An Elementary Natural Language Learning Program

Chengxiang Zhai (zhai@claritech.com)
CLARITECH Corporation
5301 5th Avenue
Pittsburgh, PA 15232, USA

Abstract

The study of language acquisition is interesting to several fields, such as, cognitive science, linguistics, and artificial intelligence. A computer model of language acquisition is particularly interesting, because it can lead to a cognitive theory of language acquisition in the form of a computer program, which can be tested. Current computer models of language acquisition are inadequate to explain human language acquisition in several aspects, including the interaction with concept development and tolerance of noisy inputs. This paper proposes a semantic memory model of language that is consistent with modern grammar theories. A development-based language acquisition model based on discrimination and generalization is presented. The model suggests some possible interactions between concept attainment and language acquisition. A program based on the acquisition model was implemented in Prolog, and example interactions with the program have demonstrated its ability to learn "nonmonotonically" from noisy inputs.

Introduction

The ability to acquire language is a common and elementary cognitive ability of humans in the sense that every child can learn his native language very early when his problem solving ability is still relatively "low". Computer simulation of human language learning is an interesting research area, because it can lead to a cognitive theory of language acquisition in the form of a computer program, which can be tested. Moreover, building computer programs for learning natural language is itself an interesting area in artificial intelligence. However, despite its importance, relatively little research has been done in computer models of language acquisition, compared with the work of language acquisition from other perspectives such as linguistics or cognitive science (see for example, Gelman & Byrnes, 1991; Dromi, 1993 among others). Morikawa (1988) offers a thorough survey of computer models of language acquisition done before and during the 1980's. More detailed reviews of some early individual models can be found in McMaster, 1975; Pinker, 1979; and Langley & Carbonell, 1987. Some recent work includes Liu & Soo, 1993 and Kazman, 1994.

Current computer models of language acquisition generally fall into two categories, "theory-based models" and "data-driven models" (Morikawa, 1988). Theory-based models all assume some kind of linguistic theory. Such models include the model by Berwick and Weinberg (1984) based on transformational grammar and the model by Block, based on syntax crystal theory. They tend to use only the surface form of utterances as input data, and avoid meaning or semantics. However, these models generally leave behind the remaining task of accounting for the acquisition (or existence) of the linguistic theory.

Data-driven models, on the other hand, start with the characteristics of early language and consider the empirical data from children and include such factors as typical linguistic and nonlinguistic input for children, children's knowledge about the real world, and conceptual development along with postulated learning rules (Morikawa, 1988). In such models, "prior knowledge" for language learning is assumed to be at a minimum, and a general cognitive mechanism is seen as accounting for language acquisition.. Two typical models are John Anderson's ACT* and Siklossy's ZBIE (Anderson, 1983; Siklossy, 1972). Another example is Selfridge, 1980. Although many such models simulate only the early part of language development but not the subsequent parts, they all somehow imply a certain cognitive mechanism behind human language acquisition.

For example, ZBIE is a program which can accept a set of "sentence-meaning" pairs and learn to generate a sentence with a new meaning accordingly. The "sentence" is simply a string of words; while the "meaning" is a structured expression in some functional language, FL. The mechanism behind the program is a pattern matcher working on a set of "translation templates" (Siklossy, 1972). ACT* also accepts a set of "sentence-meaning" pairs and learns to generate a sentence from a meaning representation. But, in contrast to ZBIE where the "meaning" is intended to be a description of an external "meaning stimulus" (just like a "speech stimulus"), the "meaning" in ACT* is essentially an "internal meaning representation". ACT* is a general cognitive architecture based on production systems and symbolic networks. Anderson (1983) has demonstrated that language acquisition can be accomplished within ACT* in a way similar to other cognitive activities.

While these data-driven models all suggest some kind of explanation of human language acquisition, there are two problems with most of them. One is that the language acquisition program only learns from "correct" sentence-meaning pairs. Specifically, these programs will fail to learn, if the "meaning" is only a partial meaning of the "sentence" in a pair. In other words, the input data are supposed to be correct. The other problem is that the acquisition program has not shown how concept develop-

ment interacts with language learning. Concepts are largely a "primitive notion" built into the formalism for meaning representation. But, Clark (1991) and Keil (1991), among others, have argued that concept learning interacts with word meaning acquisition.

