

SEMANTIC COOCCURRENCE NETWORKS  
AND  
THE AUTOMATIC RESOLUTION  
OF  
LEXICAL AMBIGUITY  
IN  
MACHINE TRANSLATION

APPROVED BY  
DISSERTATION COMMITTEE :

---

---

---

---

---

Copyright

by

Dan Walter Higinbotham

1990

To my father,  
Oliver A. Higinbotham, Jr.

**SEMANTIC COOCCURRENCE NETWORKS**

**AND**

**THE AUTOMATIC RESOLUTION**

**OF**

**LEXICAL AMBIGUITY**

**IN**

**MACHINE TRANSLATION**

by

Dan Walter Higinbotham, B.S., M.S.

**DISSERTATION**

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

**Doctor of Philosophy**

**THE UNIVERSITY OF TEXAS AT AUSTIN**

December 1990

## ACKNOWLEDGEMENTS

The research reported here has been assisted by many, through contribution of time, data, and advice. I would especially like to express appreciation to:

My committee, Dr. Robert E. Wall, Dr. Robert F. Simmons, Dr. Winfred P. Lehmann, Dr. Winfield S. Bennett, and Dr. Kent Wittenberg, for their encouragement and direction.

Larry Gibson and Executive Communications Systems, Inc., for allowing me to use computer facilities and company time to do this research.

Siemens Corporation, for providing a sample of parallel English and German texts, on which the algorithms described here were first tested.

The Humanities Research Center at Brigham Young University, for providing a machine readable copy of the Longman Dictionary of Contemporary English.

Alan Melby, for providing the parallel English and French texts used in Chapter 5.

My wife, Barbara, for her loving patience and support; my mother, Birdie, for twice giving me life; and my father, Oliver, for his encouragement and love.

SEMANTIC COOCCURRENCE NETWORKS  
AND  
THE AUTOMATIC RESOLUTION  
OF  
LEXICAL AMBIGUITY  
IN  
MACHINE TRANSLATION

Publication No. \_\_\_\_\_

Dan Walter Higinbotham, Ph.D.  
The University of Texas at Austin, 1990

Supervisors: Robert E. Wall  
Robert F. Simmons

Lexical ambiguities of major class words were categorized according to approaches by which they might be correctly resolved; manual analysis of sample texts revealed that none of the methods previously proposed was sufficient to resolve more than a fraction of the ambiguities found in real text. An algorithm for combining the methods was proposed, including a new method that uses cooccurrence statistics from parallel texts to train neural networks to identify context words at arbitrary distances that can trigger less frequent senses.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTSv

ABSTRACTvi

Chapter

1. THE RESOLUTION OF LEXICAL AMBIGUITIES	1
2. PREVIOUS APPROACHES	5
3. AMBIGUITIES IN REAL TEXT	64
4. SPREADING ACTIVATION AND TRIGGER WORDS	80
5. SEMANTIC COOCCURRENCE NETWORKS BASED ON PARALLEL TEXT CORPORA	100
6. RESOLUTION OF LEXICAL AMBIGUITY IN MACHINE TRANSLATION	121

Appendix

A. TEXTS WITH CLASSIFIED AMBIGUITIES	145
B. A PROGRAM FOR WEISS RULES AND TRIGGER COUNTING	172
C. A PROGRAM FOR CREATING A SEMANTIC COOCCURRENCE NETWORK	199
D. A PROGRAM FOR EVALUATING SEMANTIC COOCCURRENCE NETWORKS	221

BIBLIOGRAPHY 250

## Chapter 1

### **THE RESOLUTION OF LEXICAL AMBIGUITIES**

Machine Translation is difficult at best. One of the more challenging aspects is the resolution of lexical ambiguities based on context. This study concentrates on ambiguities in major class words (nouns, verbs, adjectives, and adverbs).

Even the earliest workers in machine translation realized that the vagueness of word boundaries and the mismatch in cultural perspective would make word selection difficult in a second language. They suggested limiting the subject area, so that words would be more likely to have technical meanings. They also suggested statistical methods based on cooccurrence, but these were difficult to implement. Some systems simply translated the most frequent sense of each word, or included all of the possibilities in the output document. Some projects had an ad hoc rule on each word in the lexicon. Several groups thought it might be possible to use the semantic groupings in a thesaurus to associate word senses. Others tried to classify selectional restrictions that verbs have for



their arguments and modifiers have for their heads. More recently, people have begun to build knowledge bases, hoping that some kind of spreading activation along connections in the knowledge base might simulate contextual selection. Others have looked to the budding technology of neural networks for a solution.

Of the methods that have actually been attempted, many have led to years of long, tedious labor followed by disappointment. Proposals of one sort or another fill the literature, but reports of results are sparse.

One of the purposes of the research described in this thesis was to determine what kinds of lexical ambiguities actually occur in real text, and which classes of proposed methods might be expected to work for them. The results of a manual study of actual text were rather surprising. None of the proposed methods resolved more than about 45% of the ambiguities correctly by themselves, and some of the most highly touted theories proposed in the literature averaged no more than 15%. It became clear that no single method would ever work well

enough by itself. The only logical approach was a combination of the methods.

Even after using syntactic clues, putting idioms in the dictionary, flagging words with technical meanings, using selectional restrictions, and remembering the sense of the word selected in a previous occurrence, there remained a difficult class of problems. The disambiguation of some words seemed to depend on words with only loose syntactic relations if any, arbitrarily far away; the influence of these words somehow triggered the selection of the appropriate sense. In the absence of such words, a default sense was used. The thesaurus and spreading activation methods were attempts to handle this kind of ambiguity, but the results of such methods are either negative or unknown.

A new method was therefore invented to work with trigger words and default senses. A toy example showed that a neural network might set up a feature representation of implicit cooccurrence classes. Training data was then extracted from a large sample of parallel texts in two languages, and the same method that worked in the toy example was

shown to have positive results on the larger data set.

The set of methods for lexical disambiguation were then studied, including the new semantic cooccurrence network approach, and an algorithm combining all of the methods was developed.

The contribution of this thesis consists mainly in the categorization of types of lexical ambiguity in the manual study, the new neural network method, and the organization of all of the methods into an algorithm for lexical disambiguation.

Chapter 2 discusses previous methods. Chapter 3 describes the manual study of text and classification of types of lexical ambiguities. Chapter 4 introduces the neural network method and describes the toy example and an initial experiment indicating the possible success of the method. Chapter 5 describes an experiment in which training data for the neural network was extracted from a large sample of raw text. Chapter 6 discusses the algorithm that combines all of the methods.

Appendix A presents the texts analyzed in Chapter 3. Appendix B presents a program that resolves ambiguity in parallel texts based on lists

of trigger words. Appendix C gives a program for creating semantic cooccurrence networks. Appendix D is a program for evaluating semantic cooccurrence networks.

## Chapter 2

### PREVIOUS APPROACHES

#### 2.1 Introduction

This chapter will be a survey of some of the approaches that have been taken to resolving lexical ambiguity in machine translation.

The earliest approaches were based on a suggestion in a memorandum by Warren Weaver in 1949, that statistical studies of collocation frequencies could at least partially solve the problem. Section 2.2 discusses some of the studies and approaches that arose from this idea.

Some of the early workers preferred a more direct approach, and considered each word to be a unique problem. They therefore encoded unique disambiguation rules on each word in the lexicon. An example of this approach is the CREWS project; a similar but more sophisticated version of this approach was Small's word expert parser. These are discussed in section 2.3.

Others suggested using the semantic categories of Roget's Thesaurus as the basis of contextual

analysis, and some worked on developing their own thesaurus especially for machine translation. Some of the assumptions underlying their work, and some details of their methods, are discussed in section 2.4.

Wilks' Preference Semantics emphasized selectional restrictions, but also recognized that such restrictions should be treated as preferences that could be violated. Later, Fass' Collative Semantics used the idea of selection restrictions as preferences in the context of semantic feature structures. Some of the important ideas in these approaches are mentioned in section 2.5.

Marker passing systems that spread activation along connected paths in knowledge-based frame structures are briefly considered in section 2.6.

Studies of spreading activation in neural networks have led to local connectionist approaches, exemplified by Cottrell's work, and distributed connectionist approaches, such as Kawamoto's work. These are briefly examined in section 2.7.

Section 2.8 will be a short summary of the various approaches to resolving lexical ambiguity.

## 2.2 Statistical Approaches

In July of 1949, Warren Weaver sent a memorandum to about 30 friends, suggesting the possibility of using computers to translate text from one language into another. This group and their contacts began research in machine translation, and much of the work in natural language processing (and artificial intelligence) has its roots in the early efforts of this group. Since Weaver's memo was such an important influence in the history of machine translation, it is interesting to note Weaver's feeling about the problem of ambiguity.

Weaver began the memo by quoting from a letter he had written to Norbert Wiener of MIT:

Recognizing fully, even though necessarily vaguely, the semantic difficulties of multiple meanings, etc., I have wondered if it were unthinkable to design a computer which would translate only scientific material (where the semantic difficulties are very notably less), and even if it did produce an inelegant (but

intelligible) result, it would seem to me worthwhile.

(Weaver 1967, p.  
190)

Wiener's reply, also partially included in the memo, contained the following:

... I frankly am afraid the boundaries of words in different languages are too vague and the emotional and international connotations are too extensive to make any quasi mechanical translation scheme very hopeful.

(Weaver 1967, p.190)

It is clear from these quotes that some of the major doubts about the feasibility of translation by machine were centered on the problem of the vagueness of word boundaries and multiple meaning, i.e. lexical ambiguity.



### 2.2.1 Domain Limitation

Weaver's first suggestion for overcoming the problem was to limit the domain of the input text (an idea which has come to be known as the "sublanguage" approach). The word 'log', for example, would almost certainly need to be translated differently if it meant 'logarithm' than if it meant 'a piece of wood'; by limiting the text to the sublanguage of mathematics, one could be certain in the vast majority of cases that the intended meaning was 'logarithm'. A computer lexicon could be created specifically for mathematical text, and the 'wood' meaning would not need to be included at all. In mathematical text, this would produce correct translations for most words that have a sense with a mathematical meaning. It is not so clear that it would be helpful for words with several meanings, none of which are particular to mathematics.

Nevertheless, domain limitation is still the most frequently used approach in actual systems. Even knowledge-based machine translation projects

often resolve most cases of lexical ambiguity by using domain-specific knowledge bases.

### 2.2.2 Contextual Windows

Weaver's other approach to the lexical ambiguity problem was based on the fact that a word may be ambiguous in isolation, but given sufficient context, should be unambiguous. He suggested using a window of context of  $N$  words on either side of the ambiguous word. The value of  $N$  would vary with the type of text, the grammatical category, and perhaps individually for each word. He says, however, that

... It would hardly be practical to do this by means of a generalized dictionary which contains all possible phrases  $2N+1$  long: for the number of such phrases is horrifying, even to a modern electronic computer. But it does seem likely that some reasonable way could be found of using the micro-context to settle the difficult cases of ambiguity.

(Weaver 1967, p. 195)

The year after the appearance of Weaver's memo, Abraham Kaplan wrote a paper called "An experimental study of ambiguity and context." His basic purpose was to discover the value of N, that is, to discover how much context was necessary for humans to disambiguate ambiguous words. He took examples of 140 (inflected) words in context from books on mathematics. Seven types of contexts were used: (1) One word before the ambiguous word, (2) one word after, (3) one word before and one word after, (4) two words before, (5) two words after, (6) two words before and two words after, (7) and the whole sentence. All 140 words were presented with a list of ten possible meanings each. Only "clearly distinguishable meanings" of a single grammatical category were used. Words which had less than ten "clearly distinguishable meanings" in the dictionary were supplemented by false senses of the same grammatical category; the average was 5.6 correct senses per word. Each translator was presented each of the words in isolation and asked to pick which meanings were possible for the word; the average percentage of senses classified correctly was 70%. The contexts were then mixed up so that each

translator disambiguated each word once, but each of the translators had the word in a different one of the seven contexts described above. The contexts for each translator were mixed, so that each translator had a mixture of the types of context. The most important of Kaplan's conclusions were

... A context consisting of one or two words on each side of the key word has an effectiveness not markedly different from that of the whole sentence.

... Under optimal conditions ... ambiguity is reduced from ... about 5½ senses to about 1½ or 2.

(Kaplan 1955, pp. 46-47)

Optimal conditions were obtained (1) when the translator was trained in the subject of the text (mathematics); (2) when the context included at least one word on each side of the ambiguous word; and (3) when the context words were content words (nouns, verbs, adjectives, and adverbs) rather than function words (prepositions, articles, etc.).

Kaplan's results were encouraging to many, because the study was taken as evidence that a local context (of two to four words) was sufficient to resolve lexical ambiguity; the study had shown that such a context was nearly as useful as the whole sentence for the purpose of resolving ambiguity. It is odd that such optimism was generated by the article, however, since the translators had only achieved 70% accuracy picking possible senses of words in isolation, and even with the whole sentence context, ambiguities were reduced to "about 1½ or 2" of the "clearly distinguishable meanings." Since only one of the "clearly distinguishable meanings" would probably be correct in the original text, that means that a computer with the reasoning capability of a human, using only a local or sentence context, would choose the correct meaning, and therefore the correct translation, between 50% and 67% of the time.

Although Kaplan's results were taken to mean that the size of Weaver's context variable N need not be greater than 2 words on each side of the word in question, the method of storing all such contexts was never applied directly. Even setting N to 1, it

would require a matrix of size  $M^3$  for a vocabulary of size  $M$ , where the value at any position  $(X,Y,Z)$  in the matrix would be the most frequently correct sense number of word  $Y$  in the local context " $X Y Z$ ". For example, in the phrase 'put into a bank account', ignoring the function words 'into' and 'a', it is almost certain that the meaning of 'bank' ( $Y$ ) in the presence of 'put' ( $X$ ) and 'account' ( $Z$ ) is the financial meaning rather than the geological meaning; the sense number of the financial meaning would be the value of the matrix at position ('put','bank','account'). Assuming that values in the matrix could be automatically determined based on frequencies in a text tagged with correct sense numbers, and assuming a moderate vocabulary of 10,000 words, this would require a matrix of a trillion entries, and, at a millisecond per entry, would take over thirty years to fill. In order to insure accuracy, the tagged text would have to include multiple examples of all possible phrases of 3 words long (ignoring function words) in realistic contexts. Sense-tagging an input text with even a moderate coverage of contexts would take a staggering amount of time, even after the "clearly



solution (sulfuric/sulfate) acid; obtained such  
 (means/way) syrup (process/convert) (on/at)  
 (wine/tartaric) alcohol. (According  
 to/Along/In accord with) other process  
 saccharification (accomplish/carry out) (on/at)  
 cold action very strong (sp. weight 1.21)  
 (salt/hydrochloric) acid. After removal acid,  
 remain solid product being used (as/how)  
 (food/forage) (medium/means).

(Perry 1955, p.

18)

However, "The most frequent criticism was levelled at the excessive number of alternatives given ..." and even in a word-for-word translation, "... the proper selection of English correspondents is by far the major problem facing a reader ...." (Gould 1957, p. 14).

As a result,

Booth and Richens proposed printing only the symbol 'z' to indicate an unspecified word; others have proposed leaving the word



untranslated, and others have proposed always giving the most common translation.

(King 1956, p.  
38)

The most common translation would give the best probability of having chosen the correct translation and should therefore be fairly successful.

Sometimes the most common translation was decided through introspection by the dictionary maintenance people, but others felt that the decision needed to be based on semantic frequencies determined from sufficient examples of actual text.

A trans-semantic frequency count is a listing of the words of the source language, together with the various possible renderings of each in the target language, and the frequency of occurrence of each of the latter. Such a listing would resemble a normal translation dictionary, with the addition of information, probably in the form of percentages, giving the frequency of occurrence of each meaning in the target language.

(Pimsleur 1957, p. 11)

Given such information, only the most common translation for each word would need to be entered in the computer's lexicon, and if texts to be translated were consistent with the original text, incorrect translations would be expected to be in the minority. For example, according to one study, the French word 'de' should be translated as 'of' in English about 68% of the time (King 1956, p. 39), so if the machine always translated 'de' as 'of', it would also be correct about 68% of the time.

#### 2.2.4 Cover Words

A variation in the use of most common translation was the use of "cover-words", which were words "of relatively high semantic frequency which can be used in place of words of lower semantic frequency, with little possibility of misinforming the reader." (Pimsleur 1957, p. 13) A target language "cover-word" would be a word of relatively broad coverage, whose available meanings could "cover" most of the meanings of other more specific

translations of a given source word. For example, the German word 'schwer' may mean (among other things)

heavy	as in 'ein schweres Stein',
	'a heavy stone'
laden	as in 'Das Dach ist schwer von Schnee',
	'the roof is laden with snow'
difficult	as in 'Das fällt mir schwer',
	'I find that difficult'
unfortunate	as in 'Er hat ein schwere Schicksal',
	'he has an unfortunate fate'

In the second case, it would be acceptable to say 'the roof is heavy with snow', and in the last case, it would be understandable to say 'he has a heavy fate'. By choosing the broader words 'heavy' and 'difficult', the more specific words 'laden' and 'unfortunate' might not need to be included in the computer's lexicon (example from Pimsleur 1957, pp. 12-13). Even though such "cover-words" in the target language would be relatively more vague in isolation, the human reader would not notice the

ambiguity in context. Determination of cover-words could not be based only on frequency studies of real translations, since human translators would often choose the more specific translation, but frequency studies could give a good idea of what the possible cover-words might be.

#### 2.2.5 Frequency-based Technical Tagging

Frequency studies could also be based on various types of text.

Alternative frequencies should also be given for various subject areas, scientific, military, etc.

(Pimsleur 1957, p. 11)

The "most common translation" could therefore be refined to be "the most common translation" in a particular domain (sublanguage).

Researchers at the University of Washington in 1958 tried this approach by categorizing science into about seventy subfields.

A few of the words in the target language were then tagged with numbers representing the particular field in science in which they occurred almost exclusively. Since the number of words that could be tagged in this way was small, the method was found to be successful in a very small number of cases to which it was applied.

(Madhu and Lytle 1965, p.

9)

A more refined approach was later developed, which grouped these seventy subfields into ten technical groups, namely

Group I Mathematics, Physics, Electrical  
Engineering, Acoustics, Nuclear  
Engineering

II Chemistry, Chemical Engineering,  
Photography

III Biology, Medicine

IV Astronomy, Meteorology

V Geology, Geophysics, Geography,  
Oceanography

- VI Mechanics, Structures
- VII Mechanical Engineering, Aeronautical Engineering, Production and Manufacturing Methods
- VIII Materials, Mining, Metals, Ceramics, Textiles
- IX Political Science, Military Science
- X Social Sciences, Economics, Linguistics, Etc.

