AI Magazine Volume 6 Number 1 (1985) (© AAAI)

Learning Language Using a Pattern Recognition Approach

William Katke

IBM Palo Alto Scientific Center, 1530 Page Mill Road, Palo Alto, CA 94303

Abstract

A pattern recognition algorithm is described that learns a transition net grammar from positive examples. Two sets of examples—one in English and one in Chinese—are presented. It is hoped that language learning will reduce the knowledge acquisition effort for expert systems and make the natural language interface to database systems more transportable. The algorithm presented makes a step in that direction by providing a robust parser and reducing special interaction for introduction of new words and terms.

We are developing a natural language interface to an expert system for message processing. Both the expert system and its natural language component take a knowledge-based approach. A learning mechanism has been implemented in order to facilitate knowledge acquisition. We believe that learning will extenuate the bottleneck associated with natural language processing.

The basic method for acquiring knowledge is through learning by positive example. Up to this point we have successfully developed an algorithm that learns a simple transition net grammar. It also categorizes words by parts of speech. The categories produced are similar to the ones in a standard dictionary. The syntax that is learned recognizes word phrases but makes no attempt to discover dependencies between phrases.

This algorithm is a robust parser. It does not need to prompt the user for information about new words such as conjugation rules or part of speech. The fact that the parser can deal with patterns it has never seen before makes it useful in applications that mix languages, such as natural language used with tables, shorthand, acronyms, or another natural language.

There are four basic principles to the algorithm.

- The part of speech of a word can be determined by the parts of speech of the preceding and following words: one word before and one word after.
- A transition net is created by connecting the categories of individual words as they appear in input sentences.
- Only one instance of each category is allowed in a transition net.
- A word can be put into more than one category.

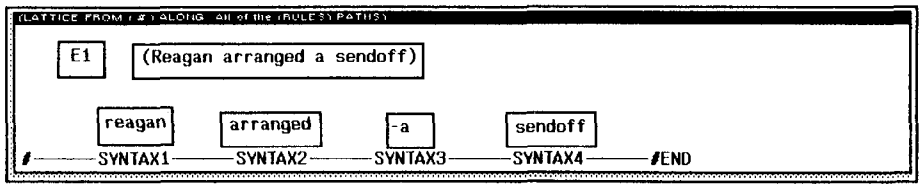
The initial experiment used two passes on the sample sentences. The first pass created two lists for each word. One list called PREVIOUS contained all the words that preceded this word in the examples by one word. The second list was called NEXT and contained all the words that immediately followed by one word. Another way to say this is that we looked at all possible sequential triples of words. For example, for the following sentences:

The party was held yesterday.
The meeting was held today.

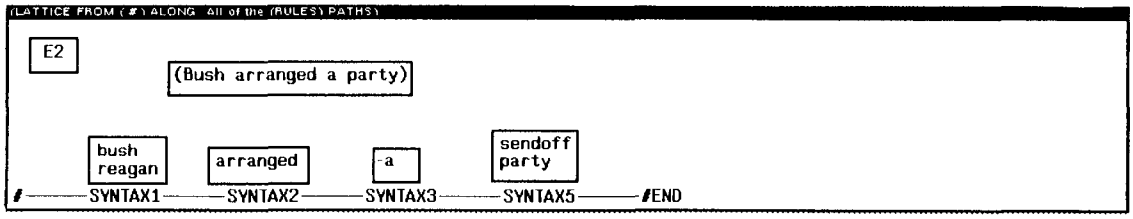
the PREVIOUS list for "was" is (party meeting). The NEXT list for "was" is (held).

The syntax categories were determined by comparing the PREVIOUS and NEXT lists of every word to the PREVIOUS and NEXT lists of every other word. If the PREVIOUS lists of two words intersected, and if the NEXT lists of the two words intersected, then the two words were placed in the same category. The number of words in the intersection was controlled by a parameter.

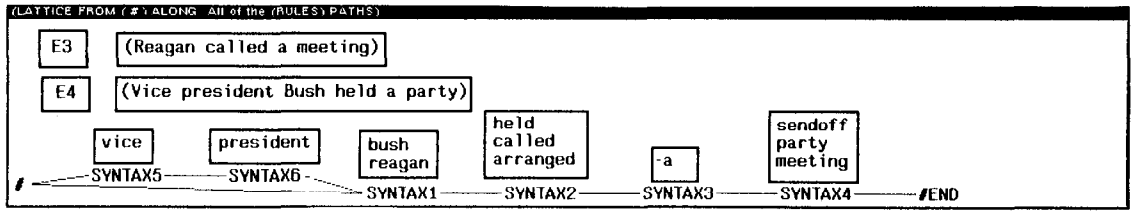
This work was done while at Planning Research Corporation, McLean, Virginia.



Example E1.



Example E2.



Examples E3 and E4.

The second pass looked up the category for each word and created a transition net by connecting the categories that followed each other in the input. If the category did not exist it was created.

The next experiment combined these two passes so that words were categorized, a transition net learned, and a parse obtained at the same time. How the learning works will be illustrated with 17 sample sentences.