This paper addresses these two issues and is a step toward answering the following two questions:

First, what's the relationship between "concept attainment" and "syntactic category acquisition"? Or, how can "concept learning" help "syntactic category learning" (and vice versa)?

Second, are "ill-formed pairs" (i.e. those where the "meaning" is inexact) useful for language learning?

The paper proposes a new computer model for language acquisition. The acquisition model is based on a semantic memory model for language acquisition, which is consistent with modern grammar theories. The learning process consists of both generalization and discrimination of semantic memory nodes. A program based on the acquisition model was implemented in Prolog. Actual running examples of the program have demonstrated its ability to learn "nonmonotonically" from noisy inputs.

Framing the Language Learning Problem

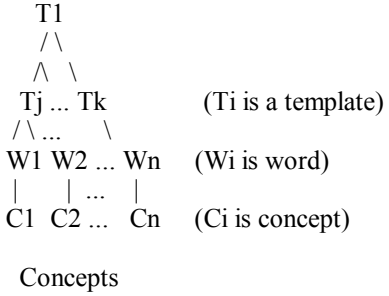
In order to focus on the study of interaction between concept development and language acquisition, we put significant constraints on both the natural language to be learned and the "world" being simulated. The natural language being learned contains only simple noun phrases (e.g., circle, large square, dark square, etc.), and the "world" is supposed to occupy a simple 2-dimensional space with a couple of simple geometric figures of different sizes and different colors. Although the natural language grammar here is almost trivial and the "world" is very, very limited, it is sufficiently complete to show some interesting aspects of any proposed language acquisition model. Besides, the learning approaches adopted by the program are not limited to the particular framing of the problem presented here, they can be used for a more general problem of language learning. We will discuss the limitations of the approach later.

The input to the program is a series of [" noun phrase", "meaning"] pairs, where "noun phrase" is a simple noun phrase and is intended to describe a concept and "meaning" is a "feature structure". This is to simulate the input that a baby would get when an adult says to him “a large triangle” while pointing to a large triangle block. The noun phrase represents the verbal input from the adult; while the feature structure represents the perceptual stimulus the baby received from the block. The meaning part describes the perceptual stimulus on the level of "features", and thus differs from the representation formalism used in most other models. “[size: 1 , color: RED, edge: 3]” is an example of feature structure, representing “a triangle object of red color and size 1”.

The program is expected to learn both to comprehend natural language and to acquire the concepts described by the natural language phrases.

A Semantic Memory Model of Language

We propose the following semantic memory model of language. The model provides an integrated representation of linguistic and conceptual knowledge. It is essentially a forest of trees of syntactic templates with natural language words as leaves. Each word is further connected to a concept node (feature structure). The picture below shows a sample model.



Nodes and Links

The three kinds of nodes are the concept node, word node, and template node. A *concept node* represents a concept and is further connected to some feature structure (which itself may be a tree-like network). Concept nodes reflect the basic concepts the program has learned so far. A *word node* represents a word in the natural language vocabulary. Word nodes reflect the vocabulary capacity the program has learned so far. A *template node* represents certain pattern of word combinations. Template nodes reflect the grammar rules the program has learned so far.

The three links are the lexicon link, abstraction link, and join link. A *lexicon link* is a link connecting a word with a concept. If a word W is linked to a concept C through a lexicon link, then W has C as one of its possible meanings. Lexicon links represent the language lexicon the program has acquired. An *abstraction link*, or "isa" link, is a link between two syntactic templates. If T1 is linked to T2 through an "isa" link, then T1 can combine with any words or pattern with which T2 can combine. Abstraction links correspond to grammar rules such as T2 -> T1 . A *join link* is a link that defines the grammar rules for obtaining a new template by joining two existing ones. A join link can be classified as one of the set { left-adjunct, left-head, left-complement, right-adjunct, right-head, right-complement }. There are only four possible combinations of join links as shown below. No other join link combination is allowed. The relation between join link combinations and grammar rules is also given below

(1). Left adjunction rule

```
          /       \
   left-adjunct  right-head     TEMP ----> TEMP1 TEMP2
        /              \                (left adjunction)
      TEMP1      TEMP2
```