Target language texts were examined, and each paragraph in the text was tagged as belonging to one of the ten groups. A count was made of how often each content word in the sample occurred in a paragraph labelled by each group. The chance that target word  $T_k$  would occur in a paragraph that should be classified as falling within subject group  $N$ , was based on the sample text, and was given as

$$p(T_k, N) = \frac{\text{(times } T_k \text{ found in a group } N \text{ paragraph)}}{\text{(total words in target sample text)}}$$

The chance that a given paragraph in a new text should be classified as belonging to subject group N was then taken to be

$$p(N) = p(W_1, N) + p(W_2, N) + \dots + p(W_k, N)$$

where it was assumed that  $W_1$  through  $W_k$  were words in the paragraph that had a single meaning. For a word with more than one possible translation, the best translation was selected by using a figure of merit, defined by

$$f(T_k) = \frac{p(N) \cdot p(T_k, N)}{N}$$

In other words, the figure of merit of a given target word was the chance that the current paragraph was of subject group 1 times the chance that the target word would be used in a text of that type, plus the chance that the current paragraph was of subject group 2 times the chance that the target word would be used in a text of subject group 2, and so forth. The figure of merit would be calculated for each possible translation, and the target word with the highest figure of merit would be the translation chosen.



For example, consider the phrase "structure/building of an algae colony", where the only ambiguity is in the choice between 'structure' and 'building'. Based on a small text corpus, the study showed the chance that the word 'structure' would occur in a paragraph of group I (multiplied by a scaling factor) was  $P('structure', I) = .1$ . Similarly,

$P('structure', III) = 1.9,$	
$P('structure', V) = .8,$	$P('structure', X) = .3,$
$P('building', I) = .1,$	$P('building', V) = .4,$
$P('building', VI) = .1,$	$P('building', IX) = .1,$

$P('algae', III) = .5,$   $P('algae', V) = 1.4,$  and the ities for these words with other groups were 0. If the phrase is considered a text unit, the only unambiguous word known to the text corpus was 'algae'. Therefore, the chance that this phrase should belong to group III is  $P(III) = .5,$   $P(V) = 1.4,$  and  $P(n) = 0$  for the other groups. Therefore, the figure of merit for 'structure' would be  $f('structure') = (.5 * 1.9) + (1.4 * .8) + (0 * .1) + (0 * .1) = 2.07,$  and the figure of merit for 'building' would be  $f('building') = (.5 * .1) + (1.4 * .4) + (0 * .1) + (0 * .1) = .61.$  Since the figure of merit for 'structure' is greater

than that for 'building', 'structure' would be the word chosen.

In a small test of 21 Russian sentences translated into English using this technique, the correct alternate was chosen in 87% of the cases (Madhu and Lytle 1965, pp. 10-12).

#### 2.2.6 Category Counting

Walker and Amsler suggested using frequencies of occurrence of domain codes in the Longman Dictionary of Contemporary English (LDOCE). In the typesetting tape, certain senses of some words are given a four-character domain code. The first two characters are field codes, and the second two characters are subfield codes; there are 120 fields, 212 subfields, and about 2600 domain code combinations actually used. Given a segment of text with ambiguous words, the domain codes on each of the senses of each of the words in the text are assembled and counted. The domain code with the highest frequency is assumed to be the main subject of the text; word senses which are marked with this domain code are chosen as the correct senses of the

given words in the text. This reduces the number of ambiguous word in the text. The next pass works exactly as the first pass, assembling domain codes of only those words which are still ambiguous. Again, the domain code with the highest frequency wins, and word senses marked with that code are chosen. The process continues in cycles.

Although this process yielded some interesting results for classifying the domain of some documents (the most frequent of all the codes), there were some problems. Walker and Amsler reported that in an eight million word corpus, only 23% of the words were in the LDOCE (Walker 1986). Even of the words that do occur in the dictionary, most of the senses are not marked with domain codes. In cases where only some senses of a word are marked, the algorithm may not choose the correct sense, and, of course, in cases where none of its senses are marked, the algorithm makes no choice at all. It is not clear what to do when more than one code ties for the lead at any point in the algorithm, especially at the end, when competing codes occur only once or twice. One must also be careful about the quantity of text used, so that enough codes can be

assembled to make choices, but there is not so much text that too many topics are addressed.

Because of the lack of domain codes on most senses in the dictionary, no data is available on the potential success rate of this method.

### 2.2.7 Proximity Lists

Stephen Weiss developed an algorithm for determining contextual rules for disambiguation based on proximity of context. Each ambiguous word has an associated list of contextual rules. Each contextual rule consists of a context word and a sense of the ambiguous word. When the ambiguous word A occurs in the context

... 9 7 5 3 1 A 2 4 6 8 10 ...

where each of the numbers represents a context word, the context words are checked in the order shown, beginning with 1. Each context word is checked to see if it occurs in a contextual rule for the ambiguous word; if it does, the sense listed in the contextual rule is the one chosen.

The set of contextual rules for an ambiguous word is generated based on a training text in the following way. For each instance of the ambiguous word A, each context word C is examined in the same order as above. There are three possibilities:

- 1) If C is mentioned in the contextual rules, and C is associated with the correct sense of A, the training jumps to the next instance of A in the text.
- 2) If C is mentioned in the contextual rules, and C is associated with an incorrect sense of A, the contextual rule containing C is removed from the list, and C is added to an exclusion list; then the next context word is examined.
- 3) If C is not mentioned in the contextual rules generated so far, and C is not on the exclusion list, a new contextual rule is added with C and the correct sense of A, and the training then jumps to the next instance of A in the text; if C is on the exclusion list, the next context word is examined.

The training text can be examined in this way for every ambiguous word of interest, generating contextual rules for each one.

Weiss tested this algorithm on three words, with 180 instances each. After training each word on its first 80 instances, Weiss found that the correct sense was chosen 97% of the time in the remaining 100 cases (Weiss 1973).

Since this is such an impressive result, this algorithm was implemented and tested as part of this dissertation, using the program in Appendix B; 80% of the text used in Chapter 5 was used as training text, and the remaining 20% was used for testing. Although the algorithm resolved ambiguity nearly perfectly for the training text (correctly resolving over 99.5% of the cases), the algorithm only achieved 63.4% on the approximately 1250 ambiguous words in the test text, whereas simply choosing the most frequent sense in the sample text would have achieved 67.2%.

### 2.3 Word Experts

Some researchers felt that waiting for detailed frequency data of any kind was impractical with the resources they had available. They felt that each word was a unique entity, and every ambiguous word was a unique problem. The best way to solve such a problem, they felt, was to attack it head on.

The CREWS project developed a special language in which to describe possible local conditions that might affect lexical selection. This language allowed them to check for certain grammatical categories, or specific words, a certain number of words before or after the ambiguous word, or within a certain range of the words near the ambiguous word. In this way, much of the syntactic and semantic processing was mixed in a way that was unique for each word. A word such as 'bank', for example, would have a specific rule to look for words in its immediate vicinity such as 'river' or 'money'. Failing to find any such words in the context, the most likely sense would be chosen as a default. Naturally, such a system continually

needed to be refined, until conditions on commonly used words became extremely difficult to maintain.

In a similar vein, Small's Word Expert Parser simply executed the procedures that were associated in the lexicon with each word in the sentence. Each procedure was a discrimination net. For example, a discrimination net for the adjectival use of the word 'deep' would be as follows:

(a) What view best describes the concept?

PERSON -----> (1) Intellectual

ART-OBJECT -----> (2) Ethereal

VOLUME: (b) Which view best describes the concept?

AIR-VOLUME -----> (3) Large Vertical Distance  
Through Air

WATER-VOLUME ----> (4) Large Vertical Distance  
Through Water

By answering the questions (a) and (b), the computer could arrive at one of the senses marked (1)-(4). By asking questions, and receiving answers based on the procedures of nearby words, such nets could be traversed until a unique meaning was reached.



Unfortunately, the procedures were long and cumbersome; the one for the word 'throw', for example, was six pages long (Hirst 1987, p. 83). Some of the questions were of a more global nature, and plans were made to define a conceptual proximity measure based on a semantic network. Developing the procedures was so time consuming, however, that questions of a global nature were just relegated to interaction with an on-line user, thus begging all the more important questions of automatic disambiguation (Adriens and Small 1988, p.18).

#### **2.4 Approaches Based on a Thesaurus**

Word by word definition of unique procedures was time-consuming, inelegant, and mostly ineffective. Scientific classification of words into ten groups or even seventy subfields was useful for resolution of some lexical ambiguities, but many ambiguities were left unresolved and unresolvable. It seemed clear that what was needed was a more thorough scheme of semantic classification that was better at capturing generalizations. The organization of Roget's Thesaurus, which lists

content words under more than 1000 concept headings, presented the possibility of being just such a semantic classification.

#### 2.4.1 Sequences of Thesaurus Categories

Roderick Gould of Harvard noticed that different senses of a word in a dictionary usually correspond to different concept categories in Roget's Thesaurus. His idea was to store lists of possible semantic category sequences. These lists would be based on an automatic frequency analysis of a large sample of source language text, in which each word had been manually tagged with the Roget concept category number of the sense in which it was being used. At first, he suggested expanding Roget's classification to include uses of prepositions and other function words, but soon found this extension to be problematic. He subsequently suggested storing possible sequences of content word semantic categories. He did not report any implementation of his method, but only noted that it would have to be based on a large text sample in order to be effective. He did not present

a specific algorithm for resolving lexical ambiguity on the basis of the frequencies stored, but noted it would be complicated by the fact that even if a stored two category sequence selected senses for two adjacent words, those senses could be disallowed by the lack of a sequence for the first word and its preceding word, or the second word and its following word (Gould 1957, pp. 15-27). It should also be noted that a matrix for sequences of two categories would require a million entries; Kaplan's data suggests that at least sequences of three categories would be required, and a matrix storing three category sequences would need a billion entries.

#### 2.4.2 Synonymy in a Thesaurus

The Cambridge Language Research Group thought of using the thesaurus in a more practical and immediate way. The idea was that a text about a certain subject or situation would probably employ a number of words that could be classified under the same concept heading in the thesaurus. For example, in the phrase 'flowering plant', 'flowering' and 'plant' are each listed in the thesaurus under a

number of different headings, but the only heading under which both are listed is 'vegetable'; we therefore assume that the correct sense of each word in the current context is the 'vegetable' sense (Masterman 1957, p. 36). It was soon discovered, however, that the method was only partially successful, and it was assumed that this was due to problems with Roget's Thesaurus. Some words were not listed at all, and other words were not listed in all of their senses.

The solution was to develop a new thesaurus more appropriate for use in machine translation. It was assumed that in this new thesaurus, "... if a set of words all come under one heading, they must be semantically related, and if a number of words are semantically related to one another, they will come together under some heading." (Sparck-Jones 1965, p. 97) Some of the relations examined were hyponymy, antonymy, incompatibility, collocation, and synonymy. It was claimed that collocation was not a genuine linguistic relationship, antonymy would be hard to determine empirically, and hyponymy and incompatibility could not be satisfactorily defined, so synonymy was the only viable semantic

basis for determining the vocabulary structure of a language, and, therefore, the contents of the new thesaurus. Two words were held to be synonymous if they had two uses that were synonymous, i.e. if they could be mutually substituted in some sentence without changing the meaning. The set of words that could be mutually substituted in a given context was called a row. Rows were then "clumped" together if each row in the clump was more "similar" to members of the "clump" than to nonmembers. A pair of rows was given a similarity score based on the number of words they shared, such that if a shared word had 10 senses, a similarity score of .1 was added based on that word; if a shared word had only two senses, .5 was added. In this way, rows were clumped together, and each clump was given a heading and listed as a new category in the thesaurus.

For example, rows could be abstracted from the examples in the definitions of the Oxford English Dictionary. One definition of the word 'toil' is "Severe labour; hard or continuous work or exertion which taxes the bodily or mental powers." An associated example is "You are many of you accustomed to toil manual; I am accustomed to toil

mental." In the example, 'toil' could be replaced by 'labour' without loss, so one possible row would be

toil labour

One definition of 'task' is "A piece of work that has to be done; something that one has to do (usually involving labour or difficulty); a matter of difficulty, a 'piece of work'." One example associated with this definition is "He had taken upon himself a task beyond the ordinary strength of man." As the definition might suggest, 'task' could be substituted with 'labour', so another row would be

task labour

These two rows share a word, 'labour', so if 'labour' had 5 senses, these two rows would be given a similarity score of .2. Assuming a clump threshold of .15, these rows could be grouped together as a clump, perhaps with the heading 'labour', and could be added to the thesaurus as a

new category (see Sparck-Jones 1965, pp. 102-103, for a less simplified version of this example).

There was also a notion of semantic distance between words. If words A and B occurred together in the same row, they were said to have distance 0 (i.e. they were synonymous). If a word C occurred with word A in some row, and with word B in another row, but A and B never occurred in the same row, A and B were said to have distance 1. This distance could also be used in resolution of lexical ambiguity, if nearby words were not found under the same heading in the thesaurus. (Sparck-Jones 1965, pp. 97-112)

For example, if there are rows

function capacity capability

function purpose business

business affair job matter

the semantic distance between 'function' and 'capacity' is 0 (they are "synonymous"), the distance between 'capacity' and 'business' is 1, and the distance between 'capacity' and 'job' is 2. In the sentence "the job was beyond his capacity", the

sense of 'job' would mean something closer to 'matter' than 'employment' or some other sense; the sense of 'capacity' would be closer to 'capability' than another sense like 'volume'. The proper senses would be chosen automatically because the semantic distance between them is only 2, whereas the distance between other senses (as determined in other rows) would presumably be greater than 2.

The intention was that words that were semantically related to each other would be found together under some heading, and that words that were semantically related would be found in the near context of an ambiguous word. Using the thesaurus method, however, two words were "semantically related" only if they were synonymous or nearly synonymous. It was evidently either not noticed or not regarded as important that two words could be semantically related without having that property. For example, in the sentence

I slept poorly, because I kept rolling over  
onto an exposed spring.

The word 'spring' could be a season of the year, a fountain of water, or a metal spiral. Clearly,



sleeping and bedsprings are related, but it is very doubtful that they or their synonyms could be substituted for each other in any context. They are related, but not by synonymy. It appears that the premise of the endeavor was somewhat flawed.

Furthermore, the procedure for building the new thesaurus was quite complicated, and the number and content of headings was completely dependent on the set of contexts that were used to determine the rows. Producing a new thesaurus with better coverage than Roget's required a tremendous amount of work, and there never had been a guarantee that it would be more suitable for machine translation than Roget's had been.

#### 2.4.3 Chains in the Thesaurus

Later work returned to the original Roget's Thesaurus, once machine readable copies became available. A model of chaining in the Thesaurus was developed by Robert Bryan at San Francisco State University. He defined chains of entries according to word-groups and categories (Sedelow 1986). The Thesaurus is divided into eight major classes,

comprised of 1042 groupings. Each grouping is divided into paragraphs, which are subdivided into semi-colon groups, which each contain a number of entries. The entries within one semi-colon group formed one of Bryan's categories. A word group consisted of all entries in the Thesaurus that were spelled the same; therefore, homographs belong to the same word group. Entries in the Thesaurus could be thought of as points in a two-dimensional matrix, with word groups as rows, and categories as columns. For example,

	c <sub>1</sub>	c <sub>2</sub>	c <sub>3</sub>
w <sub>1</sub>	e <sub>1</sub>		e <sub>2</sub>
w <sub>2</sub>	e <sub>3</sub>	e <sub>4</sub>	
w <sub>3</sub>	e <sub>5</sub>	e <sub>6</sub>	e <sub>7</sub>
w <sub>4</sub>			e <sub>8</sub>

A chain was simply a sequence of entries. Bryan defined ten types of chains, each a subtype of the previous type. Type-10 chains were the most constrained and of the greatest interest. These were defined in terms of word-links and category links. A pair  $\langle e_i, e_{i+1} \rangle$  was a word-link if both

entries belonged to the same word group (so they were in the same row of the matrix). A pair  $\langle e_i, e_{i+1} \rangle$  was a category-link if both entries belonged to the same category (so they were in the same column of the matrix). A type-10 chain was a sequence of entries such that word-links and category-links alternated, no entry was repeated, no word group or category was repeated, and every word-link and category-link was strong. A word-link was strong if the two categories it connected were also connected in the matrix by another word-link (though this second word link was not required to be in the chain being considered). A category-link was strong if the two words it connected were also connected by another category link (again, this second link was not required to be in the chain). This meant that in terms of the matrix, strong links always came in fours, called quartets, and were composed of four entries that formed a square in the matrix. For example, the entries  $e_1$ ,  $e_2$ ,  $e_5$ , and  $e_7$  in the matrix above form a square, so  $\langle e_1, e_5 \rangle$  and  $\langle e_2, e_7 \rangle$  would be strong category-links, and  $\langle e_1, e_2 \rangle$  and  $\langle e_5, e_7 \rangle$  would be strong word-links.

Evidence suggested that entries which had strong word-links belonged to the same sense of the word, and entries which had a strong category link were semantically related (perhaps synonymous according to the usage of Sparck-Jones as discussed in the previous section). Therefore, entries which could be connected in type-10 chains were expected to be semantically related. It was possible to partition the Thesaurus in such a way that only entries which could be connected by type-10 chains would be in the same group. This was done by Talburt and Mooney. They discovered that of the 199,427 entries in Roget's International Thesaurus, there were 113,963 distinct word groups. It turned out that 133,672 of the entries had no strong links with any other entry. The remainder of the Thesaurus contained 65,755 entries and was partitioned into 5966 groups. One of these groups contained 22,480 entries, or about one-third of the remaining entries. Although some of the entries in this group were instances of the words 'cozy', 'intimate', 'snug', 'familiar', 'close', 'near', 'tight', 'thick', and 'compact', the group also contained instances of words such as 'vile' and

'humble'. Of the other groups, 3373 were formed from a single quartet, so they each consisted of four entries. The other groups ranged in size from six entries to 229 (Talburt 1990).

Although this work was interesting, it is not immediately obvious how to use the results for the resolution of lexical ambiguity. The classification is too general in some cases (such as the group with 22,480 entries), and too specific in others (such as the 3373 groups with four entries each, which connect only two words). The partitioning of 133,672 entries into singleton groups completely loses the semantic information that the thesaural classification affords for those entries.

#### 2.4.4 Thesaural Category Counting

An approach similar to Walker's idea of counting categories, as discussed above, was suggested for thesaural categories by John Brady (Brady 1990). Given a text with ambiguous words, the codes for each of the words in the text would be assembled and counted (using the 1042 groupings in Roget's Thesaurus). For each word, the sense would

be chosen that corresponded to the thesaurus code that occurred most frequently in the text. If this was insufficient to disambiguate a particular word, higher levels of classification in the Thesaurus would be used in the same manner (Sedelow 1990).

It is difficult to tell how successful this approach might be in actual practice.