The algorithm was implemented on a Xerox 1108 Dandelion. The examples provided below are the actual computer output printed from the screen. E1 starts with no initial net. It creates one by making up a category for each word and connecting the categories together. Note that the beginning and the end of a sentence are treated as categories by the algorithm. The actual system passes the root and the conjugation as separate tokens to the learning algorithm. The morphology was omitted for the sake of simplicity.

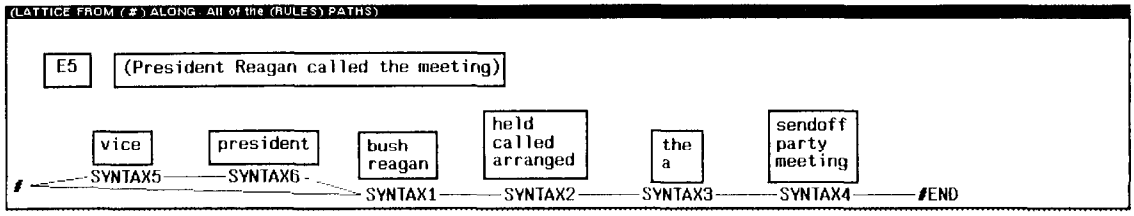
In E2, "Bush" and "party" are put in existing categories because they were preceded and followed by known categories. You may have noticed that category SYNTAX4 changed to SYNTAX5. This is due to an idiosyncrasy of the present implementation and will be explained after the rest of the examples. Please ignore it for now.

Sentence E3 adds "meeting" to category SYNTAX4 and "called" to category SYNTAX2. Sentence E4 adds "held" to SYNTAX2 and creates and links two new categories, SYNTAX5 and SYNTAX6, to accommodate "vice" and "president."

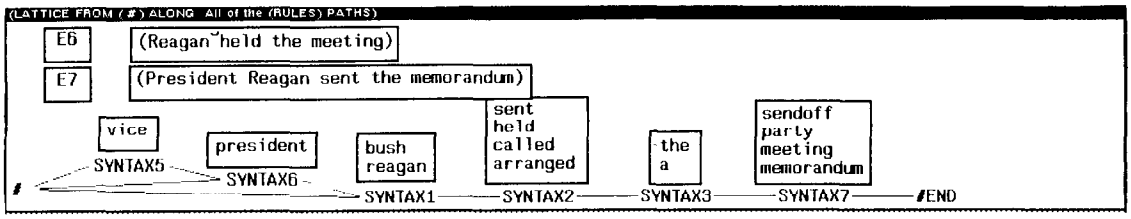
Sentence E5 adds "the" to category SYNTAX3.

Sentence E6 parses, demonstrating the learning of a grammar that can parse sentences it has never seen before.

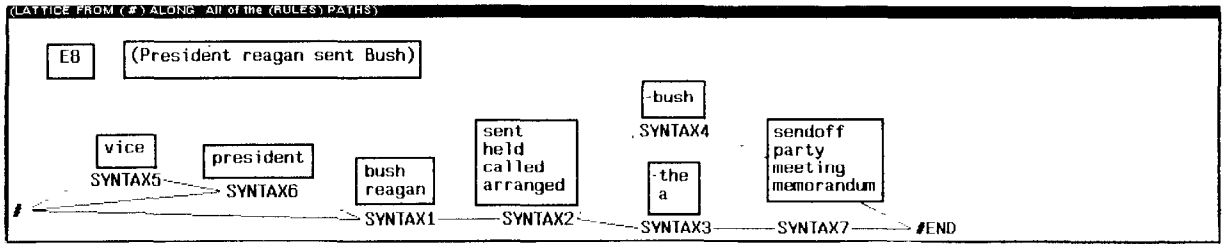
Sentence E7 builds a connection between # and SYNTAX6 and adds "sent" to category SYNTAX2. This is the



Example E5.



Examples E6 and E7.



Example E8.

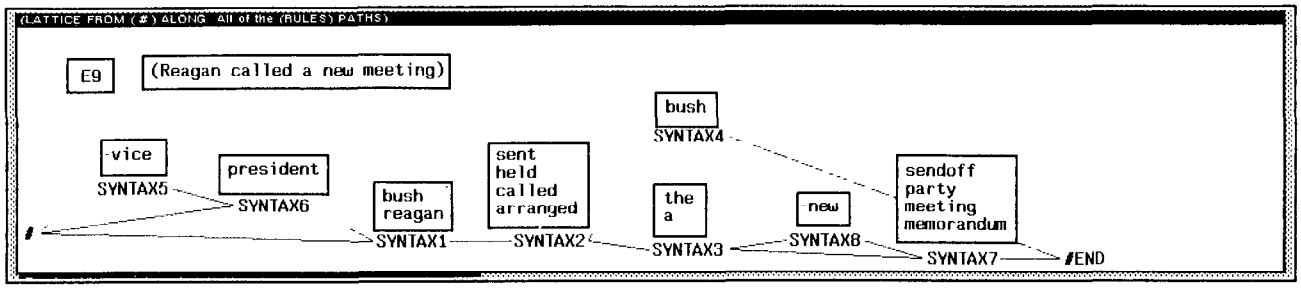
first example of connections being built between existing categories. This happens when the next two categories will match the next two words in the sentence. This is in keeping with the principle that the part of the speech is determined by the preceding and following words.