(2). Right adjunction rule

```
          TEMP
          /      \
   left-head   right-adjunct     TEMP ----> TEMP1 TEMP2
        /             \                  (right adjunction)
      TEMP1      TEMP2
```

(3). Left complementation rule

```
                TEMP
             /            \
   left-complement  right-head     TEMP ----> TEMP1 TEMP2
        /                        \          (left complementation)
      TEMP1            TEMP2
```

(4). Right complementation rule

```
          TEMP
          /        \
   left-head  right-complement      TEMP ----> TEMP1 TEMP2
        /                  \               (right complementation)
      TEMP1       TEMP2
```

A snapshot

The following is a "snapshot" of the memory model with the encoded grammar and lexical rules given on the right side.

```
                                          Rules:
          TEMP1
            /    \                          TEMP1 -> TEMP2 TEMP3
 L-Adjunct/        \ R-Head                   (left-adjuct rule)
          /             \                   TEMP2 -> Large
    TEMP2       TEMP3                       TEMP2 -> Red
    / \                / \                  TEMP3 -> Square
   /isa \isa        /isa \ isa              TEMP3 -> Triangle
  /      \          /        \
 Large  Red  Square  Triangle
 |          |      |          |
 |lex       |lex   |lex       |lex
 |          |      |          |
 FS1      FS2    FS3        FS4
```

Connection to modern grammar theories

One very interesting aspect of the memory model above is its connection with Chomsky's universal grammar theory (Chomsky, 1981; Cook,1988) and other modern grammar theories such as Head-driven Phrase Structure Grammar (Pollard & Sag, 1994). The major connection is the type of rules allowed in the model.

Most modern grammar theories have generally assumed some particular forms of grammar rules (called X-bar rules). Each grammar category is of the form X, X^1, X^2 ... , where X is a primary category, such as noun, adjective, or verb. Any grammar rule must be of the following general form.[X^i --> Y X^j] where j <= i and the order between Y and X^j is a "parameter" determined by a specific language. The two basic forms of rules implied by this are exactly the adjunction rule and the complementation rule.

1. Adjunction rule

This rule has the form X^i --> Y X^i meaning that Y is some modifier of X^i .

2. Complementation rule

This rule has the form X^i+1 --> Y X^i meaning that Y is an argument of X^i .

Explanation of comprehension

Based on this integrated model of semantic representation, the ability of humans to comprehend language can be explained as follows.

When receiving a sentence(or phrase) containing words W1,...Wk, the language user searches through the template net in a bottom-up way until a template which matches the string is found. While searching the template net, the language user simultaneously builds the semantics (i.e, the feature structure) of the sentence based on the feature structures connected with W1,..., Wk, and the links of the relevant templates.

The semantics is compositional in that the feature structure of a *combined template* (a parent in the tree) can be determined based on the feature structures of the *templates being combined* (the daughters in the tree).

Learning based on discrimination and generalization

Operators on Feature Structures

Corresponding to the two possible combinations of templates, we introduce two operators acting on the feature structures. One is INST (meaning "Instantiate") which *instantiates* one feature structure by specifying the value of one of its features to be another feature structure. The other is ADJU (meaning "Adjunct") which *extends* one feature structure by including another feature structure as its subset.

Formally, given two feature structures FS1 and FS2, and one feature f of FS1, the operators INST and ADJU are defined as follows.

INST(FS1, f,FS2) = FS1 [f->FS2].
(instantiate FS1 by assigning FS2 to feature f)
ADJU(FS1,FS2) = FS1 Union FS2
(extend FS1 by including FS2)

Starting from some primitive feature values and some basic feature structures, a complex feature structure can be built based on INST and ADJU.

Interestingly enough, the feature structures and their operator are also similar to the feature structures used in some current grammar theories (Pollard & Sag, 1994).

Discrimination and Generalization

Natural language phrase discrimination

Each input phrase will be compared with all the phrases the program can understand to determine the maximal

understandable subphrase of the input. The rest of the input is further processed by the program when possible.