## **2.5 Preference Semantics and Collative Semantics**

Yorick Wilks developed "... a set of formal procedures for representing the meaning structure of natural language, with a view to embodying that structure within a system that can be said to understand ..." (Wilks 1975, p. 329). Preference Semantics attempted to use insights of artificial intelligence to approach the problems of semantics, including the problem of lexical ambiguity. Since the goal was somewhat broader than just resolving lexical ambiguity, and Preference Semantics has been developed over a period of some twenty years, it includes many mechanisms that are well beyond the scope of this thesis. The most important idea, as applied to the resolution of lexical ambiguity, is

that predicators have preference for certain semantic classes of arguments, and modifiers have preference for certain semantic classes of head. The verb 'drink', for example, prefers an animate subject, and the adjective 'blue' prefers a concrete head. Although semantic restrictions on arguments had been previously discussed in the form of selectional restrictions, an important contribution of Preference Semantics is the idea that argument and head preferences can be violated.

Preferences themselves are represented as expectations for arguments and heads to belong to certain semantic classes. These classes are defined by a set of semantic primitives (from about 60 to 100 of them, depending on the version of the theory). Each sense of a word is defined by a semantic formula built up from these primitives, with one primitive as its central meaning. The formulas for verbs and prepositions include preferences for their arguments, and those for adjectives, adverbs, and prepositions include preferences for their head. During parsing, the central meaning of a potential argument is matched against the preference of the predicator. If the

potential argument is ambiguous, the formulas of each of its possible senses are matched against the preference of the predicator for that argument; if only one of the senses fulfills the preferences, that is the one that is chosen. Although greatly oversimplified, this is the basic mechanism in Preference Semantics for the resolution of lexical ambiguity. (Wilks 1975)

Recently, Brian Slator, working under Wilks, has implemented a program for extracting semantic data from the Longman Dictionary of Contemporary English, and automatically creating semantic formulae for each sense of each of the words listed there (Slator 1988). A considerable amount of information still needs to be added to the formulae before they can be used in a full Preference Semantics system. In any case, no data is yet available about the success of a full-scale Preference Semantics system in resolving lexical ambiguity.



Dan Fass, another of Wilks' students, has recently recast Preference Semantics in terms of frame-based feature structures. The same idea of preference is carried over, but the way in which preferences are satisfied has changed.

In Collative Semantics, word senses are the semantic primitives, and each sense of a word is defined by a frame structure. Each frame has arc information, which relates the frame to other frames, creating a hierarchical hyponymy structure, and node information, in which there are features and values, preferences for arguments, and assertions.

Each frame includes an arc which specifies the name of the next higher frame in the hierarchy. By following links, one can find all of the superordinates of a frame in the hierarchy. Preferences can be expressed by giving the name of a frame in the hierarchy. If this frame is among the superordinates of a potential argument, that argument satisfies the preference. If only some senses of an ambiguous word being used as an argument do not satisfy the argument preference, those senses are eliminated.

Preferences can also be expressed by listing a set of preferred features and values. In this case, the word sense whose frame best matches the preferred list is the one chosen. The best match is determined by a scoring algorithm; the score of a frame is the number of feature-value pairs that match minus the number of feature-value pairs that fail to match. Since the values of features in Collative Semantics are word senses (i.e. the names of other frames in the hierarchy), a feature-value pair in an argument (or head) matches if the frame of its value includes among its superordinates the value which is specified as the preferred value in the predicator (or modifier). In order to pick one sense above another by this method, it has to be clearly better than the other senses.

If no preferences of the predicator (or modifier) are satisfied, the preferences of the next higher frame in the hierarchy are checked. In this way, an analogy is sought, so that the sense of the argument (or head) is chosen which would best support a metaphorical reading.

Finally, senses with assertions (modifiers assert about their heads) are given scores depending

on whether the assertion adds inconsistent, redundant, or new information. Those which add new information are more highly valued than those which add redundant or inconsistent information.

Although the algorithms for lexical ambiguity resolution in Collative Semantics are potentially very useful, the complexity of coding word sense frames makes implementation of a full system a tremendous amount of work. The prototype system Fass developed has a vocabulary of about 400 frames, and each of the 50 test sentences worked correctly. It is difficult to assess the degree of accuracy that might be achievable in a full implementation. (Fass 1988)

Yet another group using the idea of preferences is the Distributed Language Processing group (DLT) in the Netherlands. The major disambiguation method is based on a hierarchy of word senses, and a set of wordsense:relator:wordsense triplets. These triplets represent commonly expected relationships. The relators are an abstracted form of prepositions or thematic roles. The syntax builds dependency trees, and each pair of words and the connection between them is mapped to a set of possible

triplets. The possible triplets are given scores, according to the distance in the hierarchy between the tree word senses and the triplet word senses. For each word, the scores of the triplets in which it is involved are summed, and the word sense with the highest score is chosen. A book describing this effort (Papegaij 1986) includes a section for test results, but at publication of the book, the software was not yet completed, so only the test procedure was described. As of the summer of 1988, the first results were available. The first results indicated that disambiguation would have been considerably more successful if the most frequent sense of each word had been chosen (Alan Melby -- personal communication).

## **2.6 Marker Passing**

Charniak developed a system of marker passing based on frame structures for knowledge representation. When a word in a sentence is encountered each of its sense frames is marked, even if it is not the sense eventually selected. Then, all frames referenced in these marked frames are

also marked, and so forth. Usually, some kind of strength is associated with each mark, and the strength of the mark diminishes each time until it falls below some pre-established bound, after which marking is discontinued. If some frame in the system gets marks from two different origins, the two sense frames of the two words which originated the marker passing are chosen as the correct senses for those two words. There can also be differing strengths associated with markers passed across certain arc-types in the frames, and so forth, to improve the performance of the system (Charniak 1983).

One major problem with this kind of system is the difficulty of coding the knowledge frames for word senses of any reasonable sized vocabulary. The difficulty has been such that a full-scale implementation has not been possible so far, and no data is available on how successful such a system might be in resolving lexical ambiguity in a full system.

Hirst attempted to combine the ideas of preference semantics and marker passing. His system uses information in frame structures to enforce

selectional restrictions and also uses marker passing among these frames. When a word in the sentence is encountered, it registers selectional restrictions on its arguments, and also begins a chain of marker passing. Senses of other words in the sentences can be eliminated by the selectional restrictions or chosen via marker passing. Apparently, selectional restrictions are not just preferences in his system; any senses of the argument that do not satisfy the requirements are eliminated (Hirst 1987, p. 108). It appears that markers passed from words earlier in a sentence could cause senses to be chosen that would fail to satisfy selectional restrictions registered later, or a sense eliminated by selectional restrictions could be the one that would have later been chosen by marker passing, so the selection of word sense could depend totally on the order of processing within the system. Naturally, selection by marker passing is completely dependent on the nature of the frame network, and slightly different philosophies of the nature of the features to be used could considerably alter the results of lexical ambiguity resolution. No full system has been implemented, so

no statistics are available as to the viability of this approach.

## **2.7 Connectionist Approaches**

The spreading activation idea naturally appeals to those who are interested in modelling the neural structure of the brain. Neural networks normally consist of a number of nodes and directional connections among the nodes. Associated with each node is an activation level, an output level, and sometimes a bias. Associated with each connection is a weight. At a specified point in time, each node applies an input function to the inputs it is receiving (the input on each connection is the output of the node it is connected to, times the weight of the connection) and determines its next activation level. It applies another function to determine its output from its activation (often this is a threshold function). Different networks are characterized by different numbers of nodes and different topologies of connections, as well as different activation and output functions. By receiving outside input, and cycling through

activation and output updates, these networks can calculate a variety of functions; if programmed carefully, the networks often settle into stable patterns that no longer change significantly with more cycles. By tuning weights in the network based on expected activations of certain nodes given inputs at other nodes, the networks can actually learn to compute certain functions. Automatic learning algorithms have been developed for some kinds of networks so that a sequence of input and expected output patterns can be presented to the learning routine, and it will automatically tune weights in the network so that the network will calculate the function implicit in the data. Some networks can generalize and develop internal representations for prototypical patterns; others can fill in missing portions of familiar patterns when only presented with a part of the pattern. Naturally these kinds of automatic learning and generalization capabilities are of great interest to the problem of ambiguity resolution, since it seems that humans learn words and semantic relationships by a process of examining large quantities of data and extracting generalizations.



Unfortunately, research into the abilities of neural networks for natural language processing is still in its infancy. Very few have been used in other than a toy demonstration mode, since serial computers make the highly parallel nature of large neural networks impractical to use. Nevertheless, some toy demonstration work has been done with neural networks in the area of resolution of lexical ambiguity.

Cottrell has discussed a localist approach, in which each concept is represented by a separate node (Cottrell 1988). In a three level network, the nodes of the first level correspond to words. Each word node is connected to nodes of the second level, which represent its word senses, which are connected to each other with inhibitory connections (with negative weights). These are in turn connected to nodes of a third level, which include nodes representing some syntactic concepts as well as internal semantic nodes that connect all of the word senses together. Although he does not specify the internal structure of this third level of nodes, the idea is that when a word is encountered, it excites the activation of nodes of its word senses. The

most frequent sense of the word is probably connected to the word with the most positive weight, so its activation increases fastest initially, unless nodes from the third level inhibit it based on previous context. Nodes of the third level either excite or inhibit the various word sense nodes, and since the word sense nodes are connected to each other with inhibitory connections, they compete for more input from the third level, until one wins the competition and is declared the appropriate sense of the word in the given context. Naturally all of this competition alters the third level so that it represents the previous context and the inclusion of the current word into context for the next word to be presented to the network. However, the exact connections needed in the network for all of this to function properly are not at all obvious. The approach awaits further description.

Kawamoto presents a more specific model, but prefers to use a distributed representation of concepts. This means that a concept is represented not by a single node in a neural network, but by a certain pattern of activation over the nodes of the network. In one of his networks, twelve ambiguous

words are represented by 24 patterns of activation (two for each word) over a network of 216 nodes. Each pattern consists of 48 nodes to represent the written form of the word (each of the twelve words is four letters long), 48 nodes to represent the phonetic form of the word (some words have different pronunciations in their two senses, such as /lid/ and /l\_d/ for 'lead'), 24 nodes to represent grammatical category, and 96 nodes to represent semantic features. The semantic representations assigned to each pattern are actually quite arbitrary, but were simply used to make a point. Portions of the 24 patterns were repeatedly presented to the network, and an auto-associative learning algorithm was used to modify weights in response to errors in generation of the remainder of each pattern. After sufficient training, it was possible to present partial patterns, and the network would fill in the rest of the appropriate pattern, as long as the part of the pattern presented was not ambiguous (it could not belong to more than one of the known patterns). Using a procedure called habituation, after a pattern had settled into a stable state, connection weights were

temporarily modified so that the network would decay out of the stable state. When an ambiguous new partial pattern was presented to the network in this habituated state, the network would settle on the pattern consistent with the input pattern which was most similar to the previous stable pattern. In this way, the network simulated resolution of a lexical ambiguity in terms of the context of the previous word (Kawamoto 1988).

Although the use of a one word preceding context to resolve lexical ambiguity in a neural network is impressive, it is difficult to know how to apply the research to the real world of ambiguity. The semantic features used in the toy simulation were assigned in a quite arbitrary way. Kaplan's study showed that at least two to four words of context would be necessary, and at least one or two of them should be after the ambiguous word. It is also difficult to know if the neural network scheme could ever scale up to the demands of a full vocabulary.

## 2.8 Summary

In the forty years since Warren Weaver's memo, a variety of approaches have been taken to solving the lexical ambiguity problem. None of the approaches taken so far has presented a reliable solution for an implemented real-time machine translation system.

Wilks said that

An ambiguity resolution system would be of some interest within computational linguistics even if it worked on a purely ad hoc basis, since word ambiguity is probably the problem holding up the achievement of reliable mechanical translation.

(Wilks 1968, p. 59)

Strangely enough, the systems just using cover words or the most common translations, or the most common translations within a certain kind of text, have actually been the most utilized and the most successful. Weaver's first suggestion, that ambiguity could be partially overcome by limiting

the domain of the input texts, is still the solution that is most widely used.

Weaver's next suggestion about using the 2N word context has been too difficult to implement, because of time and space considerations. The optimism that was generated by Kaplan's paper was based on an assumption that local context would be easy to deal with, but it has not proved to be the case. Methods that depended on division of science into some number of technical areas, and classification of portions of text into those areas, had a reasonable degree of success, but not enough that people were satisfied. Approaches based on shared domain codes showed promise, but lacked the underlying data base to even sufficiently test. Methods based on somewhat random single cooccurrences simply failed to be accurate enough.

Approaches using complex individual procedures for each word in the lexicon have been difficult to manage, and have not led to any general solution to the problem.

The thesaurus approach was a noble effort, but was probably based on a mistaken assumption, that synonymy was the most important semantic

relationship that an ambiguous word could have with other words in its context. Analysis of Roget's Thesaurus into chains of entries and type-10 partitions failed to produce a reliable method of semantic grouping. We still await a viable approach to using thesaurus information for resolving lexical ambiguity.

Preference Semantics was an important contribution to the field. Similarly, Collative Semantics has given a new vitality to the old idea of preferences. These approaches are difficult to implement because of the care with which dictionary entries must be prepared. The disappointing results of the DLT project seem to indicate that a system based only on selectional preferences will not be sufficient to handle a wide enough variety of cases of lexical ambiguity in real text.

The viability of the marker passing algorithm is difficult to assess, since it is so dependent on details of the frame lexicon. Hirst's system combines the use of selectional restrictions (but not preferences) with marker passing, but the uncertain ordering of elimination of senses by selectional restrictions, and the selection of

senses based on marker passing, leads to doubt about the validity of the algorithm. The care with which dictionary entries must be prepared in systems that use marker passing makes these approaches difficult to implement and therefore difficult to judge.

The neural network approaches previously developed have been extremely limited in scope. While they have offered some hope for the future, as Kaplan's paper did, they have not offered clear direction about how to expand them for use in a real full-scale system.

Part of the problem may have been a lack of clear perspective about what kinds of lexical ambiguity problems occur in actual text. The frequency approaches and neural network approaches have ignored selectional restrictions, while Preference Semantics and Collative Semantics have ignored almost everything else. The next chapter will discuss examples of various kinds of ambiguity in real raw text.



## Chapter 3

### AMBIGUITIES IN REAL TEXT

#### 3.1 Introduction

It seems to have been common practice to take a single approach to lexical ambiguity, and carry it out for a limited set of examples to show some feasibility for the approach. The last chapter gave a brief overview of some of the methods that have been tried. Because of the difficulty of creating large lexicons, however, statistics on the successfulness of various approaches in actual full-scale implementations are very rare. Although some methods look good for some well-chosen examples, it is very hard to know how useful the various proposals would actually be in a real system.

This chapter presents the results of an analysis of the types of lexical ambiguities that occur in actual text. The ambiguities found were manually classified according to which of several general approaches might successfully resolve them. This study was particularly concerned with lexical

ambiguities within the grammatical categories of adjective, verb, and noun. After a word was successfully categorized as belonging to one of these categories, several senses within that category often remained as possible meanings that would require different translations.

The study was based on four small texts. The first was a newspaper article about some whales which were trapped in Arctic ice (Provo Herald 1988); the second was made up of selections from an article on AI (Dreyfus 1985); the third was from a religious text (Smith 1978); and the fourth was from a LISP manual (Gold Hill Computers 1983). These texts are found in Appendix A. Together, the texts included 3848 words. Of these, 1537 were adjectives, verbs or nouns (about 40%). An English-to-Japanese translation dictionary was used to make a list of all the words which appeared in the texts. Each word was listed with its possible grammatical categories and possible translations within each category. Each word in the source texts was then annotated by its grammatical category in that context, along with the possible translations for that word within that category. It was found that

883 of the 1537 adjectives, verbs and nouns had intra-category ambiguities (about 57%).

Each word was then annotated with the subset of possible approaches which could hope to successfully resolve the ambiguity in its particular context. Some ambiguities could be resolved simply by syntactic considerations, or by assuming the phrase containing the ambiguity would be entered in the translation dictionary as an idiom. Others could be resolved if it was assumed that the number of translations in the dictionary was reduced so that a single cover term could be used to cover all of them. Others could be resolved by simply choosing the sense of the word that would be expected to be the most likely or frequently used sense in general text. Some words could be considered to have a technical meaning appropriate to the subject area of the text. Many ambiguities could be correctly resolved simply by using the same sense of the word that had been chosen for the previous instance of the same word in the text. Some ambiguities could be resolved by appealing to selectional restrictions. Others could be resolved by appealing to some notion of general relatedness to words in

the immediate context. Ambiguities in some words could be resolved by several of the approaches by themselves; others could be resolved only by a combination of approaches. Finally, some of the ambiguities could not be correctly resolved by any combination of these methods, and it was assumed that some higher form of logical reasoning would be required. The texts are presented in Appendix A with the annotations which classified each ambiguous word.

Sections 3.2 to 3.10 will present each of the approaches along with an example of its applicability to an ambiguity from the texts. Section 3.11 presents the results of the study. Section 3.12 comments on the possible meaning of the results.

### **3.2 Syntax**

Some intra-category ambiguities were resolvable using syntactic restrictions. Some of the restrictions considered were complement types and agreement for verbs, countability for nouns, and sentence position for adjectives.

For example, in the phrase

As Husserl saw ...

the verb could either be 'saw' as the past tense of 'see', or 'saw' as in 'sawing wood'. 'Saw' cannot have the 'sawing wood' sense in this case because it would not agree with the third person subject.

### **3.3 Idioms**

The correct sense of some words could be achieved by treating them as belonging to fixed idioms. For example, it would be more sensible to enter the whole phrase

In the first place ...

in the lexicon than to try to predict the translation of 'place' by some other method.

### 3.4 Frequency

Often, the easiest way to pick the correct sense of a word is simply to choose the sense that is most frequent in general text. For example, in the phrase

When the waitress came to the table ...

one of the ways to resolve the ambiguity is to simply take the most frequent meaning, that of furniture 'table' over 'table' of figures.

### 3.5 Technical Glossaries

Naturally, the relative frequency of word senses depends on the type of text being used as the standard. In a phrase in an article talking about knowledge representation, such as

... the script accounts for the possibilities in the restaurant game ...

the word 'script' is likely to have the technical meaning. The most likely meaning in a computer science text might be 'font', and in a text about the theatre, 'the written form of a play'. This method amounts to using a technical glossary for translating texts within a given domain.

### **3.6 Discourse Memory**

Often, the context surrounding the first use of a word is more specific than the context surrounding its later uses. Once the intended sense of the word is established by the context surrounding the first usage, the same sense is assumed in further instances of the word. For example, if the context

... thick Arctic ocean ice ...

establishes the meaning of 'ice' to be 'frozen water' rather than 'sherbet', the more difficult phrase later in the text,

... they quickly cleared the ice ...

can be resolved by simply using the same sense of 'ice' that was used earlier in the text.

Words were marked as resolvable by this method if the correct sense of the word was the same as the correct sense of the previous instance of the word in the text.

### **3.7 Cover words**

Although a source word may be translated by several different target words, one of the possible translations may be more general in meaning than the others, and actually include or "cover" those more specific meanings. For example, the verb 'decide' in English can be translated into Japanese by 'kettei suru', meaning 'decide definitely on'; or by 'kesshin suru', meaning 'decide in one`s heart'; or by 'kimeru', meaning 'decide upon'. Although the former two meanings are more specific, the latter meaning is general enough to cover them, and the context is usually sufficient for the Japanese reader to understand. In practice, this method would be utilized by entering only the cover meaning in the machine translation lexicon.