Sentence E8 creates a new category for "Bush." The reason SYNTAX2 was not connected to SYNTAX1 is that SYNTAX1 is not already connected to #END. This is counter to what happened in E7 and demonstrates how a word is placed in more than one category.

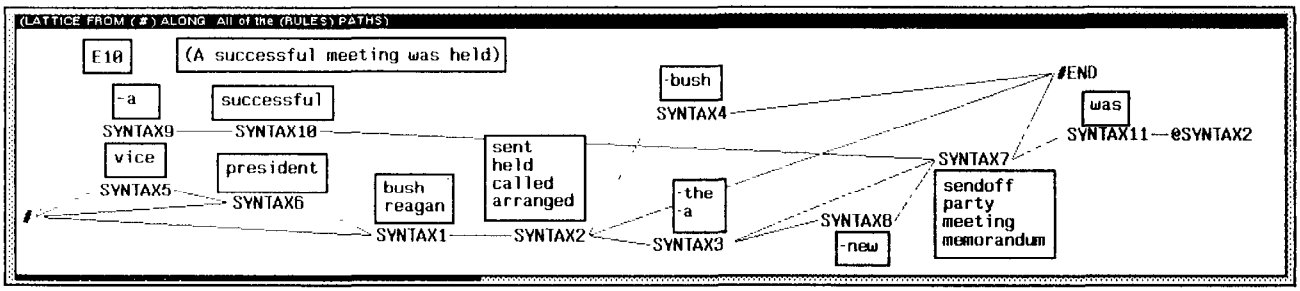
Sentence E9 parses except for "new," which causes a new category, SYNTAX8, to be created and linked between SYNTAX3 and SYNTAX7. SYNTAX8 is created and connected to SYNTAX3 because no category following SYNTAX3 contains "new." This is the same reason why categories were created in the first example. The connection between SYNTAX8 and SYNTAX7 occurs just like the connections between existing categories in E7.

The next sentence, E10, starts to exhibit some interesting behavior. Category SYNTAX11 is created for "was" and is connected back to SYNTAX2, which creates a loop. The @ in front of the SYNTAX2 following SYNTAX11 is the way that our graphic tools show a link that causes a loop. The loop is important because it identifies repeating patterns smaller than a sentence. It is the way phrases are recognized.

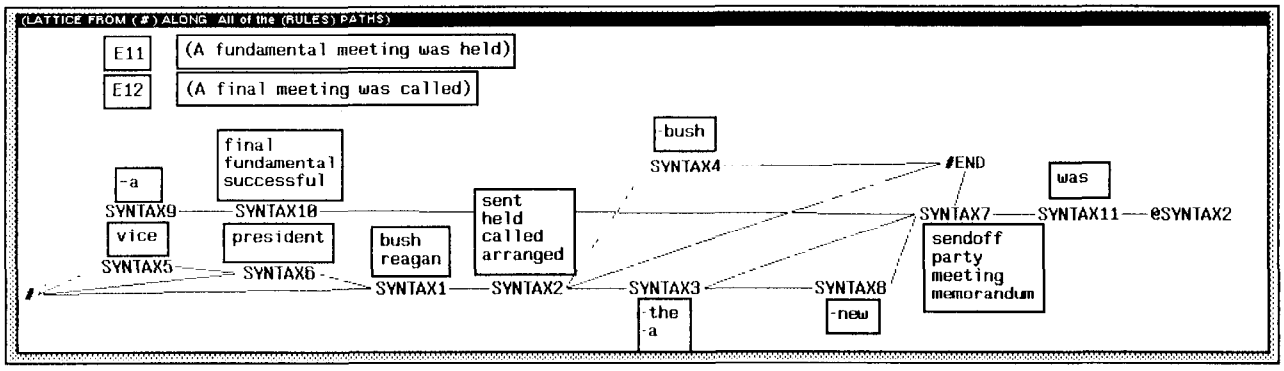
Sentences E11 through E14 show how a category similar to adjectives comes to loop on itself. This indicates that these words can appear in any order with respect to one other. We do not use a category that loops on itself



Example E9.



Example E10.