Feature structure discrimination

Two feature structures can be compared with each other to find their difference. The difference can provide a way to modify a feature structure (a concept) already learned based on the input feature structure. If one feature structure is a sub-structure of another, then discrimination can also find its complement sub-structure.

Syntactic template generalization

If the program is able to understand two sub-phrases, but not the whole phrase, the program will generalize such combination into a more general combination template. The label of the rule will be determined by the relations among the feature structures of the whole phrase and the sub-phrases.

The learning process

Given a "phrase-meaning" pair (PH,FS), where PH is a natural language phrase and FS is a feature structure, the program will follow the following learning process.

1. Find the longest sub-phrase SUBPH in PH, such that PH = [SUBPH] [XX] or PH=[XX] [SUBPH] and SUBPH is understandable. Let FS(SUBPH) be the feature structure for SUBPH. Then, XX is either [] or not. If XX is [], then do step 2, otherwise, do step 3.

2. Do feature structure discrimination between FS(SUBPH) and FS, and revise the existing feature structures according to the discrimination result. Exit.

3. If XX is understandable and its feature structure is FS(XX), then go to step 4, otherwise, do the following. Discriminate between FS and FS(SUBPH), see if we can find a feature structure FS' for XX. If not, go to step 1 and try other possible phrase discriminations, otherwise, use (XX,FS') as input, go to step 1.

4. Learn the following new rule.

NEW-TEMP-> TEMP-OF(XX) TEMP-OF(SUBPH)
NEW-TEMP-> TEMP-OF(SUBPH) TEMP-OF(XX)
The label of the rule is determined by the relations among FS, FS(XX), and FS(SUBPH). Exit.

Examples of Learning Interaction

In this section, we will give some running examples of the program to show the program's learning ability.

Learning Lexicon and grammar rules

The following example can show how the program learns the lexicon and the grammar rules, which are the two core elements of the language ability, in general.

After following sequence of pairs was given to the learning program, the program was able to understand the phrase “large triangle”.

step	Input	output
1	(“square”,[edge(4),size(any),color(any)])	Yes
2	(“triangle”,[edge(3),size(any),color(any)])	Yes
3	(“red,square”,[edge(4),color(red)])	Yes
4	(“red,triangle”,[edge(3),color(red)])	Yes
5	(“large,square”,[edge(4),size(large)])	yes

The program learned a lexicon that consists of four words “square”, “triangle”, “red”, and “large”. Note that even though “red” and “large” individually were not presented as input, the program can learn them by making discriminations between these examples. The program also learned a simple grammar rule: Words such as “red” and “large” can proceed words such as “square” and “triangle” to form a meaningful phrase (noun phrase). After learning these examples, the memory is exactly the "snapshot" given in Section 3. When given a new phrase “large triangle”, the program can comprehend it as to mean the feature structure:

Step	Input	Output
6	“large triangle”	[edge(3),color(any),size(large)]

Inferring New Concepts(New Words)

The following example shows how the program can infer new concepts and new words based on context.

The input pair sequence is:

step	input	Output
1	(“square”,[edge(4),size(any),color(any)])	Yes
2	(“red,square”,[edge(4),color(red)])	Yes
3	(“red,triangle”,[edge(3),color(red)])	Yes
4	“triangle”	[color(any),edge(3)]

After seeing these three examples, the program could understand the word “triangle” as to mean the feature structure “[color(any),edge(3)]”.

In this example, the program first learned the concept/word "square" from the first pair and then learned "red square" from the second. At this point, the program has been able to learn the word "red" and its meaning (the feature "color(red)"). This knowledge is immediately used to deduce the meaning of the word "triangle" (i.e., the concept "triangle") from the third example "red triangle". Had the program not been able to figure out the meaning of “red” before seeing the example “red triangle”, it would only learn the meaning of the whole phrase “red triangle”, but not the meaning of “triangle”.

Concept Development

The following example shows the program's ability of simulating concept development by revising the feature structure of a concept.