### 3.8 Preferences

A verb or preposition often prefers certain classes of arguments and an adjective often prefers certain classes of heads to modify. These preferences can sometimes be used to resolve ambiguities. For example, in the fragment

The ... whales became trapped in the ice two weeks ago while migrating south.

the two listed Japanese translations for 'migrate' were 'idoo suru', meaning to 'move or locomote', and 'ijuu suru', meaning to 'immigrate or emigrate'. The second one prefers a human subject, so the first translation is more likely in this context.

### 3.9 Spreading Activation

Often nearby words give a clue to the proper sense of a word. For example, in the fragment

... California gray whales, whose species is endangered, became trapped in the ice ...

'species' can be translated by 'shu', meaning 'type of living thing'; or by 'shurui', meaning 'type or kind'. In this case, the proximity of the word 'whales' makes it clear that the first is the best translation. The word 'ice' can be translated by 'koori', meaning 'frozen water'; or by 'shaabetto', meaning sherbet. Whales are clearly related to water, and therefore to 'frozen water', but there is no such immediate connection between whales and sherbet.

### 3.10 Logical Reasoning

Some kinds of ambiguities could not be resolved by any of the above methods or any combinations of them.

In the fragment

But Minsky seems oblivious to the hand-waving optimism of his proposal that programmers rush in where philosophers such as Heidegger fear to tread . . .

'oblivious' can be translated as 'kizukanai', meaning 'unaware'; or as 'wasureppoi', meaning 'apt to forget'. In this case, it seems more likely that Minsky was unaware of his "hand-waving optimism" than that he had once been aware of it, but had forgotten about it. This kind of reasoning is not something that could easily be captured by any of the previous methods for resolving ambiguities.

### **3.11 Results of the Study**

The following table reports the statistics collected. Lines 1 through 5 give general statistics on the number of words, number of words in the categories surveyed, and the number of polysemous words (with intra-category ambiguities); also the number resolvable by syntactic

considerations alone, or by assuming the words occurred in idioms that had been entered in the dictionary.

The second part is based on the number of ambiguities left after the syntactic and idiom methods had been used. The statistics in lines 7 through 12 are given as percentages of the number of remaining ambiguities (shown in line 6) that can be resolved by the given method alone. In line 13, the percentages represent the number of ambiguities left unresolved by all of methods used in lines 7 through 12 (and combinations of those methods).

	Text1	Text2	Text3	Text4
Total				
1. Total words	421	1776	1212	439
3848				
2. Adjs, verbs, nouns	194	731	411	201
1537				
3. Polysemous words	84	424	248	127
883				
4. Resolvable by syntax	20	93	61	20
194				

5. Part of an idiom	12	36	11	10
69				
6. Unresolved by above	52	297	176	97
622				
7. Frequency	46%	39%	26%	39%
36%				
8. Technical glossaries	0%	15%	3%	23%
11%				
9. Discourse memory	21%	40%	37%	79%
44%				
10. Cover words	27%	18%	27%	19%
21%				
11. Preferencess	27%	11%	15%	3%
12%				
12. Spreading activation	25%	9%	13%	3%
10%				
13. Logical reasoning	2%	2%	5%	1%
3%				

The percentages do not add up to 100% because some ambiguities are resolvable by more than one of the methods. Of the ambiguities not resolved by syntax and idioms, only 2% required some combination of the methods in lines 7 through 12. Each of the

methods in lines 7 through 12 resolved between 3% and 9% that could not be resolved by any of the others.

### **3.12 Comments on the Results of the Study**

The purpose of the study was to get a general idea of the potential usefulness of the general approaches that have been taken to resolving lexical ambiguity during machine translation. Of course, since the purpose was to get such a general idea before actually going to the trouble of trying to make a full-scale implementation of any of the methods, the data had to be analyzed manually, and the results were unavoidably somewhat subjective. The differences in the percentages in different columns show that the statistics vary for different kinds of text. The range of the percentages on each line may give some general idea of the range of usefulness of the various methods.

The results could be considered surprising in a number of respects. It might have been expected that taking the most frequently correct sense of a word would have succeeded at least 50% of the time,

but in the texts examined, it was not uncommon to find words with three or more possible senses. If a word had three senses, the first occurred 40% of the time in general text, and the others occurred 30% each, one would expect to achieve only about 40% accuracy by choosing only the most frequent sense.

It was most common for a word to be used in only one sense within a single text. Despite this fact, the discourse memory method was usable in less than 50% of the cases because many words occurred only once in the text, and the method was not usable for the first occurrence of even words that occurred more than once. The usefulness of this method is somewhat exaggerated by the statistics, because it was assumed that the previous occurrence of the word in the text had been correctly classified, and at least the first occurrence had to have been classified by some other method.

Although a word was commonly used in only one sense within a text, it was not at all clear for many of the words that they would be used consistently in that sense alone in every text of the same subject area. Words having a special meaning in the particular subject area were marked

as resolvable by the technical dictionary method. The statistics might have been higher if actual statistics had been available about the frequencies of usage in the various types of text. The first text was considered a general text, since it had come from a newspaper. Perhaps more of the words could have been considered technical if it had been labelled as a text about oceanography or zoology. Still, the number of words that could be correctly disambiguated using the technical domain approach was surprisingly small, since that is one of the major methods being used in actual systems.

Another surprise was the low score for preferences. The need for using selectional restrictions is somewhat obscured by these statistics, since disambiguation of prepositions depends heavily on them. Still, the statistics are surprising, because they imply that the best efforts of Preference Semantics might hope to resolve only about 30% of the major category class words in the best case, and only about 12% in general. Perhaps the statistics could be improved by using some extremely ad hoc features, but the prospects of using Preference Semantics alone seem rather bleak.



Since each method resolved 3% to 9% of the ambiguities that could not be resolved by other methods, and 3% would require some form of logical reasoning, it seems clear that to reach 95% accuracy in resolving lexical ambiguity (without using logical reasoning), one would have to use some combination of all of the methods. No single method will do.

## Chapter 4

### SPREADING ACTIVATION AND TRIGGER WORDS

#### 4.1 Introduction

Other than cases of ambiguity that require logical reasoning, the fuzziest division between the methods discussed in the last chapter seems to be between the frequency method and the spreading activation method. The cases in which the other methods apply are fairly clear cut. Cases where syntactic restrictions apply are easy to identify. Idioms are also easy to identify. Texts can be marked for their technical area before translation, so that word senses particularly applicable to the given area can be given precedence. When words are first encountered in a text, the word and its chosen sense can easily be stored, so that when the word is found again, the same sense can be used (as long as it has the same syntactic characteristics). The dictionary can be coded in the first place with the concept of cover words in mind, so that unnecessary ambiguities do not even become possible choices. Preferences

require considerably more work, but methods for applying them have been studied for over 20 years by those working in Preference Semantics (a simplified version of Collative Semantics may offer the best working algorithm); again, the cases in which modifier and argument preferences can resolve ambiguity are easy to delineate, once the lexicon has been correctly coded.

The frequency and spreading activation methods are not so easy to separate, however. The idea is that if a word appears in the context of certain other words, they may trigger a sense other than the most frequently used sense. The trick is to define in some manner the trigger words allowed for each less frequent sense and how they may interact; if none of these are found in the context, the most frequent sense should be used as a default.

Capturing and representing trigger information is no easy matter, however. Chapter 2 discussed the idea of Weaver's contextual windows; to use a window of size  $n$  on each side of a word, a matrix of  $M(2n+1)$ , where  $M$  is the size of the vocabulary, would be required. A simpler method might be an  $M^2$  sized matrix, where rows are ambiguous words,

columns are potential trigger words, and the value in a cell of the matrix would be the correct sense of the ambiguous word (row) in the presence of the trigger word (column). Alternatively, since many of the cells on the row of an ambiguous word would contain the sense number of its most frequent sense, it would be enough to specify the most frequent sense and list each of the other senses with the trigger words that trigger them (the program in Appendix B is actually an implementation of this approach). However, how does one obtain the list of trigger words and the senses they trigger? Even if this information can be obtained, what should be done if trigger words that trigger different senses are found in the context? Does distance from the word matter? Should trigger words have different strengths?

The thesaurus methods of Chapter 2 were an attempt to quantify trigger words. A word was a trigger if it appeared under the same heading as the ambiguous word (in one of its senses) in the thesaurus. The notion of semantic distance could also be used to give different words a different trigger strength. However, the complexity of

building the thesaurus makes this method very difficult to use, and no statistics are available about how successful even one implementation might be. The fact that the thesaurus is based solely on synonymy also makes it doubtful that this method could be very useful.

The marker passing methods also attempt to address the trigger word problem. By spreading activation among nodes in a knowledge base, trigger words are allowed to choose the triggered sense of an ambiguous word. Trigger words can also be given different strengths. There may even be a way to handle cases where words that trigger different senses occur in the same context. But since knowledge bases are so difficult to build, no statistics are available on the possible success of these methods in a full implementation, and there is no guarantee that success in one such system would imply success in another one with a different knowledge base.

The connectionist approaches are appealing because they can be trained by real data, and they have the ability to generalize. It seems plausible that the process of deciding on a word sense among

several alternatives is more like a weighing process than a logical discrete process. However, previous work has been limited to vague discussion and toy examples that give no clear direction for implementation in a full system.

An optimal approach would be one that could automatically determine most frequent sense from actual textual data, and at the same time gather information about potential triggering of less frequent senses. If this information could then be used in some manner to train a connectionist network, complex and arbitrary coding of thesauri or knowledge bases could be avoided, and triggering and frequency information could be combined in a probabilistic algorithm for resolving ambiguity. The approach described in this chapter is a beginning attempt at just such a system.

## **4.2 Neural Cooccurrence**

One of the goals of using a connectionist approach would be to find an architecture for a neural network that could generalize classes of cooccurring words. An experiment will be described

in this section, in which a small list of ambiguous words were correctly categorized into cooccurrence classes by a neural network.

The input words and general categories of intended senses were as follows:

bank	money	water
bail	container	money
bat	animal	sports
bridge	cards	water
calf	animal	bodypart
chest	bodypart	container
fence	container	sports
file	office	tool
palm	bodypart	plant
pen	container	office
pitcher	container	sports
poker	cards	tool
pool	sports	water
spade	cards	tool
squash	plant	sports

The network was a feed-forward neural network. Such a network is a connected directed acyclic

graph; it is composed of nodes and one-way connections between pairs of those nodes. Each node in the network has an associated real number called its activation level, and another real number called its bias. Each connection is also associated with a real number called its weight. The activation of each node changes once each cycle, and is calculated based on the following equation:

$$a_i = - ( b_i + \sum_j a_j * w_{ji} )$$

where  $a_i$  is the activation of node  $i$

$a_j$  the activation of node  $j$

$b_i$  the bias of node  $i$

$w_{ji}$  the weight of the

connection

from node  $j$  to node  $i$

and

$$\sigma ( x ) = \frac{1}{1 + e^{-x}}$$



This particular network was composed of three sets of nodes, namely input nodes, hidden nodes, and output nodes. Each input node had a directed connection to each hidden node, and each hidden node had a directed connection to each output node. The network had 16 words as input units, 8 hidden units, and 32 output units. Each input unit corresponded to one of the 16 words. There was one output unit for each sense of each word. The network was set up using the bp (back propagation) program in the Explorations in Parallel Distributed Processing (PDP) software of McClelland and Rumelhart. The biases and weights in the network were trained according to the back propagation formulas. In these formulas, each non-input unit has an error term calculated. The error terms for output units are simply

$$E_i = t_i - a_i$$

namely, the target activation minus the actual activation.

The error term for each hidden unit depends on activations and error terms of all of the output units the hidden unit is connected to, as follows:

$$E_i = \sum_j w_{ij} * D_j$$

$$\text{where } D_j = E_j * a_j * (1-a_j)$$

Weights and biases are changed based on these values and constants which control the gradual descension of the network into a stable state which has learned the patterns presented. The change in each bias and weight is stored for use the next time they are calculated; these are

$$\_w_{ij} = \_i j * D_i * a_j + \mu * \_w_{ij}$$

$$\_b_i = \beta_i * D_i + \mu * \_b_i$$

The constants are  $\mu$  (for momentum) and  $\_i j$  for weight learning rate and  $\beta_i$  for bias learning rate. Each weight is then modified by adding the  $\_w_{ij}$  term, and each bias is modified by adding the  $\_b_i$  term. Each weight and bias in the network can be

modified once each time a training pattern is presented, or once after the whole set of training patterns has been presented. For this test, weights were modified after each training pattern was presented, and the parameters were  $\mu=.9$ ,  $\alpha=.5$  for connections from hidden units to output units,  $\alpha=.08$  for connections from input to hidden units,  $\beta=.5$  for output units, and  $\beta=.08$  for hidden units (see McLelland 1988).

The following pairs sharing a sense in the same category were presented as input to the network:

pool bank	(water)	pool pitcher	(sports)
pool bridge	(water)	pool fence	(sports)
bank bridge	(water)	pool squash	(sports)
bank bail	(money)	pool bat	(sports)
bank stock	(money)	pitcher squash	(sports)
bail stock	(money)	pitcher bat	(sports)
stock calf	(animal)	fence squash	(sports)
stock bat	(animal)	fence bat	(sports)
calf bat	(animal)	squash bat	(sports)
chest calf	(bodypart)	bail chest	(container)
chest palm	(bodypart)	bail pitcher	(container)
palm calf	(bodypart)	bail fence	(container)

squash palm	(plant)	bail pen	(container)
file pen	(office)	chest pitcher	(container)
file spade	(tool)	chest fence	(container)
file poker	(tool)	chest pen	(container)
bridge spade	(cards)	pitcher fence	(container)
bridge poker	(cards)	pitcher pen	(container)

For each training pattern, the activations of the input units corresponding to the two input words were set to 1, and the target activations of the correct output senses were set to 1, but the target activations of the incorrect senses were set to 0. The error terms of all output units which were not senses of the input words were set to 0. The total sum of squares is a measure of how well the network has learned the input patterns; it is simply the sum of the squares of the error terms of the output units. The network in this test was trained until the total sum of squares fell below 0.4. All of these parameters, as well as the architecture of the network, were specified in PDP network and pattern files.

After the network was trained, the values of the hidden units were examined for each of the

training patterns. The activations for the hidden units of each pattern (rounded to 0 if less than .5, and to 1 otherwise) were as follows:

11010000	pool	bank	(water)
11010000	pool	bridge	(water)
11010000	bank	bridge	(water)
10000010	bank	bail	(money)
10110110	bank	stock	(money)
10100010	bail	stock	(money)
00110000	stock	calf	(animal)
00110110	stock	bat	(animal)
00111100	calf	bat	(animal)
10101000	chest	calf	(bodypart)
10101000	chest	palm	(bodypart)
00101000	palm	calf	(bodypart)

01000000	squash palm	(plant)
01011101	pool pitcher	(sports)
01011101	pool fence	(sports)
01010100	pool squash	(sports)
01010100	pool bat	(sports)
01010110	pitcher squash	(sports)
00010101	pitcher bat	(sports)
01010110	fence squash	(sports)
00010101	fence bat	(sports)
01010100	squash bat	(sports)
10101011	bail chest	(container)
10101111	bail pitcher	(container)
10001111	bail fence	(container)
10101111	bail pen	(container)
10101111	chest pitcher	(container)
10001111	chest fence	(container)
10101111	chest pen	(container)
10101111	pitcher fence	(container)
10001111	pitcher pen	(container)
00000111	file pen	(office)
00010011	file spade	(tool)

00010011 file poker (tool)

10110000 bridge spade (cards)

10110000 bridge poker (cards)

From this data, it appears that the network generalized to representing not specifically the pair of inputs, but rather the common class that both of their senses belong to when they appear together. The classes could be summarized as follows:

office	00000111
tool	00010011
animal	0011???0
sports	0?01?1??
plant	01000000
bodypart	?0101000
container	10?01?11
money	10??0?10
cards	10110000
water	11010000

In other words, the network set up what could be considered a binary feature representation of the underlying cooccurrence classes.

### 4.3 Training with Raw Text

Given a neural network structure that can potentially generalize cooccurrence classes, the next step is to train such a network with data from real text.

For this experiment, the Longman Dictionary of Contemporary English was searched for occurrences of the word 'bank' within the texts of definitions (parenthetical material was not considered). The following are words that cooccur three or more times with 'bank' in its financial meaning within those definitions:

account book business can central certain cheque  
close door give interest money order paper  
particular pay people person print public put record  
room state sum supply system take various



The following words cooccur at least three times with the geographical meaning of 'bank':

built earth ground high lake overflow river rock  
sand sea stone stream underwater water wide

A program and parameters similar to the one above was used (see Appendices C and D), with eight hidden units; one input unit for each context word and one for 'bank'; and one output unit for each word except 'bank', which had one for financial 'bank' and one for geographical 'bank'. Each training pattern consisted of a pair of input words, namely 'bank' and one of the above context words; in the target patterns, the context word and the correct sense of 'bank' had target activations of 1, the incorrect sense of 'bank' had target activation of 0, and the error terms of other output units were set to 0.

Each definition containing the word 'bank' was then presented to the network as input. For each definition, an input unit was given an activation of 1 if the unit corresponded to a word which occurred in the definition (possibly more than once);

otherwise it was given an activation of 0. The activation levels of the output units for the senses of 'bank' were then compared, and the one with the highest activation was chosen to be the proper meaning in that context.

Of 97 definitions including the word 'bank', 32 referred to a geographical bank. Of these 97 definitions, the network got the wrong meaning of 'bank' in only one case, namely the definition

a deep bank or mass of snow formed by the wind

which includes none of the context words above.

The experiment was tried again, using only the first 76 definitions as input for training pairs. Only words which cooccurred three or more times with 'bank' in these 76 definitions were used as companions for 'bank' in training pairs. The words 'door', 'particular', and 'room' were no longer paired with financial bank, and the words 'ground', 'high', and 'underwater' were no longer paired with geographical bank. The remaining 21 definitions were then presented as input, and the network failed

in 2 cases, one being the one above, and the other being

a bridge consisting of a high tower on each bank connected by lengths of steel rail from which a flat carriage level with the ground hangs

The words 'high' and 'ground' triggered correct resolution in the first case, but were not known as trigger words in the second. The network therefore succeeded in over 90% of the cases of previously unseen definitions.

#### **4.4 Summary**

One of the most difficult problems in resolving ambiguity is deciding when sufficient words are present in the given context to trigger a sense that is not the most frequent sense. One would prefer a method that could automatically extract frequency and trigger information from real text, and use it in an automatic algorithm that would not have to depend on arbitrary coding of thesauri or knowledge bases.

The experiment of section 4.2 showed that a neural network can correctly generalize cooccurrence classes if trained by presenting pairs of ambiguous words and the correct senses they induce in each other. The network set up what could be regarded as a feature representation of the cooccurrence classes, as shown by the activations of the hidden units.