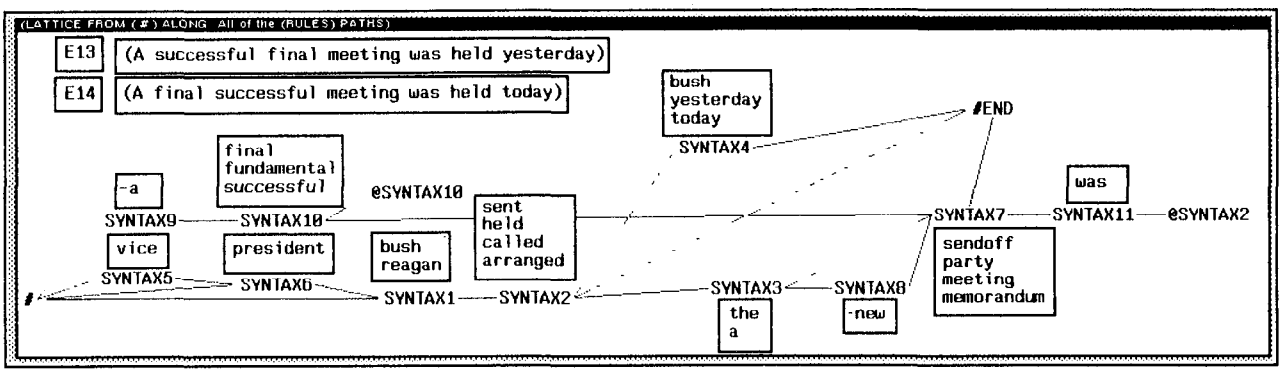


Examples E11 and E12.

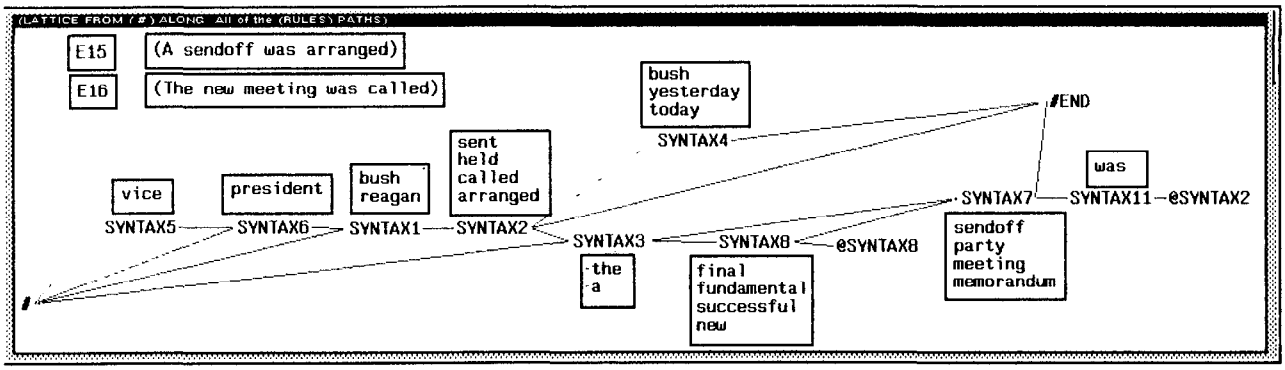
to delineate phrases. E11 and E12 add adjectives to category SYNTAX10. Sentence E13 caused "final" to be put in a new category between SYNTAX10 and SYNTAX7. This category is not shown because it is combined by the action of E14. SYNTAX10 and the new category for "final" had SYNTAX9 in common before and SYNTAX7 in common after. Since SYNTAX10 was linked to the new category, that link was changed to point to SYNTAX10 when they were combined.

To understand the next example, look first at the net for sentences E13 and E14. Sentence E15 causes a link between SYNTAX9 and SYNTAX7 due to "A sendoff." This causes SYNTAX9 and SYNTAX3 to have a NEXT category, SYNTAX7, in common.

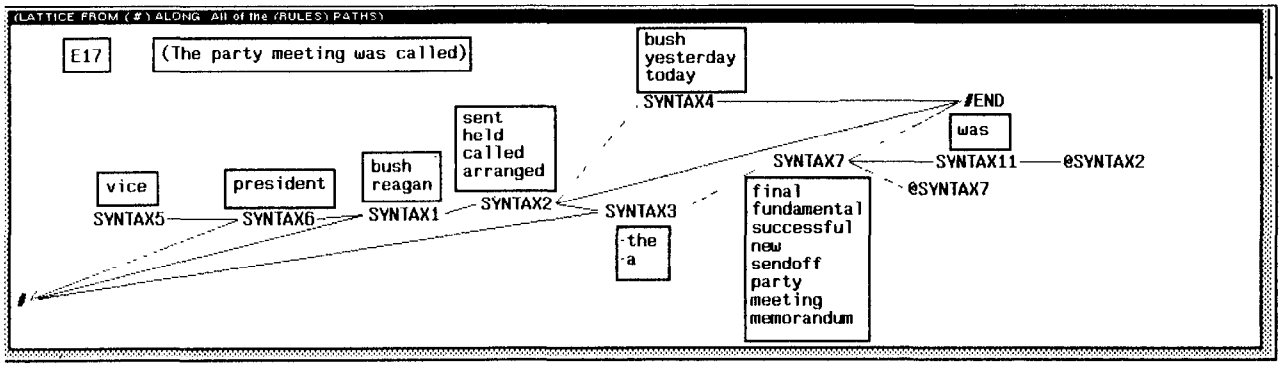
Sentence E15, by beginning the sentence with "The", causes a link to be made between # and SYNTAX3. This



Examples E13 and E14.