Step	Input	Output
1	(“triangle”,[edge(3),size(any),color(any)])	Yes
2	(“red,triangle”,[edge(3),color(red)])	Yes
3	(“triangle”,[edge(3),height(5)])	Yes
4	“triangle”	([height(5), edge(3), size(any), color(any)]
5	(“red,triangle”,[edge(3),color(red),height(3)])	yes
6	“triangle”	([height(any), edge(3), size(any),

		color(any)])

The program first learned "triangle", "red triangle" as before, but, then, it learned another case of "triangle" with a new feature "height(5)". Thus, it revised the feature structure for the concept "triangle" to include the feature "height" but with a wrong value specification. When it learned another case of "triangle" with a different height, it finally formed the correct feature structure of the concept.

Nonmonotonic Learning (Learning with noise)

This example shows how the program can learn from input with some "noise", that is a pair where the feature structure is not an *exact* meaning representation of the phrase. (It may miss some features or contain extra noise features).

step	Input	output
1	(“square”,[edge(4),size(any),color(red)])	Yes
2	(“red,square”,[edge(4),color(red),size(small)])	Yes
3	? “red”	[adjunct(true), size(small)]
4	(“red square”, [edge(4),color(red),size(large)])	Yes
5	? “red”	unknown
6	(“square”,[edge(4),color(blue)])	Yes
7	(“red,square”, [edge(4),color(red),size(large)])	Yes
8	? “red”	[adjunct(true), color(red), size(large)]
9	(“red,square”, [edge(4),color(red),size(small)])	Yes
10	? “red”	[adjunct(true), color(red)]

The program first learned a wrong concept of "square" (with its feature "color" having to be "red"), thus when it learned "red square", it formed a wrong meaning for "red" (feature "size"). But, when it further learned an example of "red square" with an inconsistent meaning (size is large), it realized the inconsistency, and revised the meaning of "red" and cleaned it up because the meaning is empty. Now, the program learned another case of "square" which allows it to induce the correct feature structure for the concept "square". But, because of the existence of noise feature "size", when it further learned the case "red square", it formed another wrong meaning of "red" (both "size" and "color"), then, after learning another case of "red square" with a small size, it finally formed the correct meaning of "red".

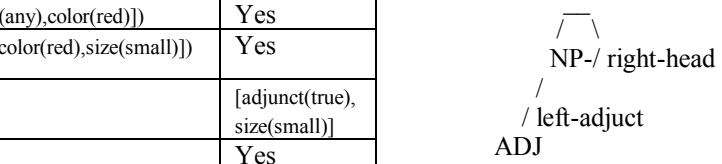
Limitation of the learning program

One major limitation of the learning program is its inability to learn recursive rules. It is generally agreed that human language is recursive. For instance, in English, it is theoretically possible to "stack" an infinite number of adjectives as the modifiers of a noun, thus generating a phrase like "a red, large, delicious, ..., fresh apple".

Obviously, it is impossible for any person to explicitly store all such phrases in his memory, hence there must be some kind of (recursive) rule like the following.

<Noun Phrase> :: = <Noun>
<Noun Phrase> :: = <Adjective> <Noun>
<Noun> :: = apple
<Adjective> :: = red | large | delicious |... |fresh|...

However, the learning process proposed in this paper can not construct such recursive nodes. In order to do this, we need to extend the generalization step to allow "collapse" of two nodes that have dominance relations (i.e., generalization of a parent node and its descendant node). Then, we can easily represent such recursive rules in the proposed semantic memory by a "cyclic node" of the following form.



But, such a generalization rule implies more complex parsing process, since a recursive node can cause infinite search. One possible solution to the problem of infinite search is to keep track of the nodes visited at any time. This will be an interesting issue for further exploration.

Conclusions and Future Work

The paper has proposed a semantic memory model of language and an elementary language learning approach based on discrimination and generalization. A learning program based on the model and approach was implemented in Prolog, and several running examples have been given to show diverse abilities of the program. The paper suggests a unique approach for computer to learn the ability of natural language comprehension at an elementary level. It also shows the possibility to learn from ill-formed phrase-meaning pairs.

The model provides the following answer to the question "How concept attainment interacts with language acquisition?".

- Concepts guide the syntactic discrimination and grammar rule formation
- Linguistic phrases put constraints on the granularity of concepts

One interesting thing with the language learning program is its connection to modern grammar theories. Both the memory model and the feature structures are consistent with several modern grammar theories.