The experiment of section 4.3 showed that training pairs can be extracted from real text. Using a network similar to the one in section 4.2, and the automatically extracted training pairs, a network was built that could choose the correct sense of 'bank' in over 90% of the previously unknown test cases.

The next chapter will show how the same method can be applied to a large text corpus with many ambiguous words.

## Chapter 5

**SEMANTIC COOCCURRENCE NETWORKS BASED ON  
PARALLEL TEXT CORPORA****5.1 Introduction**

Unfortunately, many studies of lexical ambiguity have either been vague treatises that are difficult to formulate precisely, or detailed descriptions of toy systems that may or may not generalize for implementation in full systems. Some approaches are based on hand-tailored thesauri or knowledge bases that take so long to create that full implementations have yet to be completed, so no information is available about how successful the approaches may actually be in the world of machine translation.

This chapter presents statistics from the application of the method described in chapter 4 to an English text of over a quarter million words, which contained a vocabulary of over 9000 base forms. A parallel French text was used to determine different senses, and generate word pairs for training a large neural network.

Section 5.2 discusses the problem of the number of word senses for a given word. Section 5.3 discusses the method for extracting training pairs from parallel text corpora. Section 5.4 discusses training of the neural network. Section 5.5 is an evaluation of the results. Section 5.6 discusses problems that remain. Section 5.7 summarizes the conclusions of the study.

## **5.2 Word Senses**

One of the problems in dealing with lexical ambiguity is deciding exactly how many different senses a word really has. Yorick Wilks has considered

... the suggestion that there never was lexical ambiguity until dictionaries were written in roughly the form we now have them, and that lexical ambiguity is no more or less than a product of scholarship ...

since

... different dictionaries may give 1, 2, 7, 34, or 87 senses for a single word and they cannot all be right. Worse yet, different segmentations of usage into discrete senses may not even be inclusive. Sometimes different dictionaries will segment usage into senses for a given word in non-comparable ways: perhaps play<sup>3</sup> (the third of three) in dictionary A could not be associated with any one of the eight senses of 'play' in dictionary B.

(Wilks 1987, p.

3)

He later rejects the idea that lexical ambiguity exists only in the minds of lexicographers; his argument that words really do have different sense meanings essentially boils down to the fact that the same word in different contexts cannot always be translated by the same word in another language.

The question remains, how many senses does any particular word have? But, perhaps that is the wrong question. It doesn't really matter how senses

are divided up until the purpose for the division is known. In terms of machine translation, the real question is, "How many words in the target language are necessary to sufficiently cover the possible meanings of a given source word?"

When the algorithm for resolution of lexical ambiguity is based on parallel text corpora, this question becomes relatively easy to answer. The number of target words needed to translate the given word into the target language is no more than the actual number of target words that were used to translate that source word in the parallel texts. If some of the target words were synonymous, or some could cover the meanings of others, the actual number of target words needed could be less than the actual number used. In any case, the text corpora give an upper bound to the number of ways the word can be divided into senses.

Furthermore, if the target translations are used to partition the meanings of the source word into senses, the frequency of usage of each of the senses can be immediately calculated, and contexts which may trigger each of the senses are also available.



### 5.3 Extracting Training Pairs from Parallel Texts

In order to test the reliability of the method presented in chapter 4, an experiment was conducted based on parallel texts of approximately a quarter million words of English and French text, which contained a variety of government and non-government documents (this text was obtained from Alan Melby, from some texts used to test the DLT algorithm discussed in chapter 2). The text was divided into over 6000 parallel sections, each of which contained from one to seven sentences.

Function words were removed from the texts, and remaining words were reduced to base form. A bilingual dictionary was created that showed nearly all of the French translations for the 9103 English base words that occurred in the corpus.

The parallel texts were divided into a sample corpus, namely the first 80% of the parallel texts, and a test corpus. The sample corpus was searched for pairs of English words that occurred together five or more times within five words of each other; if the pair of English words mapped to the same pair of French translations in 85% of their

cooccurrences, the cooccurrence was deemed to be significant. 54 words had such cooccurrence data for more than one sense. The program in Appendix B was used to determine that 67.3% of the instances of ambiguity of these words in the test corpus could be resolved correctly simply by picking the sense that had been most frequent in the sample corpus. The trigger words collected from the sample corpus were then used to resolve ambiguity in the test corpus. Trigger words within five words of the ambiguous word were counted, and the translation with the most trigger words was chosen. If there was a tie, the translation among those that tied, which had been used most frequently in the sample text, was the one chosen. If there were no trigger words within five words, the translation which had been most frequent in the sample text was chosen. Using this method, 76.4% of the ambiguities were resolved correctly.

#### 5.4 Training the Neural Network

Normally, the practice is to keep neural networks small. The interaction of cooccurrence data, however, requires a critical mass of data in order for broad contextual information to affect the proper choice of sense. In the network, there is one input node for each source word, and one output node for each target word. Each input node is connected to every hidden node, and each output node is connected to every hidden node. This means each source word has its own set of weights, which are connections to the hidden units, and each target word has its own weights, which are connections from the hidden units. Source weights are only modified during training when the given source word is one of the two in the training pair; target weights are only modified when the given target word is one of the possible translations of the two source words, either a correct or an incorrect translation. Unless the training patterns overlap in either source or target words, the network will behave like many smaller networks, that just happen to use the same hidden units. Much like the same park being

used one day for a church picnic and the next for a hard rock concert, the one will not have much effect on the other unless some of the participants meet each other. When a source word maps to a particular target word, this combination could be considered to be a sense of the source word. The set of senses of source words available in the training data constitutes a kind of semantic cooccurrence concept world, where each sense is treated as a concept. Senses in this concept world are connected if they cooccur. In other words, each training pair can be considered to be two concepts in the concept world, which are connected by a cooccurrence line. Concepts can only have a contextual effect on other concepts if they are transitively connected by cooccurrences.

Only 54 of the words in the corpus had cooccurrence data for multiple senses, using 5 cooccurrences within 5 words. In order to establish greater connectivity among the senses in the corpus, pairs were extracted that occurred together 3 or more times in the parallel texts, and the context was allowed to be the whole segment in which the word was found; again, each pair of English words

was required to map to the same pair of French translations in 85% of their cooccurrences. Using these parameters, there were 2855 English words with cooccurrence data, involving 2462 French words, with a total of 3835 unique source-target mappings in the cooccurrence data. Considering each of these mappings as a concept in conceptual space, and connecting concepts that cooccurred, it was determined that the conceptual space would be partitioned into nine sets of concepts. Concepts in each set were related by the fact that some sequence of concepts could be found such that every pair of adjacent concepts in the sequence cooccurred in the training data. It turned out that eight of the nine sets of concepts contained only two concepts each, and the ninth set contained the remaining 3817. In other words, the training data allowed 3817 of the concepts to have a contextual influence on all of the others.

Now, armed with underlyingly connected training data, a neural network was created with one input unit for each English word and one output unit for each French word in the training data. The network used 54 hidden units, approximately the square root

of the number of input units. The learning parameter  $\alpha$  for input-to-hidden weights and the bias parameter  $\beta$  for hidden units was set to .5, and the learning parameter  $\alpha$  for hidden-to-output weights and the bias parameter  $\beta$  for output units was set to .08. The momentum parameter  $\mu$  was set to .9, and the epoch learning mode and permuted training were used.

Since the network was so large, the PDP software was unable to handle the task. Therefore, the network training program in Appendix C was used. It uses the same algorithms as the PDP software, except that weights of connections emanating from a single input unit are normalized so that they form a vector of length 1.0. It also interfaces better with large amounts of training data. In this case, there were 20,738 training patterns. Each cooccurring pair of English words was presented as the input pattern, with the French words they mapped to receiving positive feedback, and other French words they could have mapped to receiving negative feedback. Each pattern was presented to the network 176 times.

### 5.5 Evaluation of Network Performance

The program in Appendix D was written to test the performance of the network. Input to the program included the ambiguous word, a list of context words, the list of possible translations of the ambiguous word in the original dictionary, and the correct translation in the given context.

The network was tested on the five English words 'articles', 'committee', 'company', 'major', and 'office.' 'Articles' was considered a base form, since it occurs as a key word in the Longman Dictionary of Contemporary English. A modified version of the program in Appendix B was used to extract contexts and correct translations from the parallel texts for each of these five words.

Each of the five words will be listed below with its translations and common cooccurrences, the total number of test sections for that word, the percentage that could be achieved simply by choosing the most frequent translation, and the percentage achieved by the network.

articles articles aid, appropriate,  
attendance, base,  
committee, confidential,  
council, debate, only,  
other, parliament,  
parliamentary, sign,  
substance, treaty,  
unofficial

statuts  
accordance, activity,  
board, company, directive,  
dividend, entrust, form,  
limit, ordinary,  
possibility, proposal,  
protection, provisions,  
register, related, second,  
shares, status, statutory

32 instances, 56% translated 'statuts',  
network achieved 87%

committee commission apply, articles, brief,  
cassette, confidential,  
confidentiality,  
delegation, enable,  
experience, head, last,



open, option, parliament,  
 political, sign,  
 submission, substance,  
 superior, unofficial  
 committee trust  
 comité alternance, centre,  
 complementary, concrete,  
 consultative, decision-  
 making, desirable,  
 division, employment,  
 especially, foreigner,  
 generally, impact,  
 importance, instrument,  
 integration, language,  
 least, needs, position,  
 principle, problem,  
 qualifications,  
 recommendation, session,  
 situation, social,  
 standing, technology, town,  
 unemployment

159 instances, 54% translated 'comité',  
 network achieved 60%

company	compagnie	---
	entreprise	acceptable, common, compensate, complexity, conditions, crisis, difficulty, discourage, education, expose, handicap, key, knowledge, lighting, manpower, objective, optical, organize, permit, population, possibly, potential, principle, productive, requirement, responsibility, signal, skill, sound, strengthen, technological, transmission, treat, type, venture
	société	articles, corresponding, debtor, directive, entrust, explicitly, governing, indirectly, issue, legally, list, meeting, network, normal, bureau, orders,

payable, register,  
 regulation, statutory,  
 structural, third, three

166 instances, 62% translated 'entreprise',  
 network achieved 66%

major	grand	budgetary, machine, quality
	gros	appliance, asset, domestic
	important	lighting, steel
	majeur	alternate, court
	principal	---

114 instances, 43% translated 'grand',  
 network achieved 47%

office	bureau	abroad, communication, hour, manual, migrant, transport
	office	enterprise, federation, finance

37 instances, 54% translated 'bureau',  
 network achieved 62%

On these words, where sufficient cooccurrence data was available, the network was successful. In

general, the network achieved 64.6% for the 909 words with cooccurrence data, whereas picking the most frequent sense would have achieved 69.2%.

### **5.6 Problems with Neural Cooccurrence Networks Based on Text Corpora**

Naturally, the quality of the text corpora greatly affects the success of the network. Human translators are not always consistent in their selection of translations for the same word. Approximate synonyms in the target language may be used almost interchangeably for what could be considered a single sense in the source language. In these cases, although either might be acceptable translations in a given context, the automatic method of measuring accuracy used here insisted that the network use the same translation that was found in the text. This is more of a problem with the measuring algorithm used than the network itself, since either translation might be understandable to a human reader.

Subtle gradations of meaning in source words are difficult even for human translators, and also

present a problem for the network. Because the network is based on real number ranges of activations and weights, however, this problem is dealt with more realistically than in systems that use only discrete mathematics. Most of the ambiguous words encountered in this text involved the subtle gradations of meaning, so their resolution was not as clear cut as the 'bank' example of chapter 4.

The parallel corpora used for this example included some odd cooccurrences. For example, some of the texts reported vote tallies of countries in the United Nations. Whenever countries voted similarly 3 or more times, the country names appeared together with the same translations enough times that the system considered them to be training pairs. There were other cases in which the same wording was used in several different paragraphs, as if it were being quoted from some standard. In these cases, words whose cooccurrence was really more accidental than meaningful became training pairs. Parallel texts of any length will probably contain a certain amount of idiosyncratic cooccurrence. If enough text is used, these kinds

of idiosyncracies should tend to cancel each other out.

Unfortunately, some words occurred so infrequently in the text that very little cooccurrence data was generated for training purposes. Infrequent words tend to be less ambiguous, and therefore somewhat more useful to humans for resolving ambiguity, but this is not sufficiently captured by the current approach. Although humans probably do not use words they have heard only 3 times for much of anything including lexical disambiguation, it is not currently feasible to present the machine with the massive amounts of input data that humans receive. The training data using a minimum of 3 cooccurrences was not as accurate as that using a minimum of 5 cooccurrences. Using the program in Appendix B and default translations from the sample text, the correct translation could be achieved in 69.2% of the cases of ambiguity for the 909 words in the corpus that had cooccurrence data. Using the cooccurrence data as input to the same program, and picking translations based on tallying the number of trigger words in the context, 69.1% of the cases could be

resolved correctly. Clearly, the training pairs were less accurate using a minimum of only 3 cooccurrences. It is anticipated that a larger training corpus would allow more accuracy in finding training pairs, and the performance of the network would be considerably better.

Although the matched training that humans do to learn word meanings is based on readily available source words and target situations, parallel translated texts may not be so easy to find. Naturally, the texts should reflect the kind of material to be translated, so that the contexts and frequencies are not too skewed. It may not be easy to find or create parallel texts similar enough to the material to be translated and large enough to sufficiently train the network. Fortunately, the availability of machine-readable texts in general, and parallel texts as well, should greatly increase in the future.

The approach as given here presents the entire context to the network at one time by turning on the input unit associated with each word that occurs in the context. At first thought, this seems like an odd way to find words that trigger a particular

meaning, since it is common for only one or two words in a context to actually trigger the correct meaning. An alternate method would be to present the ambiguous word and a context word as a pair to the network, once for each word in the context, and tally up the scores of each of the senses. It turns out, however, that the more the network is trained, the more each context word has an opinion about what the correct sense of the ambiguous word ought to be, even if it is not really a trigger word for humans. Even establishing a threshold of activation for target senses in order to enter the tally does not seem to work, perhaps because different training pairs are learned by the network at different rates. Apparently, presenting the whole context as input to the network makes the words with uninformed opinions cancel each other out, leaving the trigger words to decide the correct sense. If insufficient context is presented, cancelling may not be complete enough.



## 5.7 Conclusion

This chapter has shown that the neural network approach for resolving lexical ambiguity that was introduced in the last chapter scales up for use in a real system, based on data automatically extracted from parallel texts. Although the results are not as clear cut as the 'bank' example in chapter 4, the results of this chapter show that the network performs reasonably well even with words which have a subtle gradation in sense meanings.

## Chapter 6

**RESOLUTION OF LEXICAL AMBIGUITY IN MACHINE  
TRANSLATION****6.1 Introduction**

Chapter 2 discussed previous approaches to the resolution of lexical ambiguity in Machine Translation. Warren Weaver's first suggestion was to limit the text to technical areas in which the technical meanings appropriate to that area would be given preference. His second suggestion was to use windows of context, but direct implementation of his idea was found to be impractical. Methods based on word experts, routines particular to each ambiguous word, were found to be ad hoc and difficult to maintain. Methods based on semantic groups found in a thesaurus were found to be inadequate if based on a published thesaurus, and custom-built thesauri suffered from the mistaken assumption that the only valid relationship for grouping was synonymy. Preference Semantics was found to be useful, but inadequate in scope. Marker passing schemes are potentially useful, but extremely complex to implement

in a full system, since they depend on a full knowledge base, and no statistics are available about their actual usefulness in a full system. Previous connectionist schemes were either too vague or else just toy systems that would be difficult to scale up to a full implementation.

Chapter 3 discussed a manual analysis of four texts to determine the kinds of ambiguities that might actually occur, and methods that might be used to disambiguate them. Some ambiguities were resolvable purely by syntactic restrictions. Some could be resolved by entering idioms in the translation dictionary. Some could have been resolved by more careful coding of the dictionary, so that specific target translations whose meanings could be covered by more general target words would not be included in the dictionary at all. Some were resolvable by marking certain translations as especially appropriate for certain technical areas, and then marking texts to be translated with the technical area of their subject matter. Some could be resolved simply by choosing the same translation that had been used for the word in its previous occurrence in the same text. Some required some

kind of checking of argument or modifier preferences. Others could only be resolved with the assumption that certain trigger words in the context would signal the appropriate meaning. Some could be resolved correctly by simply choosing the most frequent translation. Finally, some could be resolved only by using some form of logical reasoning, but only 3% of the cases were found to fall into this category. The most surprising result was that none of the methods alone would be sufficient. In fact, to achieve 95% accuracy, a combination of all of the methods (except logical reasoning), would be required.

Chapter 4 discussed the difficulty of determining when trigger words could be used to indicate a translation should be chosen other than the most frequent translation. An experiment was discussed that indicated that a neural network might be able to extract cooccurrence classes from cooccurrence pairs by developing an internal feature representation in its weights that would appear in the activations of hidden units. The application of this kind of network to contexts of the word 'bank' in dictionary definitions showed that appropriate

training of the network could lead to over 90% correct selection of word sense in previously unseen text.

Chapter 5 presented results based on parallel English and French texts that showed that cooccurrence data can be successfully used to isolate trigger information for the resolution of lexical ambiguity. Neural networks were successfully trained to mirror this semantic cooccurrence information.

The next section will introduce an algorithm for combining the methods discussed in chapter 3, utilizing the neural network algorithm introduced in chapter 4. Section 6.3 will discuss lexical information required by the algorithm. Section 6.4 will discuss the order and interaction of the steps of the algorithm.

## **6.2 The Algorithm**

The steps of the following algorithm include the methods discussed in chapter 3, and include the neural network algorithm of chapter 4:

- 1) If the ambiguous word occurs in an idiom, translate the entire phrase with its idiomatic meaning; otherwise
- 2) Attempt to reduce the number of possible senses by using syntactic constraints; if only one sense is left, use the appropriate translation; otherwise
- 3) Reduce the number of possible senses by taking into account argument preferences of verbs and prepositions, and modifier preferences of adjectives and prepositions; if only one sense is left, use the appropriate translation; if no senses are left, ignore the preferences and continue; otherwise
- 4) If the word has already occurred in the text, use the same translation used in its previous occurrence; otherwise
- 5) If any of the senses are marked with the same technical area as the text being translated, eliminate other senses; if only one is left, use the appropriate translation; otherwise
- 6) Present context words as activations to input units of a neural network trained on cooccurrence pairs from parallel texts, as in

chapters 4 and 5; pick the target translation with the highest activation.

### **6.3 The Order and Interaction of Steps in the Algorithm**

The purpose of the algorithm is to choose the correct word sense in the greatest majority of cases. The examples in this section are given to show the rationale behind the ordering of the steps.

#### 6.3.1 Idioms and Syntax

Idioms may need to be marked with some syntactic conditions.