Examples E15 and E16.



Example E17.

in turn causes SYNTAX9 and SYNTAX3 meeting the requirements to be combined. This can be seen in the net with E15 and E16.

The action does not stop there. The combining of SYNTAX9 and SYNTAX3 causes SYNTAX8 and SYNTAX10 to meet the requirements also. SYNTAX10 and SYNTAX8 already have a NEXT category, SYNTAX7, in common. When SYNTAX9 and SYNTAX3 were combined it caused SYNTAX10 and SYNTAX8 to have SYNTAX3 in common.

Unlike the original experiment, this algorithm does

not compare every category to every other category. It only compares categories that have new or changed links with their immediate relations. When "The" in E16 was recognized, a link was made from # to SYNTAX3. This caused SYNTAX3 to be compared with all categories following #, which are (SYNTAX9 SYNTAX5 SYNTAX6 SYNTAX1). When SYNTAX9 and SYNTAX3 were combined it caused the links to SYNTAX7 and SYNTAX10 to be changed. This caused SYNTAX3 to be connected to SYNTAX10, which caused SYNTAX10 to be compared to all the other categories that follow SYNTAX3. These are (SYNTAX7 SYNTAX8). This in turn causes SYNTAX8 and SYNTAX10 to be combined.

The syntax learned initially is more complicated than it needs to be. As more input is received and categories combine, the transition net simplifies and the categories start to resemble dictionary parts of speech.

Sentence E17 shows the reason why the number of categories in the intersection of NEXT and PREVIOUS categories should be hgher than one. The algorithm has combined adjectives and nouns into one category. This happened because "party meeting" caused SYNTAX7 to loop on itself. SYNTAX7 and SYNTAX8 already had SYNTAX3 in common via PREVIOUS. When SYNTAX7 is modified to loop on itself, SYNTAX7 and SYNTAX8 now have SYNTAX7 in common via NEXT. This happens because some nouns can function as adjectives without any morphological change. Eventually, verbs would be put into this category as well. If the categories are not combined, a word that can be used as a noun or an adjective will be placed in both categories.

There is an upper bound for combining categories achieved by requiring that all the PREVIOUS and NEXT categories must match. In the above examples only one PREVIOUS and one NEXT category were required to match. When all categories must match, a net that combines semantics and syntax is produced. (Walker, 1981) obtains the same result using the method discussed by (Sager, 1981) Walker's approach achieves taxonomic analysis of text by using word triples to determine semantic word classes and co-occurrence between the classes. Walker has found that analysis useful for text retrieval.

It is time to explain why some of the syntax categories change names from example to example. When sentence E2 was processed it created SYNTAX5 for "party" and connected it between SYNTAX3 and #END. Since SYNTAX4 and SYNTAX5 have NEXT and PREVIOUS categories in common, they combined.

SYNTAX4 becomes an available name and reappears when E3 is parsed. This is the mechanism used to place words in existing categories. It requires no extra code because the ability to combine nodes already exists. This type of combining puts new words into existing categories and functions even when all the PREVIOUS and NEXT categories must agree. This was discussed in the previous paragraph.

As mentioned earlier, this algorithm is a robust parser. It parses ill-formed input because it learns new categories and paths. To prevent constructs that occur rarely from having equal standing with common constructs, use counts are kept, with all links showing how often they have been used. The use counts can be used to garbage collect both the links and categories used just once. Since they can be reconstructed by a single example, nothing is lost, provided garbage collection is not done so often that reinforcement can not occur.

The algorithm has been used for sample Japanese and Chinese sentences and appears to work well. This suggests that the algorithm will work for a large class of human languages. The Chinese examples appear in the appendix.

Future Research

The advantages of this algorithm can be realized only if the algorithm learns a useful syntax. For this reason future research using this approach needs to address several problems. First, a relationship between phrases needs to be learned. For example, a second-level analysis could take tokens for phrases as input instead of words. These tokens could be generated by the current algorithm. Second, over-generalizing needs to be addressed. This could be done by learning more than one grammar at a time. Each one would have a different degree of overlap controlling the combining of categories. By setting parameters, the algorithm will be generalized so it can be used to do both of these things. The number of syntax nets learned and the parameter settings need to be generated automatically to be useful outside a research environment. Third, context beyond one word before and one word after needs attention especially for controlling agreement during generation.

Other Work

Other programs have been developed that learn syntax. One such example is Autoling done in 1968 by Sheldon Klein. It learned a phrase structure grammar including nested rules. The Autoling system used negative examples as well as positive examples. The negative examples were obtained by generating sentences with the current grammar. If the generation produced an ungrammatical sentence the user typed in the correct sentence. Autoling was order sensitive and kept a memory of all the previous input for testing new rules.