One important future work is to study the limit of complexity of language structures learnable in the acquisition model of the form currently proposed. One particular interesting work is to study how sentences (not just noun phrases) could be learned in the current model (even without recursive extension). There are at least two issues deserving investigation here. One is the acquisition of verbs

and the complementation rules. This will involve the study of more complex concepts (e.g., actions, events) and their attainment. The other has to do with more complex noun phrases, such as those with quantifiers and prepositional phrases.

Another important future work is to study how to extend the current model so as to deal with recursive structure, as mentioned in the previous section.

Further exploration of the connection between the proposed language acquisition model and modern grammar theories will also be very interesting and promising. It is quite possible that many other principles of modern grammar theories will have some implication on the extension of the current model or learning process. For example, "theta-criteria" is a principle in GB which essentially says that syntactic arguments (roughly maximal noun phrases) and the thematic roles of the verb should always match in each grammatical sentence (Chomsky, 1981) (Thus not allowing sentences like "John liked", or "John liked kids the dogs"). Children seem rarely making such mistakes, though they make many other mistakes such as tense and number disagreements (e.g, "he like it"). This proposes the question how to acquire configuration of such thematic roles of verbs. References

Acknowledgments

The author is grateful to Dr. Herbert A. Simon for his advice and comments on this work.

References

Anderson, John, 1983, The Architecture of Cognition, Cambridge, Mass. : Harvard University Press, 1983.

Berwick, R.C., and Weinberg, A.S., 1984, The grammatical basis of of linguistic performance: Language use and acquisition, Cambridge, MA: The MIT Press.

Chomsky, Noam, 1981, Lectures on Goverment and Binding, Dordrecht, Holland; Cinnaminson, [N.J.]: Foris publications, 1981.

Clark, Eve, 1991, Acquisitional principles in lexical development, in Gelman et al. (eds), Perspectives on language and thought. Cambridge Univ. Press. 1991.

Cook, V. J. 1988, Chomsky's unviersal grammar: an introduction, Oxford, UK; New York, USA: Blackwell, 1988.

Dromi, Esther,(ed), 1993, Language and cognition: a developmental perspective, Norwood, N.J.: Ablex Pub. Corp. c1993.

Gelman, Susan, and Byrnes, James,(eds.) 1991, Perspectives on language and thought. Cambridge Univ. Press. 1991.

Kazman, Rick, 1994, Simulating the child's acquisition of the lexicon and syntax -- experiences with Babel, Machine Learning, Vol. 16, 1994, Nos. 1/2, July/Aug. 1994, pp. 87-120.

Keil, Frank, 1991, Theories, concepts, and the acquisition of word meaning, in Gelman et al. (eds), Perspectives on language and thought. Cambridge Univ. Press. 1991.

Langley, Pat, and Carbonell, Jaime, 1987, Language Acquisition and Machine Learning, in B. MacWhinney (Ed), Mechanism of language acquisition, (pp. 115-155), Hillsdale, NJ: Erlbaum.

Liu, Rey-Long, and Soo, Von-wun, 1993, Parsing-driven generalization for natural language acquisition, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 3, pp. 621-644.

McMaster, Ian, 1975, A Proposal for Computer acquisition of natural language, Technical Report, TR75-3, May 1975, Dept. of Computing Science, The Univ. of Alberta, Edmonton, Alberta, Canada.

Morikawa, Hiromi, 1988, Computer Models of Language Acquisition, Computers and Behavior, 1988, Vol. 4. No. 2. pp. 133-45.

Pinker, S., 1979, Formal Models of Language Learning, Cognition, 7(3), 217-284.

Pollard, Carl Jesse and Sag, Ivan, 1994, Head-Driven Phrase Structure Grammar, Stanford: Center for the Study of Language and Information; Chicago: University of Chicago Press, 1994.

Selfridge, M. 1980, A Process model of language acquisition, (Research Rep. No. 172), New HAven, CT:Yale University, Computer Science Department.

Siklossy, Laurent 1972, Computer Natural Language Learning, in Simon, Herbert, and Siklossy, Laurent (eds.) 1972, Representation and meaning: experiments with information processing systems. Englewood Cliffs, N.J., Prentice-Hall, 1972.