For example,

John kicked the buckets.

should not mean that John died. Similarly, in

John saw horses.

the idiom 'saw horse' should not take precedence over the normal SVO reading. Nevertheless, idioms take precedence over normal syntactic conditions. For example, in a sentence without idioms like

John took care of the impressive grounds.

countability indicates that 'grounds' should be the sense meaning 'land around a building', rather than 'earth'. But in

John took care of the coffee grounds.

the same countability conditions apply, but the idiom clearly takes precedence.

### 6.3.2 Idioms and Preference



Fixed idiomatic phrases usually retain their meanings even when they appear in positions where a semantic attribute of the head word alone would satisfy a preference. For example, in the sentence

The snow man was asking for immediate attention.

'snow man' is still the best reading, even though 'ask' would normally prefer a human subject.

### 6.3.3 Idioms and Discourse Memory

A word may appear by itself in a text, and be used later as part of an idiom.

The engine had needed a new gasket. As soon as it was fixed, the fire engine was ready to go again.

The idiomatic meaning takes precedence in such a case.

#### 6.3.4 Idioms and Technical Tags

The following could be a story problem in a mathematics text:

If a coffee table is  $2\frac{1}{2}$  feet wide and five feet long, how many potted plants ten inches in diameter can it hold?

Although 'table' would usually mean a chart of figures in a mathematics text, the idiom 'coffee table' clearly takes precedence.

#### 6.3.5 Idioms and Trigger Words

Idiomatic readings take precedence over meanings implied by trigger words.

The water tank was blown up by a bazooka.

Although bazookas have more to do with military vehicles than storage containers, 'water tank' is clearly the intended meaning.

### 6.3.6 Syntax and Preference

Even if a preference is met more strongly by one reading than another, syntactic constraints may force the less preferred reading.

The man saw a two-by-four.

Although a piece of wood would satisfy the material preference of the verb 'to saw', this 'saw' has to be the past tense of 'see', since the third person present of 'to saw' would be 'saws'.

### 6.3.7 Syntax and Discourse Memory

Syntactic issues can force a meaning different than the one used previously in a text.

The gardener could scarcely get a shovel in the ground during the frozen months. But the owner insisted the grounds be immaculate year round.

Here, the 'grounds' clearly refers to something other than the mass meaning of 'ground'.

### 6.3.8 Syntax and Technical Tags

Sometimes syntax indicates a different sense than the one which would be more likely in a technical text. For example, 'time' might be expected to be the physical dimension in a high school physics text. But in the sentence

Comets pass near the earth several times each decade.

the countability makes it clear that 'time' means 'occurrence'.

### 6.3.8 Syntax and Trigger Words

Syntactic constraints override weaker influences by trigger words.

The carpenter saw where he had put his hammer.

Here, the 'see' meaning wins, even though 'carpenter' and 'hammer' are closely related to the verb meaning 'to saw'.

#### 6.3.9 Preference and Discourse Memory

Preference information can override a word sense used previously in a text.

The sergeant drafted a memo to his commander, saying, "Today, we drafted ten more men."

Both occurrences of 'draft' are determined by preferences, but the second gets a different reading than the first one, which has become the sense remembered in the discourse.

#### 6.3.10 Preference and Technical Tags

Preferences can take precedence over technical senses. In a military document, one would expect

the verb 'draft' to be used in its military meaning of 'recruit', but in

The sergeant drafted a memo.

the correct sense means something like 'outline' or 'sketch'.

#### 6.3.11 Preference and Trigger Words

Preferences usually have a more binding influence than trigger words.

The astronomer married a star.

Although they spend much of their time studying celestial objects, even astronomers normally prefer marrying humans.

#### 6.3.12 Discourse Memory and Technical Tags

The interaction of these two steps is quite rare, since the previous occurrence of a word in a technical document is usually the technical sense,

if it has one. However, in a military text containing

Two water tanks were punctured. Each tank lost over 20 gallons.

the latter 'tank' would have the same meaning as the non-technical sense in the first sentence.

#### 6.3.13 Discourse Memory and Trigger Words

Words are often introduced in a context which makes their intended sense clear, but are later used without clear context. In these cases, the neural network will probably default to the most frequent sense, which may not be the one originally introduced.

Wise fisherman usually fish on the south bank, because the underbrush gives more hiding places for the fish. The less experienced often prefer the other bank, because it is easier to get to.

Although trigger words could conceivably indicate a new sense correctly, data from the survey of Chapter 3 indicates that once a sense is introduced, the word appears with the same meaning in almost all cases. If the original sense is not the default sense, the network would probably choose the wrong meaning in many of the later occurrences.

#### 6.3.14 Technical Tags and Trigger Words

Technical tags can overcome spurious triggering. In a sports column,

The rain had little effect on the pitcher.

'pitcher' is clearly a baseball player, even though pitchers often contain water and rain is made of water.

### 6.4 Lexical Information Required by the Algorithm

One requirement in a practical Machine Translation system is the ability for end users to enter new words. According to Mark Liberman,



although 12,000 words cover about 95% of the words in English text, it is nearly impossible to cover the other 5% with any dictionary of reasonable size. This estimate was based on 15 million words of English text from the AP wire service, in which he found a total of about 100,000 words (other than capitalized words, which were assumed to be names). He also reported that the data revealed no asymptotic behavior; above the 15 million words, each additional million words of text added about the same number of new words (Lieberman 1989). One might conclude from his report that it is nearly impossible to provide a dictionary that will contain all of the words that an end user may ever need. Particular end users also tend to have their own jargon and often coin new words, so it is essential for the end user to have the ability to add new words to the translation dictionary.

Naturally, the algorithm for resolving lexical ambiguity depends on having correct information in the dictionary. Users can usually enter idiom and syntactic information if sufficient examples and helps are given, and once technical areas have been defined, it is easy to add technical tags.

The preference information, however, may be somewhat more difficult. Some have approached this problem by creating knowledge bases in which each word sense has its own knowledge frame (such as in Collative Semantics), and the frames are organized in a semantic hyponymy hierarchy (sometimes called an ISA hierarchy). Preferences can either be matched by requiring a sense frame to have certain features, or by requiring the sense frame to be subordinate to the sense frame of some preference category in the hierarchy. Because of inheritance mechanisms in the knowledge base, these may not realize the same results. Typically, a sense frame has certain features of its own, and inherits others from parents in the ISA hierarchy; if any conflicts arise, the features of the sense frame take precedence. For example, the sense frame of 'mammal' might indicate something about the ability to walk, but although a 'whale' ISA 'mammal', 'whale' would presumably have some feature indicating it does not have the ability to walk, and this would take precedence over conflicting information inherited from 'mammal'. The sets of features and values available to be included in

different knowledge bases vary widely, and it is extremely difficult to get uniformity among different data entry operators creating a knowledge base. Requiring end users to have the expertise to decide exactly what features and values are required on a new entry may well be too much to ask, since there would have to be such a large set to choose from.

A simpler approach may be sufficient for lexical ambiguity resolution, however. Most of the information that would appear in such a knowledge base would never actually be used by the algorithm. Although it is beyond the scope of this work to determine exactly, it seems very unlikely that the classes of preferences required by actual predicates and modifiers of a language would even nearly approach the number of words in a language, let alone the number of word senses. If one could set up an ISA hierarchy of just the preference categories that are actually required in the language, this should be considerably simpler than an entire knowledge base.

It would be unlikely that new words would require too much modification to this hierarchy, because

most new words would be nouns rather than predicators. Once the hierarchy was determined, sufficient training should enable end users to be able to determine categories of new words to be added to the dictionary.

The creation of training data for the neural network is also something that end users could do. All that is required is a sufficient quantity of parallel text, and the algorithms for creating the network are completely automatic. Incremental training on new data is also possible.

### **6.5 Inevitable Errors**

Human translators often balk at the idea that a computer can ever even approach the task of translation. Translation is an art, and language is the ultimate mirror of human intelligence. Mapping between the thought structures of different cultures is often difficult even for human translators. How can a machine ever hope to even attempt such a task?

Claude Piron, a translator who had experience at the United Nations, spoke at the New Directions in Machine Translation Conference held before COLING '88. He gave the following example:

He could not agree with the amendments to the draft resolution proposed by the delegation of India.

(Piron 1988)

In this case, another translator had assumed that it was the resolution that had been submitted by the delegation of India. Mr. Piron checked, however, and determined that it was only the amendments that were proposed by the delegation of India. This information was not in the document, and surely could not be expected to be in a world knowledge base. Although the document could have been written more plainly, the translator had no control over the quality of the documents he was asked to translate. Some translations were required at night, and the translators were not allowed to contact the authors. Mr. Piron also reported a case in which he wrote a letter to ask an author his original intent, but was

informed that the author was no longer living. In such cases, the translator had to guess.

Mr. Piron also stated that good translators can translate about 150 words per minute, and often read directly into a dictaphone in the target language until they meet a big problem, which may take hours or even days to research. It is unreasonable to think that machines, even programmed with vast amounts of world knowledge, will be able to overcome these kinds of problems. The correct translation may well depend on details of events, not necessarily given in the text, that have occurred more recently than the building of the knowledge base, which is necessarily limited to the time frame and perspectives of those who built it.

The conclusion has to be that no machine will ever translate perfectly, any more than a human can. There will inevitably be mistakes. The only question is what kind of mistakes, and how many.

There will be cases in which the algorithm of this chapter gives the wrong answer. The goal of the algorithm, however, is to give the correct answer in the highest possible percentage of cases. It is possible that a system using a world knowledge

base could be more accurate in resolving lexical ambiguity. But how much more accurate? The data of chapter 3 indicate that only about 3% of the ambiguities in real text require logical reasoning. Surely even a system based on a world knowledge base will make mistakes when faced with the kinds of problems described above. Considering the major investment in time and resources required to produce a world knowledge base, and keep it current, such a system may never be economically feasible, if it is even possible. Moreover, the assumption that a system based on a world knowledge base would be more accurate is merely a conjecture until it can be proven to be so.

In the meantime, the algorithm given here is a practical method that can be utilized by anyone attempting to build a machine translation system. The data in chapter 3 indicates that a system that ignores any of the major categories of ambiguities discussed will miss from 3% to 9% of the cases that will be encountered in real text. The algorithm is a practical approach to recognizing categories of potential lexical ambiguities and integrating methods for their resolution.

## 6.5 Summary

One of the most difficult problems encountered in machine translation is the resolution of lexical ambiguities. Many schemes have been tried to meet this problem, but most have lacked perspective about the kinds of ambiguities that actually exist in real text; this may have led to the claim that the machine must understand the text to be even moderately successful. The data of chapter 3 partitioned naturally occurring ambiguities into classes. This showed that many of the ideas that had been tried before (a sample of which were discussed in chapter 2) were on the right track for certain classes of ambiguities. It also showed that none of the methods for resolving ambiguities was sufficient by itself.

The insight of this thesis has been to identify the naturally occurring classes of lexical ambiguity (of content words), and to select from known methods of resolution for those classes, and then to combine the methods into a single resolution algorithm that incorporates the insights of many previous workers.



Because one of the more difficult interactions of methods was found to be between using trigger words and simply using the most frequent translation, a new method using semantic cooccurrence networks was introduced in chapter 4. Cooccurrence data gathered from parallel texts was shown to be useful for detecting the interaction between trigger words and frequency. Semantic cooccurrence networks were successfully trained to mirror the cooccurrence data.

The result is an algorithm that can be utilized in a practical machine translation system.

Future research should include more rigorous testing of the algorithm, with various language pairs and more lengthy parallel texts. The types of errors remaining could then be classified, and the algorithm improved. The parameters of the neural networks should also be tuned, perhaps by varying the number of hidden units, or the strengths of contextual input units based on distance from the ambiguous word.

## Appendix A

**TEXTS WITH CLASSIFIED AMBIGUITIES**

The texts analyzed and described in Chapter 3 are presented here. Each word with an intra-category ambiguity is annotated with its grammatical category, the number of intra-category senses that were listed in the English-Japanese translation dictionary, and the methods of disambiguation that might be successfully used.

The notation for each word is as follows:

/cnxyz

where

C is a grammatical category

V verb

N noun

A adjective

n is the number of senses within that category

x, y, and z are methods that could be successfully used alone

A spreading activation

C cover term

D	discourse memory
F	frequency
I	idiom
L	logical reasoning required
S	selectional restrictions
T	technical term
X	syntax

Note that an ambiguity that can be resolved by a combination of methods is marked (x+y).

### **A.1 Newspaper Text**

A huge icebreaking barge/N4F began its journey to rescue three trapped whales as scientists/N2FC worried/V3X that plunging temperatures/N2FC and polar/A2I bears would threaten the mammals they have named/V2X Bonnet, Crossbeak and Bone.

At dawn/N2X, a National/N2I Guard/N3I helicopter rigged/V2S to tow the 185-ton "hover-barge" was to resume/V2FS the 230-mile trip/N2AFS along the desolate/A2FS Arctic coast from Prudhoe Bay. It moved/V3X about five miles

Wednesday through sand bars/N2I, mud and shallow water/N2F.

The 24- to 30-foot/N3X-long/A2S California gray whales, whose species/N2AS is endangered, became trapped in the ice/N2AS two weeks/N2C ago while migrating/V2S south.

Eskimo whalers/N2F using chain/N2I saws in sub-zero temperatures/N2F have been cutting holes/N2F in the thick/A4AFS Arctic Ocean/N2C ice/N2ADF to help/V3CF the mammals breathe. They got a boost/N3I Wednesday when two brothers-in-law from Minnesota brought six \$400 de-icers to the \$500,000 rescue effort/N2CF.

Greg Ferrian and Rick Skluzacek, of Lakeland, flew to Alaska at their own expense to demonstrate/V2AS the devices, invented by Skluzacek's father/N3F, when rescue coordinators/N3C politely refused their offer/N2F to help/N3C.

Early today/N2X, under the skeptical supervision of the coordinators/N3CD, they quickly cleared/V2X ice/N2ADF and slush/N3AF from the two breathing holes/N2I, accomplishing in hours/N2X what had been taking the rescue team, using chain/N2I

saws, pick-axes and steel bars/N3AS, a day/N2X to do.

The cylindrical de-icer, about 9 inches around and 14 inches long/A2DS, floats in the water/A2X suspended beneath a styrofoam slab. A small/A2FS, double-blade propeller pulls warmer water/N2X up from below and emits it with 34 pounds/N2X of thrust/N2S.

Within 24 hours/N2X, the breathing holes/N2I should be about 30 feet by 70 feet -- three to four times/N2X the original/N2(C+S) size/N2C, Skluzacek said.

The devices are used in marinas to keep boats/N2C from becoming frozen/V4FS in the winter, Ferrian said.

The tired/A3F whales can survive for several more weeks/N2X despite being battered/V2L and bleeding from grating against jagged ice/N2DF, said John Lien, professor/N2C of animal/N3(C+F) behaviour at Memorial University in Canada.

"They can bleed a barrel/N2C and still be fine/A2S," said Lien, who has helped/V3C free whales from ice/N2ADF for 20 years/N3F. "We're talking/V2X

about a big animal/N3(C+F)D here, between five to 20 tons/N2C."

The whales also may have to contend with polar/A2I bears. Reporters flying/V3S by helicopter Wednesday spotted/V3FS five prowling the ice/N2DF a few miles from the whales.

"There are polar/A2I bears that are certainly going to be attracted to this," said Ron Morris of the National/A2X Marine Fisheries Service/N3I. "If they're going to take/V4L the whales, we're not going to stop them."

(Daily Herald, 20 Oct 1988, page 1)

## **A.2 Extracts from an Article on Artificial Intelligence**

Just constructing a knowledge/N3I base/N3I is a major intellectual research problem/N2AF .... We still know far too little/A2X about the contents/N2F and structure/N2S of common-sense/N3I knowledge/N2AF. A "minimal" common-sense system/N3F must "know" something about cause-effect, time/N3C, purpose/N3X,

locality/N3CF, process/N3(C+F), and types/N2X of knowledge/N2FD .... We need/V2C a serious/A4(A+S) epistemological research effort/N2C in this area/N4A.

Minsky's naivete/N3F and faith/N3CF are astonishing. Philosophers/N2F from Plato to Husserl, who uncovered all these problems/N2DF and more, have carried on serious/A4CD epistemological research in this area/N3AD for two thousand years/N3X without notable/A2S success. Moreover, the list Minsky includes in this passage/N3AF deals only with natural/A2X objects/N3F, and their positions/N3FS and interactions. As Husserl saw/V2X, intelligent behavior also presupposes a background/N3A of cultural practices/N3A and institutions/N3A. Observations in the frame/N3T paper/N4T such as: "Trading/V2F normally occurs in a social context of law/N3C, trust/N3X, and convention/N3F. Unless we also represent/V3T these other facts/N2X, most trade/N3AF transactions/N3AF will be almost meaningless" (p. 34/102) show/V5X that Minsky has understood this too. But Minsky seems oblivious/A3L to the hand-waving optimism of

his proposal/N2C that programmers/N2T rush in where philosophers/N2DF such as Heidegger fear/V2F to tread, and simply make/V4X explicit/A2S the totality/N3F of human practices/N3D which pervade our lives/N3C as water encompasses the life/N3D of a fish.

...

No doubt many of our social activities are stereotyped, and there is nothing in principle/N3I misguided in trying/V3X to work out primitives/N2T and rules/N3T for a restaurant game/N3F, the way/N3X the rules/N3S of Monopoly/N2X are meant to capture a simplified version/N2F of the typical/4AF moves/N3AF in the real estate/N3I business/N3F. But Schank claims that he can use this approach/N2C to understand/V3X stories/N3F about actual restaurant-going -- that in effect/N3I he can treat/V2C the sub-world of restaurant going/V3X as if it were an isolated micro-world. To do this, however, he must artificially limit the possibilities/N2X: for, as one might suspect, no matter/N3I how stereotyped, going/V3X to the restaurant is not a self-contained game/N3DF but a highly variable set/N2X of behaviors which open out



into the rest/N3I of human activity. What "normally" happens when one goes/V3X to a restaurant can be pre-selected and formalized by the programmer/N2T as default/N3T assignments/N3FT, but the background/N3D has been left out so that a program using such a script/N3T cannot be said to understand/V3X going/V3X to a restaurant at all. This can easily be seen by imagining/V2X a situation/N3F that deviates from the norm/N2C. What if when one tries/V3X to order/V2X he finds/V4X that the item/N2S in question/N3I is not available/A4C, or before paying/V2X he finds/V4X that the bill/N3AF is added/V3X up wrongly? Of course/N3I, Schank would answer/V2X that he could build/V3X these normal ways/N3F restaurant-going breaks/V3I down into his script/N3DT. But there are always abnormal ways/N3DF everyday activities can break/V3DI down: the juke box/N2I might be too noisy, there might be too many flies on the counter/N2F, or as in the file Annie Hall, in a New York delicatessen/N3C one's girl/N3I friend/N2I might order/V2X a pastrami sandwich on white/A3(F+S) bread with mayonnaise. When we understand/V3X going to a restaurant we understand/V3X how to cope with even these abnormal

possibilities/N2X because going/V3X to a restaurant is part/N3F of our everyday activities of going/V3X into buildings/N2X, getting things/N2F we want/V6F, interacting with people/N3F, etc.

