 0 to a maximum 1.(see Fig. 2.2b). This scheme saves a lot of units in the Input layers, when a large number of semantic features are required.

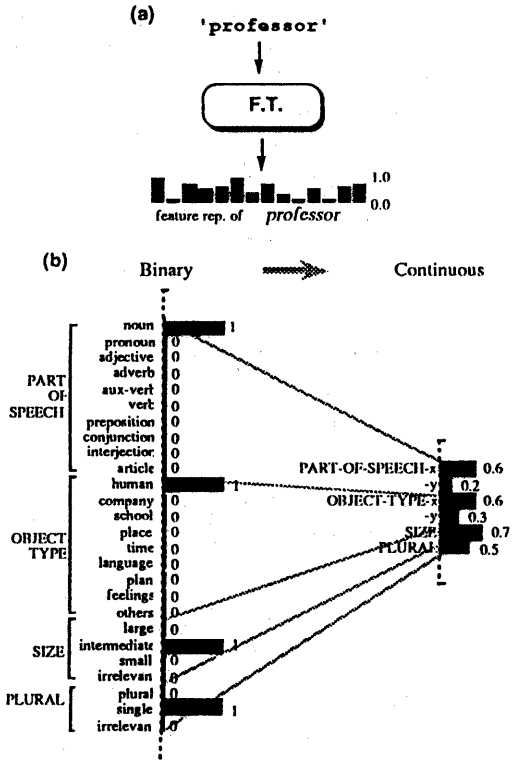

Figure 2.2.2 (a) Feature Translator Module (b) word feature of 'professor' in binary/continuous representation

(iii) *Connectionist Parser*

Figure 2.3 shows the structure of Connectionist Parser (CP) module[5]. CP takes a sentence (the series of word-features) as input and generates semantic parse tree of the sentence. For Semantic structure, case-based representation is used [8] [9].

CP consists of 5 neural networks, all of which are feed forward type with a single hidden layer and with partial connections. The different networks are linked by C-programs, which handles the correct input-output relations of the networks. Sometimes these modules need to do some data manipulation before delivering to the input of the next network..

PHRASE network detects boundaries between phrases of a sentence inputted as a series of words(features). CLAUSE network takes a series of phrase blocks and detects boundaries between clauses in the sentence. The ROLE network correctly labels the deep cases, e.g. ACTION, AGENT, RECIPIENT, etc. The INTERCLAUSE network correctly determines the mood label of the whole sentence, e.g. STATEMENT, INTERROGATIVE, etc. The MOOD network correctly determines the interclause label e.g. MAIN, SUBORDINATE, RELATIVE, etc. to each of the clauses.

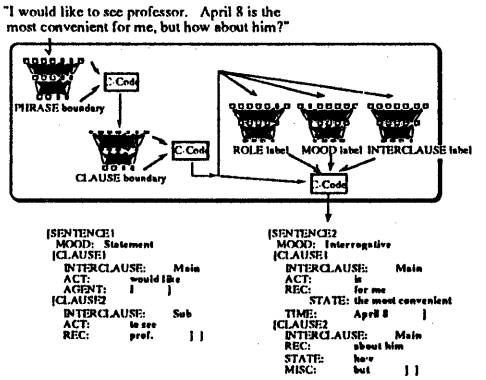

Figure 2.3 Connectionist Parser Module

(iv) *Function Deciding Module*

This is the main module of our proposed System (Figure 2.1). The Function Deciding Module (FDM) takes semantic structure of input sentences from CP. Then by using NN described in Section 3, it classifies the input to some proper function. The corresponding function script is sent to NL Generator.

Detailed explanations are given in Section 3.

(v) *NL Generator*

From this module, NL Sentences are generated, to be delivered to the user. NLG takes as input the script from FDM and responses from TH. Based on the procedure described in the script, NLG select the template (fixed output sentences) most appropriate to the TH-responses. The blanks (variables) are filled with proper values received from the TH, to generate the NL response for the input.

3. Function Deciding Module

Figure 3.1 shows the construction of Function Deciding(FD) Module. This module consists of a single Neural Network(NN) and a set of Function Script(FS).

Semantic structure from parser is presented to this

NN, transformed by C-code to a suitable form. Then the network classify the input to some proper function, and the corresponding FS is sent to NL-Generator module.