Conclusion

Pattern recognition is the foundation of our method of learning and language processing. It has much in common with the neural net approach and the psychological model. It is the author's belief that human cognitive processes consist of pattern recognition controlled by feedback and

the need to discriminate. Logic, even logic less formal than first-order predicate calculus, is not an innate human ability, but must be learned. This does not imply that logic, especially the work done with plans (Schank, 1977), is not a useful basis for language understanding. It is a very important contribution, but even the logic needed for plan-based understanding is learned by the powerful pattern recognition capability of biological brains. Work being done by Robert Levinson and Elaine Rich on self-organizing data bases for interface to an expert system holds a similar view (Levinson, 1984).

Bibliography

Block, H (1961) The Perceptron: A Model of Brain Functioning *Rev. Math. Physics* 34(1): 123-135.

Buchanan, B. & T. Mitchell (1978) Model-Directed Learning of Production Rules. In D A Waterman & F. Hayes-Roth, *Pattern Directed Inference Systems.* New York: Academic Press.

Haas, N & G. Hendrix (1983) Learning by Being Told: Acquiring Knowledge for Information Management In Ryszard Michalski, Jaime Carbonell, & Tom Mitchell, *Machine Learning.* Palo Alto, CA: Tioga.

Klein, S., W Fabens, R. Herriot, W. Katke, M Kuppin, & A. Towster (1967) The Autoling System—Testing an Automated Linguistic Fieldworker. The 42nd Annual Meeting of the Linguistic Society of America.

Holland, J. (1980) Adaptive algorithms for Discovering and Using General Patterns in Growing Knowledge Bases *Policy Analysis and Information Systems* 4(3)

Lebowitz, M. (1980) Generalization and Memory in an Integrated Understanding System. D. Phil. dissertation, Yale University

Levinson, R , E. Rich, & C Wilcox (1984) Self-Organizing Databases for Intelligent Retrieval, *SIGART Newsletter* ACM #86.

Minsky, M. & S Papert (1969) *Perceptrons.* Cambridge, MA: MIT Press

Montague, R. (1974) The Proper Treatment of Quantification in Ordinary English. In R Thomas (Ed.), *Formal Philosophy: Selected Papers of Richard Montague.* New Haven: Yale University Press

Schank, R & R. Abelson (1975) Scripts, Plans, and Knowledge. *IJCAI 4*

Schank, R & R. Abelson (1977) *Scripts, Plans, Goals and Understanding* Hillsdale, NJ: Lawrence Earlbaum Associates

Sager, N. (1981) *Natural Language Information Processing: A Computer Grammar of English and Its Applications* Reading, MA: Addison-Wesley.

Walker, D. & J. Hobbs (1981) Natural Language Access to Medical Text. Proceedings of The Fifth Annual Symposium on Computer Applications in Medical Care, 269-273.

Winograd, T. (1983) *Language as a Cognitive Process.* Menlo Park, CA: Addison-Wesley

Uhr, L (1966) *Pattern Recognition, New York: John Wiley and Sons.*

Woods, W. (1970) Transition Network Grammars for Natural Language Analysis. *Communications of the ACM* 13(10): 591-606

Appendix: Chinese Example

The following Chinese examples contain translations of the English examples. When we typed in the Chinese, we made several spelling mistakes. We left them in because the misspelled words ended up in the same categories as the correct spellings. This gives rise to implications for spelling categories.

In the last net, in category SYNTAX3, "ye" should be "yi." In SYNTAX16, "chengongde" should be "chenggongde." In SYNTAX12, "zhoaji" should be "zhaoji."

Chih-king Yang has verified that the transition net produced for Chinese is a valid one.

The Chinese example was provided by Chih-king Yang. This work would not have been possible in so short a time without the PIKS tools developed at PRC by Bruce Loatman. The PIKS tools implement Minsky's frame-based system for knowledge representation and provide graphic tools for displaying and editing frames and lattices of frames. The conversations, assistance, and suggestions by Julie Onna helped a great deal.