To deal with this sort of objection Schank has added/V3X some general rules/N3DT for coping with unexpected/A2X disruptions/N2X. The general idea/N3F is that in a story/N3DF "it is usual/A2X for non-standard/N3FS occurrences/N2X to be explicitly mentioned" (Schank and Abelson, 1977; p. 51); so the program can spot/V3F the abnormal events/N3C and understand/V3X the subsequent events/NCD as ways/N3X of coping with them. But here we can see/V3F that dealing with stories/N3DF allows/V3X Schank to bypass the basic/A3C problem/N2DF, since it is the author's understanding/N3F of the situation/N3DF which enables him to decide/V3C which events/N3CD are disruptive enough to mention.

This ad hoc way/N3X of dealing with the abnormal can always be revealed/V2S by asking/V3I further/A2S questions/N3I, for the program has not understood a restaurant story/N3DF the way/N3DF people/N3DF in our culture do, until it can answer

such simple/A3L questions/N2DFS as: When the waitress came to the table/N2FS, did she wear/V2S clothes? Did she walk forward or backward? Did the customer eat his food with his mouth/N2X or his ear/N2X? If the program answers, "I don't know," we feel/V2X that all of its right/A6(C+S) answers were tricks/N3C or lucky guesses/N2A and that it has not understood anything of our everyday restaurant behavior. The point/N3F here, and throughout, is not that there are subtle things/N3S human/N1I beings can do and recognize which are beyond the low-level understanding/N3F of present/A4(F+S) programs, but that in any area/N3D there are simple/A3AD taken-for-granted responses/N3F central/A2X to human understanding/N3DF, lacking which a computer program cannot be said to have any understanding/N3DF at all. Schank's claim/N3X, that "the paths/N2F of a script/N3DT are the possibilities/N2X that are extant in a situation/N3DF" (1975b; p.132) is insidiously misleading. Either it means/V2X that the script/N3DT accounts for the possibilities/N2X in the restaurant game/N3D defined by Schank, in which case/N3C it is true/A3(F+S) but uninteresting; or he

is claiming that he can account for the possibilities/N2X in an everyday restaurant situation/N3DF which is impressive but, by Schank's own admission/N3L, false.

Real short/A4F stories/N3DF pose a further problem/N2DF for Schank's approach/N2CD, in a script/N3DT what the primitive actions/N3F and facts/N2T are is determined/V2X beforehand, but in a short/A4DF story/N3DF what counts/V2X as the relevant facts/N2DT depends on the story/N3DF itself. For example, a story/N3DF that describes/V2F a bus trip/N2AF contains in its script/N3DT that the passenger thanks the driver/N3F (a Schank example). But the fact/N2DT that the passenger thanked the driver/N3DF would not be important/A3S in a story/N3DF in which the passenger simply took the bus as a part/N3F of a longer/A2S journey, while it might be crucially important/A3D if the story/N3DF concerned a misanthrope who had never thanked anyone before, or a very law-abiding young man/N3F who had courageously broken the prohibition/N3C against speaking/V2X to drivers/N3DF in order to speak/V2X to the attractive/A2C woman driving/V4S the bus. Overlooking this point/N3DF,

Schank claimed at a recent meeting/N3C that his program, which can extract death/N3C statistics/N3X from newspaper accident reports/N2AF, had answered my challenge/N2L that a computer would count/V2X as intelligent only if it could summarize a short/A4DF story/N3DF. But Schank's newspaper program cannot provide/V2F a clue concerning judgements/N3L of what to include in a story/N3DF summary because it works/V4L only where relevance/N2F and significance have been predetermined, and thereby avoids dealing with the world/N2F built up in a story/N3DF in terms/N3I of which judgments/N3DFS of relevance/N2DF and importance are made.

Nothing could ever call/V4I into question/N2I Schank's basic assumption that all human practice/N3X and know-how is represented/V3DT in the mind/N3A as a system/N2C of beliefs/N3C constructed from context-free primitive actions/N3DF and facts/N2DT, but there are signs/N3A of trouble/N3C. Schank does admit that an individual's "belief/N3CD system/N3DF" cannot be fully elicited from him; although he never doubts that it exists and that it could in principle/N3I be represented/V3DT in his formalism. He is therefore led to the desperate/A2F

idea/N3DF of a program which could learn/V2S about everything from restaurants to life/N3DF themes the way/N3DF people/N3DF do. In one of his papers/N4DT he concludes/V3X:

We hope/V3X to be able to build/V3X a program that can learn/V2DS, as a child/N2F does, how to do what we have described/V2DF in the paper/N4DT instead of being spoon-fed the tremendous information necessary.

In any case/N3I, Schank's appeal/N3FS to learning is at best another evasion/N2X. Developmental/A2I psychology/N2I has shown that children's learning does not consist merely in acquiring more and more information about specific/A2X routine/N3CF situations/N3CD by adding new primitives/N2DT and combining old/A3S ones as Schank's view/N3A would lead one to expect/V2X. Rather, learning of specific/A2X details/N2C takes place/N3I on a background/N3AD of shared practices/N3D which seem to be picked/V3I up in everyday interactions not as facts/N2DT and beliefs/N3DF but as bodily skills for coping with the world/N2D. Any learning presupposes

this background/N3D of implicit/A3F know-how which gives/V4S significance to details/N2CD. Since Schank admits that he cannot see/V3X how this background/N3D can be made explicit/A2DS so as to be given/V4S to a computer, and since the background/N3D is presupposed for the kind/N3X of script/N3DT learning Schank has in mind/N3I, it seems that his project of using preanalyzed primitives/N2DT to capture common sense/N3I understanding/N3AD is doomed.

...

... AI researchers have consistently run/V4I up against the problem/N2DF of representing/V3DT everyday context. Work/N3X during the first five years/N3X (1967-1972) demonstrated the futility/N2F of trying/V3X to evade the importance of everyday context by creating/V2F artificial/A2F gamelike/N3DF contexts preanalyzed in terms/N3I of a list of fixed-relevance features/N3T. More recent work/N3X has thus been forced to deal directly with the background/N3D of common-sense know-how which guides our changing sense/N3S of what counts/V2X as the relevant facts/N2DT. Faced with this necessity/N3C researchers have implicitly tried/V3X to treat/V3X

the broadest context or background/N3AD as an object/N3DF with its own set of preselected descriptive/A2F features/N3DT. This assumption, that the background/N3D can be treated/V3X as just another object/N3DF to be represented/V3DT in the same sort of structured description/N2T in which everyday objects/N3DF are represented/V3DT, is essential/A4C to our whole philosophical/A2F tradition/N2C. Following Heidegger, who is the first to have identified and criticized this assumption, I will call/V4X it the metaphysical assumption.

... Just as it seems plausible that I can learn/V2S to swim by practicing/V2F until I develop/V3X the necessary patterns of responses, without representing/V3DT my body/N3AF and muscular movements/N3A in some data structureN2T, so too what I "know" about the cultural practices/N3DF which enables me to recognize and act/V4F in specific/A2X situations/N3DF has been gradually acquired through training in which no one ever did or could, again on pain/N2C of regress/N3C, make/V4X explicit/A2DS what was being learned/V2X.

...



The idea/N3DF that feelings/N3DF, memories/N3X, and images/N3C must be the conscious/A3S tip/N3C of an unconscious/A2C framelike/N2DT data structure/N2DT runs/V4I up against both prima facie evidence/N2C and the problem/N2DF of explicating the ceteris paribus conditions/N3A. Moreover, the formalist assumption is not supported by one shred of scientific/A2CF evidence/N2CD from neuro-physiology or psychology/N2A, or from the past successes of AI, whose repeated failures/N2X required/V4X appeal/N3DFS to the metaphysical assumption in the first place/N3I.

... If one thinks/V2X of the importance of the sensory-motor skills in the development/N2C of our ability to recognize and cope with objects/N3CD, or of the role of needs/N3C and desires in structuring all social situations/N3A, or finally of the whole/A3X cultural background/N3AD of human self-interpretation involved in our simply knowing how to pick/V3I out and use chairs/N3FS, the idea that we can simply ignore this know-how while formalizing our intellectual understanding/N3DF as a complex system/N3DF of facts/N2DT and rules/N3DT is highly implausible.

(Dreyfus 1981)

### **A.3 A Religious Text**

Sometime in the second year/N3X after our removal/N3C to Manchester, there was in the place/N3F where we lived/V2F an unusual excitement/N3C on the subject/N3C of religion. It commenced/V2X with the Methodists, but soon became general among all the sects in that region of country/N3X. Indeed, the whole district/N3C of country/N3X seemed affected/V2C by it, and great/A4AS multitudes/N3C united themselves to the different religious parties/N3C, which created/V2F no small/A2S stir/N3C and division/N3C amongst the people/N3F, some crying/V2X, "Lo, here!" and others, "Lo, there!" Some were contending/V2X for the Methodist faith/N3A, some for the Presbyterian, and some for the Baptist.

For, notwithstanding the great/A4S love/N3C which the converts to these different faiths/N3DF expressed at the time/N3I of their conversion/N3T, and the great/A4DS zeal manifested by the respective clergy, who were active in getting/V5I up and promoting this extraordinary/A2L scene/N3C of

religious feeling/N3AF, in order/N4I to have everybody converted, as they were pleased to call/V4X it, let them join/V2ST what sect they pleased; yet when the converts began to file/V2X off, some to one party/N3CD and some to another, it was seen that the seemingly good feelings/N3DF of both the priests/N2C and the converts were more pretended than real; for a scene/N3CD of great/A4DS confusion/N3A and bad feeling/N3X ensued --priest/N2CD contending/V2X against priest/N2CD, and convert against convert; so that all their good feelings/N3DF one for another, if they ever had any, were entirely lost/A4(F+S) in a strife of words/N3X and a contest about opinions.

I was at this time/N3C in my fifteenth year/N3X. My father's family/N2C was proselyted to the Presbyterian faith/N3AD, and four of them joined/V2X that church, namely, my mother/N3CF, Lucy; my brothers Hyrum and Samuel Harrison; and my sister/N2C Sophronia.

During this time/N2CD of great/A4DFS excitement/N2CD my mind/N3A was called/V3I up to serious/A4S reflection and great/A4DS uneasiness/N2F; but though my feelings/N3ADF were

deep/A2F and often poignant, still I kept myself aloof from all these parties/N3CD, though I attended/V2S their several meetings/N3S as often as occasion/N2C would permit. In process/N2C of time/N3CD my mind/N3D became somewhat partial/A2L to the Methodist sect, and I felt/V2A some desire to be united with them; but so great/A4DS were the confusion/N3CD and strife among the different denominations/N2C, that it was impossible/A3L for a person/N3X young as I was, and so unacquainted/A3C with men and things/N2F, to come/V3I to any certain/A5C conclusion/N3I who was right/A6AF and who was wrong/A2AS.

My mind/N3AD at times/N3I was greatly excited/V2X, the cry/N3F and tumult were so great/A4DS and incessant. The Presbyterians were most decidedly/V3X against the Baptists and Methodists, and used all the powers/N3FS of both reason/N3X and sophistry/N2C to prove/V3S their errors/N3L, or, at least, to make/V3X the people/N3F think/V2X they were in error/N3D. On the other hand/N3I, the Baptists and Methodists in their turn/N3X were equally zealous/A2S in endeavoring to

establish/V2S their own tenets and disprove all others.

In the midst of this war/N2A of words/N3X and tumult of opinions, I often said to myself: What is to be done? Who of all these parties/N3CD are right/A6AD; or, are they all wrong/A2ADF together? If any one of them be right/A6AD, which is it, and how shall I know it?

While I was laboring/V2C under the extreme difficulties caused/V2X by the contests of these parties/N3CD of religionists, I was one day/N2X reading the Epistle of James, first chapter and fifth verse/N3X, which reads/V3CD: If any of you lack wisdom/N2F, let him ask/V3X of God, that giveth/V4X to all men/N3DF liberally, and upbraideth not; and it shall be given/V4X him.

Never did any passage/N4A of scripture come with more power/N3A to the heart/N3L of man/N3X than this did at this time/N3CD to mine. It seemed to enter with great/A4DS force/N2C into every feeling/N3ADF of my heart/N3AD. I reflected/V4(C+S) on it again and again, knowing that if any person/N3DF needed wisdom/N2DF from God, I did; for how to act/V4F I did not know, and unless I could

get/V5X more wisdom/N2DF than I then had, I would never know; for the teachers of religion of the different sects understood the same passages/N4AD of scripture so differently as to destroy all confidence/N3C in settling/V4S the question/N3F by an appeal to the Bible/N3C.

At length I came to the conclusion/N3C that I must either remain in darkness and confusion/N3C, or else I must do as James directs/V3F, that is, ask/V3X of God. I at length came to the determination/V3C to "ask/V3X of God," concluding/V3X that if he gave wisdom/N2DF to them that lacked wisdom/N2DF, and would give/V4DF liberally, and not upbraid, I might venture/V2F.

So, in accordance with this, my determination/N3CD to ask/V3X of God, I retired/V2X to the woods to make/V4X the attempt. It was on the morning/N2C of a beautiful, clear/A7S day/N2X, early in the spring/N3X of eighteen hundred and twenty. It was the first time/N3X in my life/N3C that I had made such an attempt, for amidst all my anxieties I had never as yet made the attempt to pray vocally.

After I had retired/V2X to the place/N3DF where I had previously designed/V3X to go/V3X, having

looked around me, and finding myself alone/A2X, I kneeled down and began to offer up the desires of my heart/N3ADF to God. I had scarcely done so, when immediately I was seized/V2S upon by some power/N3ADF which entirely overcame me, and had such an astonishing influence/N2I over me as to bind/V2FS my tongue/N2S so that I could not speak/V2X. Thick/A4FS darkness gathered around me, and it seemed to me for a time/N3CD as if I were doomed to sudden destruction.

But, exerting all my powers/N3ADF to call/V3I upon God to deliver me out of the power/N3DF of this enemy/N2C which had seized/V2X upon me, and at the very moment/N2C when I was ready/A4L to sink/V2F into despair and abandon myself to destruction -- not to an imaginary ruin/N3L, but to the power/N3DF of some actual being from the unseen world/N2F, who had such marvelous power/N3DF as I had never before felt/V2D in any being -- just at this moment/N2DF of great/A4DS alarm/N2X, I saw/V2L a pillar/N2X of light/N3X exactly over my head/N3X, above the brightness/N3A of the sun, which descended/V2L gradually until it fell upon me.

It no sooner appeared/V2X than I found myself delivered from the enemy/N2CD which held me bound/A4F. When the light/N3D rested upon me I saw/V2X two Personages/N3AF, whose brightness/N3A and glory/N3AT defy all description/N2F, standing above me in the air/N3F. One of them spake unto me, calling/V4X me by name/N3I and said, pointing/V2X to the other -- This is My Beloved/A3L Son. Hear Him!

My object/N3X in going/V3X to inquire of the Lord/N3X was to know which of all the sects was right/A6DF, that I might know which to join/V3DT. No sooner, therefore, did I get/V5X possession/N2X of myself, so as to be able to speak/V2X, than I asked/V3X the Personages/N3D who stood above me in the light/N3D, which of all the sects was right/A6DF.

I was answered that I must join/V2DT none of them, for they were all wrong/A2ADF; and the Personage/N3D who addressed me said that all their creeds/N2C were an abomination/N2X in his sight/N3F; that those professors/N2F were all corrupt; that: "They draw/V2I near to me with their lips/N2X, but their hearts/N3D are far from me; they teach/V2X for doctrines the commandments/N2C of men/N3DF, having a



form/N3(C+S) of godliness, but they deny the power/N3DF thereof."

(Smith 1978, v. 5-19)

#### **A.4 A Text from a Computer Manual**

Scope and Extent/N3T

Naming/V2S something and then referring to that thing/N2F by its name/N3F at some other place/N3F or time/N3C is a fundamental/A2C part/N2F of every language/N2C; be it a natural/A2I language/N2I like English/N2X, or an artificial/A2C language/N2CD like COMMON LISP/N2I. Although English/N2X and COMMON LISP/N2I are very different languages/N2CD, their basic/A3F concepts of naming/V2ADF and referring (or referencing) are quite similar/A2X.

In COMMON LISP/N2I, every entity/N2X can have a name/N3DF. When one wants to refer to an entity/N2X, one uses its name/N3DF. As in English/N2X, a name/N3DF may refer to different entities/N2X at different places/N3DF and times/N3CD. The word/N3X President/N3X exemplifies the context-sensitive nature of names/N3DF in English/N2X. President/N3D refers to a different

person/N3F in different places/N3DF (e.g., Gold/N3I Hill Headquarters, Washington, D.C., Paris). Within the same place/N3DF, President/N3D may also refer to different people/N3F over the course/N4C of time/N3CD. For example/N2I, within a single business/N3F meeting/N3F (held in 1984), President/N3D may refer to Stan Curtis, Ronald Reagan, and Francois Mitterand over the course/N4CD of the meeting/N3DF.

In COMMON LISP/N2I, the region in which a name/N3DF refers to a particular/A3S entity/N2X is called the scope of the name/N3DF. The interval/N2AF of time/N3CD during which a name/N3DF refers to a particular/A3DS entity is called/V4X the extent/N3DT of the name/N3DF. Scope concerns the spatial, textual, or lexical representation/N3T of a LISP/N2T form/N3T (e.g., its appearance/N2AF on a piece/N3I of paper/N4X). Extent/N3DT concerns the time/N3CD during which the form/N3DT is being evaluated.

Before a name/N3DF can refer to an entity/N2X, however, a correspondence/N3T between the name/N3DF and that entity/N2X must be established/N2L. Only functions/N3T and certain/A5L special/A2F forms/N3DT

(e.g., let) are able to establish/V2D names/N3DF. The scope and extent/N3DT of a name/N3DF are relative/A3X to the form/N3DT which established/V2D it. The scope of the name/N3DF can be limited to or independent/A2C of the textual region which the establishing/V2D form/N3DT encloses. Likewise, the interval/N2ADF of the time/N3CD during which the establishing/V2D form/N3DT is being evaluated.

These various kinds/N2DF of scope and extent/N3DT are defined in COMMON LISP/N2I as follows/V2I:

#### Lexical Scope

A name/N3DF which has lexical scope can only be used within the lexical (i.e., textual) region of the establishing/V2D form/N3DT.

#### Indefinite/A2F Scope

A name/N3DF which has indefinite/A2DF scope can be used anywhere, regardless of the lexical region of the establishing/V2D form/N3DT.

Dynamic Extent/N2DT

A name/N3DF which has dynamic extent/N3DT can only be used during the interval/N2ADF of time/N3CD between the start/N3C and finish/N2C of the evaluation of the establishing/V2D form/N3DT.

Indefinite/A2DF Extent/N2DT

A name/N3DF which has indefinite/A2DF extent/N3DT can be used at any time/N3CD after being established/V2D, regardless of whether the establishing/V2D form/N3DT is still in the process/N3C of being evaluated.