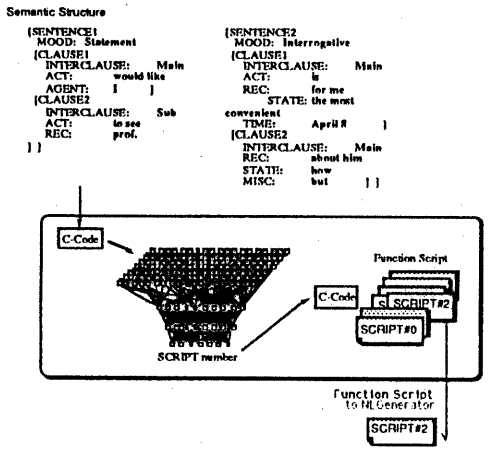

Figure 3.1 Function Deciding Module

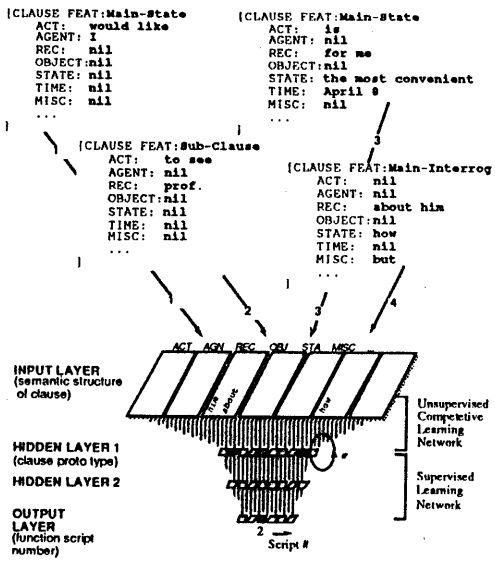

Figure 3.2 Function Deciding Network

Figure 3.2 shows the structure of FD Network. This network is characterized by the following three features.

(1) *Hybrid Learning Schemes*

This is a hybrid network, for which the upper part is an Unsupervised Competitive Learning (UCL) network and the lower part is a Supervised Learning (SL) network.

By using UCL, we could accelerate the learning speed, which would otherwise be extremely slow, if only Back-Propagation learning is used.

(2) *Representation of Input Layer*

Input to the IA may be several sentences, and each sentence again consists of several clauses. If we present all the input semantic structure directly to input layer, extremely large number of input units and connections would be required. For an input of three sentences, each consists of 3 clauses, we would need about **15,000** input neurons. The calculations are as follows.

14,634 = [(27Features/word × 4Words/case ×
15Cases/clause + 5Labels/clause) ×
3Clauses/sentence + 3Labels/sentence] ×
3Sentence/input

To facilitate using smaller network, the whole semantic structure needs to be divided. For this, some mechanism would be required to memorize the information of the previous sequence. To implement this, a network with recurrent connection [6] [7] is suitable. Our model takes semantic structure, as input and keep the previous context in the Hidden Layer 1(HL1).

Instead of representing the whole semantic structure of input sentences together, the network sequentially takes clauses, which are local structure of the total input sentences. A clause is a congenial unit into which the whole input should be divided. This is because the minimum unit of structure in deep-case is an Action-rooted tree, which is usually closed within a clause.

By UCL network these clauses are clustered into similar groups. Depending on the input clause, only one unit in HL1 fires (set to '1'). This unit is a prototype of a cluster, and represent the category to which the input clause belongs to.

Activations of HL1 units are gradually decayed with cycles of input clauses, by self-recurrent connection α (<1, here we use 0.85). So after all the clauses are presented, HL1 would memorize the whole input, represented by activation pattern of clause-prototypes. Activation would be 1 for the last input clause and α^N for the first input clause, when total N number of clauses are inputted.

Finally, this HL1 pattern is classified by lower SL network, and we would get the proper function

By the proposed method, we could reduce the number of input neurons from the above 15,000 to about 1,200. We succeeded to classify even very long inputs, consisting of 3 sentences and 6 clauses, in our experiment.

(3) *Distance*

In our network, the input domain forms a n-dimensional space, where n is the number of features in a clause. The UCL network clusters input vectors based on their distances in that input space.

However, each axis of the input space represents a word-feature. For one such dimension, the value 1.0 may be used to represent 'human', 0.8 to 'object', and 0.6 to 'language'. So, nearness between those values is no consequence to the resemblance of the meanings.

Instead of using Euclidean distance, we define a new distance D between input vector ξ and i-th prototype W_i, based on feature representation:

$$D(\xi, W_i) \equiv \Sigma_j (\xi_j \mathrel{!}= W_{ij})$$

where any $\xi_j \mathrel{!}= W_{ij}$ contributes 1

else $\xi_j = W_{ij}$ contributes 0.