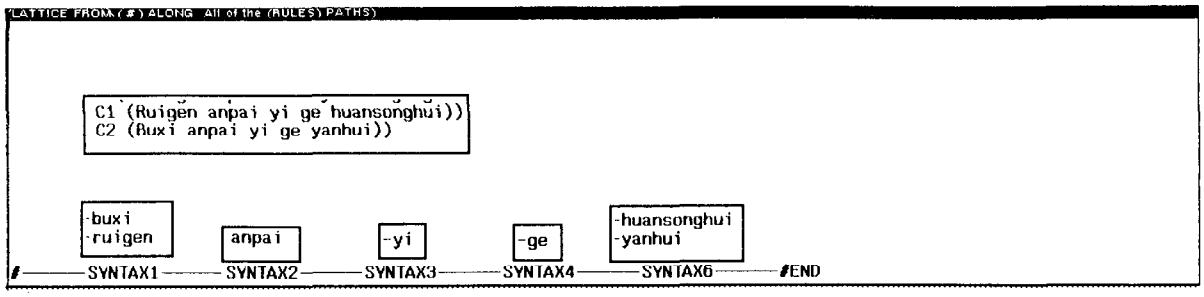
Appendix continued on next page

Announcement

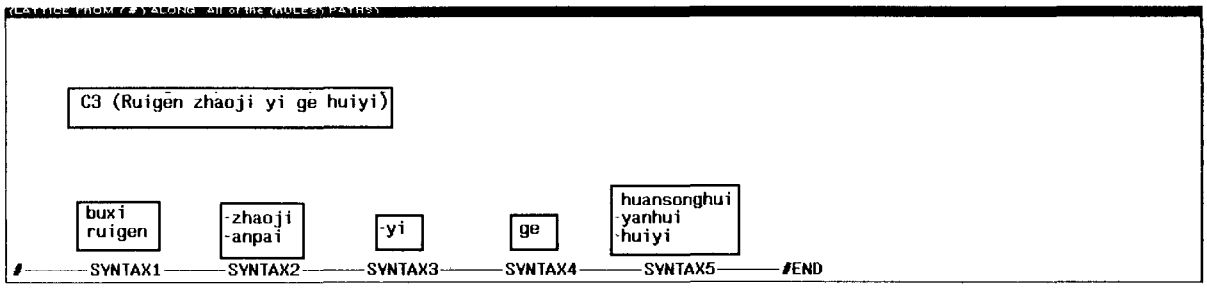
Hewlett-Packard announced February 21st that it will grant $50 million worth of advanced engineering workstations, along with computer software for the development and application of artificial intelligence technology, to selected universities in the United States.

MIT was named as the first of ten to twelve schools to receive twenty workstations per year for three years. HP will support all hardware and software for one year after it is granted.

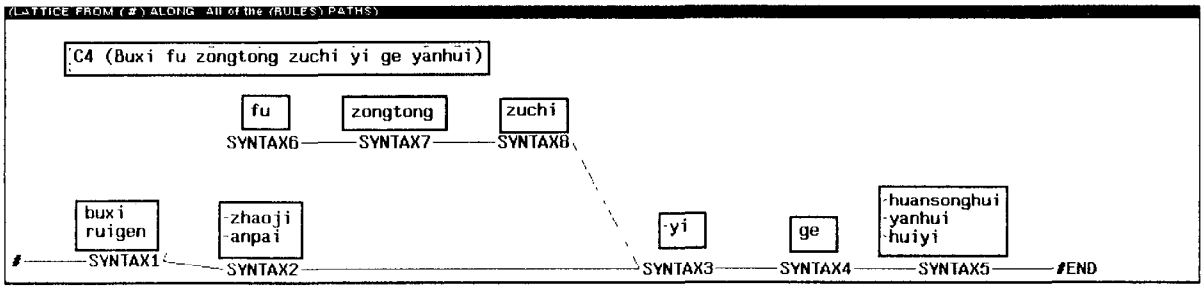
Universities interested in this program should contact.
Seth Fearey
HP Laboratories
1501 Page Mill Road
Palo Alto, California 94304
(415) 857-7409



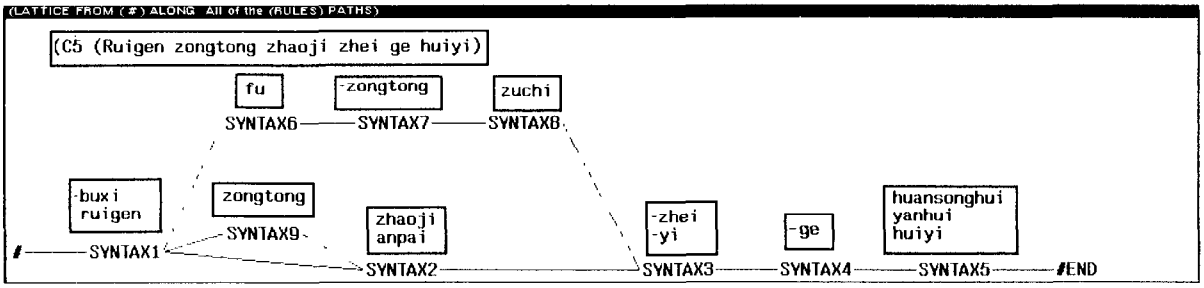
Examples C1 and C2.



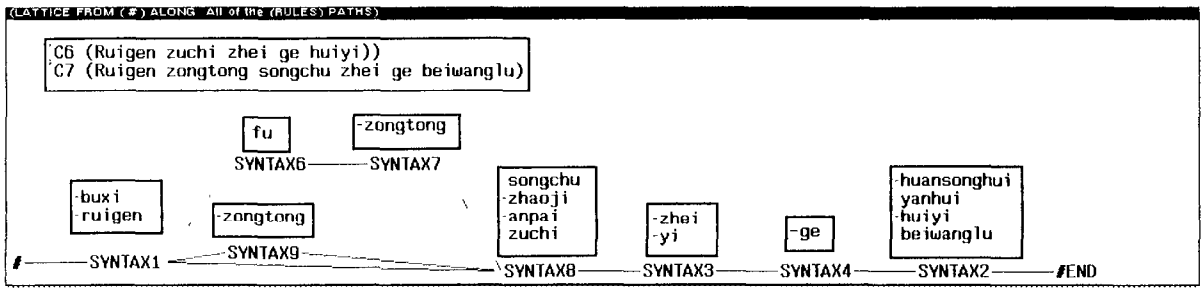
Example C3.