4. Experiment

The proposed generalized model is simulated and then trained with the task of a Secretary. Here agent should act for secretary of a professor by executing tasks such as to answer inquiries about professor's schedule, to make appointment, to take a message, etc.

For that task, 7 commands for Task Handler are defined, as a start e.g.'when?' in Figure 2.2.1. For IA, we defined 8 functions which are some combinations of these 7 commands, as well as some different functions.

To train the network, we need different variations of input sentences. This was done by collecting examples from different persons. The whole input domain consists of about 200 words and 118 patterns (106 patterns in conversational forms and 12 patterns of broken sentences). Each pattern itself consists of a few sentences of which the longest sentence consists of 6 clauses.

We here describe two experiments on the Function Deciding Network.

First, generalization performance is examined. By generalization we mean how the network behaves, when unseen sentences are inputted. 106 input patterns of conversational style are divided into two groups of Training Samples(68 patterns) and Testing Samples (39 patterns). The network is trained with 68 training patterns.Then its generalization ability is examined by testing with the unseen 39 test patterns.

Figure 4.1 shows the network performance and learning curve with training epochs. After completion of training, the network showed nearly perfect performance(92.3%) on testing patterns. Moreover, as shown in the learning curve, the training converged very quickly in less than 100 epochs.

we also examined robustness to broken input sentences. After the completion of training with 68 samples, the network performance is tested with 12 broken sentences as input, in Figure 4.1. The result is fairly satisfying, considering that the network had neither special noise-modeling nor any training for such broken sentences.

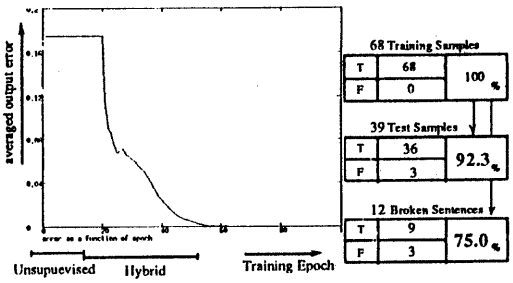

Figure 4.1 Generalization Performance

5. Conclusion

In this paper, we have proposed a trainable NL Interface for Intelligent Agent. Our intention was to design the model to be a general one. It could be used for different purposes, after it is trained with proper NL subset of its domain. The model is based on neural network, so that it could have such properties as *Learnability, Adaptivity, Noise-Torelance*, which we don't expect with Rule-based NLP.

For the NL Interface, the parser is a modified version of a previous work on NN NL parser. We designed two modules namely Function Deciding Module and Natural Language Generator.

In the experiment, we trained and examined the main network on the specific task of a professor's secretary. The result was fairly satisfactory.

However, in the proposed NLI, we haven't treated the higher level processing, such as context. For future extension of this work, it would be possible to add some context processing mechanisms, and to design an extended model of IA.

References

[1] McClelland, J.L. and A.H.Kawamoto (1986): Mechanism of sentence processing: Assigning roles to constituents, in Parallel Distributed Processing, vol.2, ed. J.L.McClelland and D.E.Rumelhart, The MIT Press.

[2] Rumelhart, D.E.and J.L.McClelland (1986): On Learning the Past Tenses of English Verbs, in Parallel Distributed Processing, Vol.2, ed. J.L.McClelland and D.E.Rumelhart, The MIT Press.

[3] Waltz, D. and J.Pollack (1985): Massively Parallel Parsing: A strongly interactive model of natural language interpretation, Cognitive Science 9:51-74.

[4] Cottrell, G.W.(1989): A Connectionist Approach to Word Sense Disambiguation, San Mateo, CA: Morgan Kaufman.

[5] Jain, Ajay N (1991): PARSEC: A Connectionist Learning Architecture for Parsing Spoken Language, doctoral thesis in the field of computer science, Carnegie Mellon.

[6] Stonetta, W.S., T. Hogg, and B.A. Huberman (1988): A Dynamical Approach to Temporal Pattern Processing. In Neural Information Processing Systems(Denver 1987), ed. D.Z. Anderson, 750-759. New York: American Institute of Physics.

[7] Mozer, M.C. (1989):A Focused Back-Propagation Algorithm for Temporal Pattern Recognition. Complex Systems 3, 349-381.

[8] Fillmore, C (1968): The case for case, In Universals in Linguistic Theory, ed. E.Bach and R.T.Harms, New York: Holt.

[9] Bruce, B. (1975): Case systems for natural language, Artificial Intelligence 6(4):327-60.