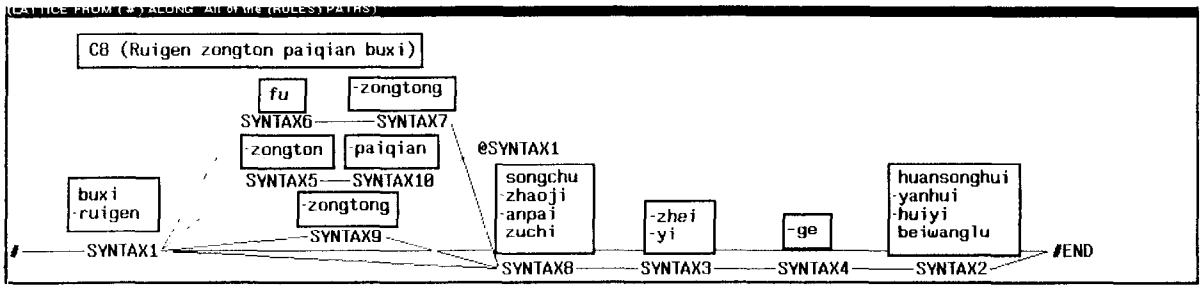
Example C4.



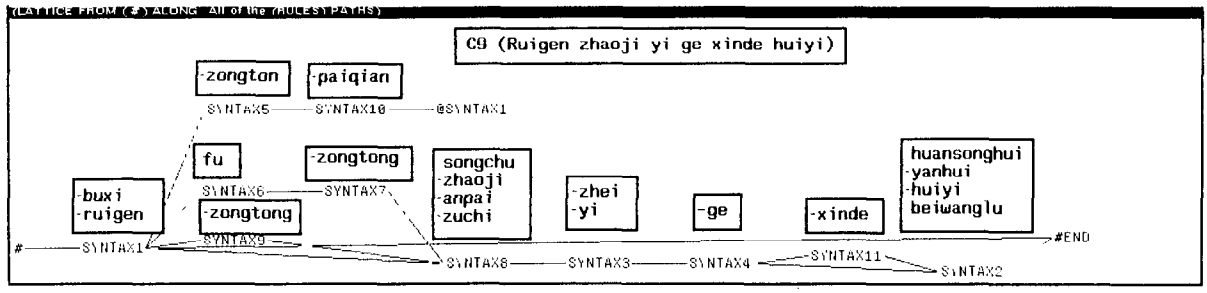
Example C5.



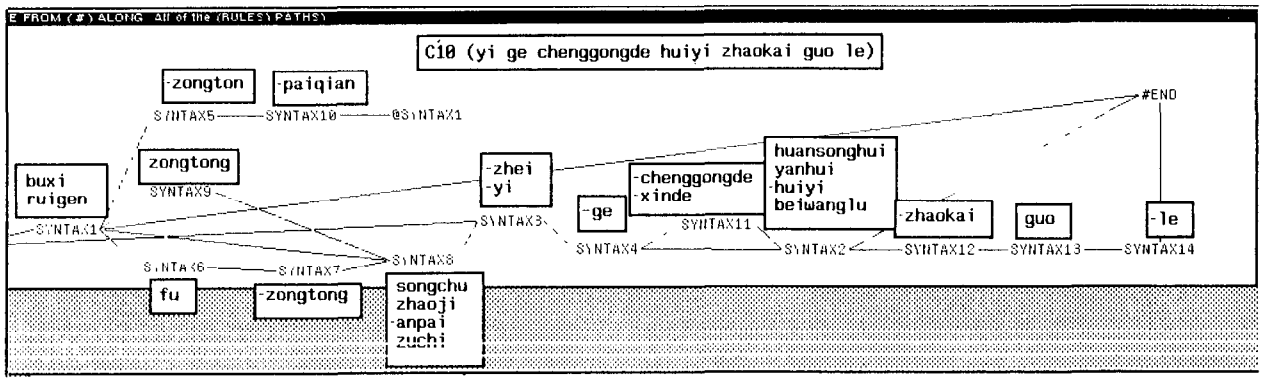
Examples C6 and C7.



Example C8.



Example C9.



Example C10.

```
C11 (Yi ge jichude huiyi zhaokai guo le))
C12 (Yi ge zuihoude huiyi zhoaji guo le))
C13 (zuotian yi ge chengongde zuihoude huiyi
          zhaokai guo le))
C14 (jintian yi ge chengongde huiyi zhaokai guo
          le))
C15 (ye ge huansonghui anpai guo le))
C16 (zhei ge xinde huiyi zhaoji guo le))
C17 (zhei ge zhengdangde huiyi zhaoji guo le))
C18 (zhei ge anpaide huiyi zhaoji guo le)))
```

Examples C11-C18.