(GCLISP 1983, pp. 20-21)

## Appendix B

**A PROGRAM FOR WEISS RULES AND TRIGGER COUNTING**

The C program presented here implements two related algorithms. The first is the algorithm of Stephen Weiss as given in section 2.2.7. The second resolves lexical ambiguity by counting the number of trigger words for various senses of an ambiguous word in a given window of context; it picks the one with the highest count, or, if there is a tie, picks the highest frequency sense among those that tied.

Both algorithms are based on having parallel source and target texts available. It is assumed that inflected words have been replaced by base forms, and function words have been removed, so that only base forms of context words remain. Each source base form has an assigned index, and each target base form has an assigned index; each set of indexes starts at 1. There are source and target index files, which map indexes to the assigned base form strings. There are also line files, that mirror the source and target texts, but are composed of numerical indexes instead of strings; lines end with a NULL. There are source

and target line index files; given a line number, they map to the file position in the line file where the given line starts. There is also a usage table. Every occurrence of every source word in the text is given, sorted by the source word. A table index file points to the position in the table file where the lines are recorded in which the given source word occurs; if it occurs  $n$  times in the line, the line number is given  $n$  times in the table file. These files allow quicker access to the data in the correlated source and target texts.

In this program, each ambiguous source word (one that can map to more than one target word) in the text is examined. For each instance, the corresponding target segment is searched for possible translations, according to the source-target dictionary. Only single words are handled; there is no idiom processing. When the proper target word is found, the instance is recorded in the Instants array, with the file position of the source segment, the sentence position of the instance, and the index of the correct target translation. The ambiguous word is required to occur the same number of times as its target

translation in the corresponding segment; otherwise, the segment is ignored for that word. If several target words qualify, the one with the highest frequency in the full target text is chosen.

Once an array of instances and their correct translations has been determined for an ambiguous word, the input file of contextual rules is read in. The Weistabl array is indexed by the index of source words, and the value in each position of the array is the correct translation of the ambiguous word in the presence of the source word corresponding to the index. If the value is 0, the source word has no contextual effect. If the value is -1, it is on the exclusion list during training, meaning it has no contextual effect, and no further rule can be made with it. The Weistabl array is cleared and then filled in from the input file. The input file may also specify the default translation for the ambiguous word (stored in variable Devault).

The tryit routine does the training and testing. If the Weiss method was chosen, and there is no input file, Weiss training is done (see the <dbg> parameter and the discussion in section 2.2.7). In any case, each instance of the ambiguous

word is analyzed using either the Weiss method or the trigger counting method, as specified by the user.

The total number of trials and total number of errors for each ambiguous word are recorded and written to a file. The cumulative totals are also written at the end. At higher debug levels, the trigger words that caused certain senses to be chosen are also written in the statistics output file. Another file is written which records the rules which were used.

The parallel texts are represented in an inverted file format, using the following input files:

<stt> This file amounts to a source-target dictionary. Each source and target word are assigned an index. Each source word has 8 (POLYSEEM) possible target translations. For each one, there is an 8-byte structure, the first 4 bytes of which are the index of target word.



- <slin> This is a representation of the source document in which each source word has been replaced by its 4-byte index. Each segment ends with a NULL.
- <slx> This is an index into the <slin> file. It is simply an array of pointers into the <slin> file. The nth 4-byte word in the file points to the nth segment of the source text in <slin>.
- <tlin> This is a representation of the target text similar to <slin>.
- <tlx> This is an index into <tlin>, similar to <slx>.
- <stbl> This lists, for each source word, the lines in which each of its instances occurs; each line is a 4-byte number. If a source word occurs more than once in the same line, the line number occurs the same number of times in this file. The list for each source word is NULL terminated.
- <sdex> This is an index into <stbl>. It is an array, the nth position of which points to the position in <stbl> where the list for the source word with index n begins.

- <swd> This is a list of source words, each one terminated by a 0 byte.
- <swx> This is an index into <swd>. It is an array of museum structures, the nth position of which points to the position in <swd> where the word starts (the freq field is not used).
- <twd> This is a list of target words, like <swd>.
- <twx> This is an index into <twd>, like <swx>.

Output files are:

- <orul> This is a set of contextual rules, in the form
- A <TAB> B <TAB> C
- where A is the ambiguous word, B is the context word, and C is the correct target translation of A. If B is <CNTRL-A><CNTRL-A>, C is the default translation of A.
- <ostat> This lists each ambiguous word in the source text, how many times it was correctly resolved divided by its total

number of instances, and the percentage correct. A cumulative total is given at the end of the file.

Other parameters are:

<wndw> This is the context window. If it is 5, five words on either side of the ambiguous word are processed. If it is 0, all words in the segment are used as context.

<in> This optional file is a file in the same format as <orul>, which is used as the contextual rules for processing.

<dbg> This is a debug level, with three digits of significance XYZ. If Y=1, the Weiss algorithm is used; if there is no input file, the text is used to train; otherwise, the text is tested. If there is an input file, the first X0% of the segments of source and target data are ignored, and the rest is used to test on. If there is no input file, the first X0% is used. Z determines how many debugs are shown.

```

#include <stdio.h>
#include <dos.h>
#include <string.h>

#define YUP 1
#define NOPE 0

#define POLYSEEM 8

#define MAXLINLN 4000
    /* maximum line length for context segments */
#define MAXNSTNC 1000
    /* maximum number of word instances in text */
#define NELFUB 200
    /* buffer length */

#if defined(OS2)
#define INCL_BASE
#include <OS2.H>
#endif

typedef struct wax {
    long freq;
    long wdoft;
} museum;
    /* structure for index files */

typedef struct {
    long philpos;
    /* file position of a source line */
    long answer;
    /* index of correct target sense */
    int sentpos;
    /* position in the source sentence */
} NINSTANC;

char *Insent,*calloc(),huge *halloc();

FILE *Ofp,*Nfp,*Ifp,*Sstfp,*Slxfp,*Slinfp;
FILE *Tlxfp,*Tlinfp,*Sdexfp,*Stblfp;
FILE *Swxfp,*Swdfp,*Twxfp,*Twdfp;

int Showit;
    /* debug level */
int Window;
    /* number of words on each side to use as

```

```

        context; if 0, use whole segments */
int Frumafil;
    /* true if there is an input file */
int Efil;
    /* true if end of input file was reached */
int Weisteki;
    /* true if operation in Weiss rule mode */

long Ssttot,Slxtot,Slintot,Tlxtot,Tlintot;
long Sdextot,Stbltot,Swxtot,Swdtot,Twxtot,Twdtot;
    /* sizes of input files */
long Instop;
    /* number of instances of the ambiguous word */
long Tride;
    /* number of trials */
long Blode;
    /* number of errors */
long *Zazzoo;
    /* for recording local context */
long Inkey,Incon,Intar;
    /* input source, context, and target words */
long Devault;
    /* default translation target word index */
long Lowlin,Hilin;
    /* low and high line in input text to consider
*/
long Zonk[POLYSEEM];
    /* records target translations with most
    frequent first */

NINSTANC *Instants;
    /* array recording instances of the current
    ambiguous word in the source text */

long huge *Ssttab,huge *Slxtab,huge *Slintab;
long huge *Tlxtab,huge *Tlintab;
long huge *Sdextab,huge *Stbltab,huge *Weistabl;
museum huge *Swxtab,huge *Twxtab;
char huge *Swdtab,huge *Twdtab;
    /* input file arrays */
main(argc,argv)
int argc;
char *argv[];
{
    short func,prcent;
#ifdef OS2
    VioSetAnsi(1,0);
#endif
#endif

```

```

if (argc<9)
{
printf("\nweissrul
<dbg><orul><ostat><stt><slx>");
printf("<slin><tlx><tlin>");
printf("\n      <sdex><stbl><swx><swd><twx>");
printf("<twd><wndw>(<in>)");
printf("\n <dbg> =  1, no <in>      ");
printf("show stats using most frequent only");
printf("\n <dbg> = 11, no <in>      ");
printf("learn Weiss rules");
printf("\n <dbg> = Xyz, no <in>      ");
printf("do above to first X0%c of data
lines",37);
printf("\n <dbg> = Xyz, with <in>      ");
printf("do below to all but first X0%c of data");
printf(" lines",37);
printf("\n <dbg> = 11, with <in>      ");
printf("apply Weiss rules");
printf("\n <dbg> =  1, with <in>      ");
printf("apply rules using tally");
printf("\n <wndw> = n use window of ");
printf("n words on each side");
printf("\n <wndw> = 0  ");
printf("use whole line as context");
exit(1);
}
cls();
pos(1,1);
Showit = 0;
if (sscanf(argv[1],"%d",&Showit)==0)
{
printf("\n\nNo debug level in command line: ");
printf("try again");
exit(1);
}
prcent = Showit/100;
/* percentage of file to learn on */
Showit %= 100;
Weisteki = Showit/10;
/* Weiss method or counting method */
Showit %= 10;
if (Showit<0 || Showit>5)
Showit = 0;
if ((Ofp=fopen(argv[2],"w"))==NULL)
{
printf("Error opening output file %s",argv[2]);
}

```

```

    queueut(1);
}
if ((Nfp=fopen(argv[3],"w"))==NULL)
{
    printf("Error opening input file %s",argv[3]);
    queueut(1);
}
indat(argc,argv);
/* read in input files */
makary();
/* allocate and initialize arrays */
Lowlin = 1L;
Hilin = Slxtot;
if (prcent!=0)
{
    if (Frumafil)
        Lowlin = ((long )prcent*Slxtot/10L)+1L;
    else
        Hilin = (long )prcent*Slxtot/10L;
}
/* determines which section of the text to use
*/
lesdoit();
fcloseall();
}

/*****
This routine reads in the input files for the
parallel texts.
*****/
/

indat(filnum,filnams)
int filnum;
char *filnams[];
{
    FILE *ripopen();
    long aboutnow;
    Sstfp = ripopen(filnams[4],&aboutnow,
        &(long huge * )Ssttab,YUP);
    Ssttot = aboutnow/(sizeof(long));
    Slxftp = ripopen(filnams[5],&aboutnow,
        &(long huge * )Slxtab,YUP);
    Slxtot = aboutnow/(sizeof(long));
    Slinftp = ripopen(filnams[6],&aboutnow,
        &(long huge * )Slintab,NOPE);
    Slintot = aboutnow/(sizeof(long));
    Tlxftp = ripopen(filnams[7],&aboutnow,

```

```

        &(long huge * )Tlxtab,YUP);
Tlxtot = aboutnow/(sizeof(long));
Tlinfp = ripopen(filnams[8],&aboutnow,
        &(long huge * )Tlintab,NOPE);
Tlintot = aboutnow/(sizeof(long));
Sdexfp = ripopen(filnams[9],&aboutnow,
        &(long huge * )Sdextab,YUP);
Sdextot = aboutnow/(sizeof(long));
Stblfp = ripopen(filnams[10],&aboutnow,
        &(long huge * )Stbltab,NOPE);
Stbltot = aboutnow/(sizeof(long));
Swxfp = ripopen(filnams[11],&aboutnow,
        &(long huge * )Swxtab,YUP);
Swxtot = aboutnow/(sizeof(museum));
Swdfp                                     =
ripopen(filnams[12],&aboutnow,&Swdtab,YUP);
Swdtot = aboutnow/(sizeof(char));
Twxfp = ripopen(filnams[13],&aboutnow,
        &(long huge * )Twxtab,YUP);
Twxtot = aboutnow/(sizeof(museum));
Twdfp                                     =
ripopen(filnams[14],&aboutnow,&Twdtab,YUP);
Twdtot = aboutnow/(sizeof(char));
if (sscanf(filnams[15],"%d",&Window)==0)
    Window = 5;
if (Window<0 || Window>=MAXLINLN)
    Window = 5;
if (filnum>16)
{
    if ((Ifp=fopen(filnams[16],"r"))==NULL)
    {
        printf("Error opening input file %s",
            filnams[16]);
        queeut(1);
    }
    Inkey = 0L;
    Incon = 0L;
    Intar = 0L;
    Efil = NOPE;

```



```

    Frumafil = YUP;
    /* set up to read input file for rules */
}
else
    Frumafil = NOPE;
if (Tlxtot!=Slxtot)
{
    printf("\n\nFatal: Source and Target have the
");
    printf("different number of lines");
    queueut(1);
}
}

/*****
This routine reads one input files into memory.
*****/

FILE *ripopen(filnam,siz,ptr,readit)
char *filnam;
char huge **ptr;
long *siz;
int readit;
{
    FILE *fp;
    long curptr=0L,ftell();
    if ((fp=fopen(filnam,"rb"))==NULL)
    {
        printf("Error opening input file %s",filnam);
        queueut(1);
    }
    if (fseek(fp,0L,SEEK_END)!=0)
    {
        printf("Error sizing input file %s",filnam);
        queueut(1);
    }
    if ((*siz=ftell(fp))== -1L)
    {
        printf("Error measuring input file %s",filnam);
        queueut(1);
    }
    if (fseek(fp,0L,SEEK_SET)!=0)
    {
        printf("Error      positioning      input      file
%s",filnam);
        queueut(1);
    }
}

```

```

if (readit)
{
    if ((*ptr=(char huge *)
        (long huge *)halloc(*siz+2L,1))==NULL)
    {
        printf("\nError allocating memory for file ");
        printf(" '%s'",filnam);
        queeut(1);
    }
    clearerr(fp);
    for(;!feof(fp) && !ferror(fp);curptr+=512L)
        fread((char *)&(*ptr)[curptr],512,1,fp);
    if (ferror(fp)!=0)
    {
        printf("Error reading input file %s",filnam);
        queeut(1);
    }
}
return(fp);
}

/*****
This routine allocates and initializes arrays.
*****/

makary()
{
    if ((Insent=calloc(NELFUB,sizeof(char)))==NULL)
    {
        printf("\nNot enough room for Input Buffer");
        queeut(1);
    }
    if ((Zazzoo=
        (long *)calloc(MAXLINLN,sizeof(long)))==NULL)
    {
        printf("\nNot enough room for Zazzoo array");
        queeut(1);
    }
    if ((Weistabl=
        (long huge
)halloc(Swxtot,sizeof(long)))==NULL)
    {
        printf("\nNot enough room for trigger array");
        queeut(1);
    }
}

```

```

    if ((Instants=(NINSTANC huge * )
        halloc((long
)MAXNSTNC,sizeof(NINSTANC)))==NULL)
    {
        printf("\nNot enough room for Weiss Link
array");
        queueut(1);
    }
}

/*****
This routine starts reading a file at a given
position.
*****/
/

long tabloid(eon,fp)
long eon;
FILE *fp;
{
    long chime=0L,tabridges();
    if (fseek(fp,eon,SEEK_SET)==0)
        chime = tabridges(fp);
    clearerr(fp);
    return(chime);
}

/*****
This routine continues reading a file at a given
position.
*****/
/

long tabridges(fp)
FILE *fp;
{
    long john;
    fread(&john,4,1,fp);
    if (ferror(fp) || feof(fp))
    {
        clearerr(fp);
        john = 0L;
    }
    return(john);
}

```

```

/*****
This is the standard error exit routine.
*****/
/

queeut(oot)
int oot;
{
    fcloseall();
    exit(oot);
}

/*****
This routine pauses display for user input.
*****/
/

contin()
{
    char dum;
    pos(23,77);
    printf("~");
    pos(23,77);
    printf(" ");
    dum = toupper((getch() & 0xFF));
    if (dum == 'x' || dum == 'X') queeut(0);
}

/*****
This routine clears a line on the screen.
*****/
/

clreol(row,col)
int row,col;
{
    int i;
    if (row < 0 || row > 25 || col < 1 || col > 80)
        return;
    pos(row,col);
    i = 81 - col;
    while(i--)
        printf(" ");
    pos(row,col);
}

```

```

/*****
This routine is the heart of the processing. For
each ambiguous word, instances in the text are
located and correct target translations determined.
Trypt is called to train or test each occurrence.
Success rates are then written to the statistics
file, and contextual rules are written to the rule
file.
*****/
/

lesdoit()
{
    short poof, linlen, sentpos, showiz;
    short showaint, foople, mountain, majesty;
    long timenow, timewas, zoot1, woid1, woid2;
    long kook1, kook2, oozaz, taz, answer, mistakes;
    long zcnt[POLYSEEM], zlinecnt[POLYSEEM];
    cls();
    pos(7,33);
    printf("Weissrul Program");
    Tride = 0L;
    Blode = 0L;
    for(woid1=1L;woid1<Swxtot;woid1++)
        /* for each word in the dictionary */
        {
            for(woid2=1L;woid2<Swxtot;woid2++)
                Weistabl[woid2] = 0L;
            /* clear the rule table */
            zoot1 = 2*POLYSEEM*(woid1-1);
            for(poof=0;poof<POLYSEEM;poof++)
                if ((Zonk[poof]=
                    Ssttab[zoot1+(long )(2*poof)])==0L)
                    break;
            /* put the possible translations in Zonk
*/
            foople = poof;
            /* number of senses */
            if (foople<=1)
                continue;
            if (Showit)
            {
                clreol(12,48);
                pos(12,29);
                printf("Source word %6ld = %s",
                    woid1,&Swdtab[Swxtab[woid1].wdoff]);
            }
        }
    }
}

```

```

else
{
    pos(12,38);
    printf("%6ld",woid1);
}
for(poof=0;poof<foople;poof++)
    zcnt[poof] = 0L;
for(timenow=tabloid(Sdextab[woid1-1L],Stblfp);
    timenow!=0L;timenow=timewas)
    /* look at each line containing the
       ambiguous word */
    {
        kook1 = 1L;
for(timewas=tabridges(Stblfp);timenow==timewas;)
    {
        timenow = timewas;
        timewas = tabridges(Stblfp);
        kook1++;
    }
    if (timenow<Lowlin || timenow>Hilin)
        continue;
        /* only consider lines in the specified
           range */
    for(poof=0;poof<foople;poof++)
        zlincnt[poof] = 0L;
    for(oozzaz=tabloid(Tlxtab[timenow-1L],Tlinfp);
        oozzaz!=0L;
        oozzaz=tabridges(Tlinfp))
    {
        for(poof=0;poof<foople;poof++)
            if (Zonk[poof]==oozzaz)
                zlincnt[poof]++;
    }
        /* count translations in the corresponding
           target line */
    for(poof=0;poof<foople;poof++)
        if (zlincnt[poof]==kook1)
            zcnt[poof] += kook1;
        /* accumulated target sense frequencies */
    }
for(mountain=0;mountain<foople;mountain++)
    /* sort Zonk by frequency */
    {
for(majesty=mountain+1;majesty<foople;majesty++)
    {

```

```

        if (zcnt[majesty]>zcnt[mountain])
        {
            oozaz = Zonk[majesty];
            Zonk[majesty] = Zonk[mountain];
            Zonk[mountain] = oozaz;
            kook1 = zcnt[majesty];
            zcnt[majesty] = zcnt[mountain];
            zcnt[mountain] = kook1;
        }
    }
}
Instop = 0L;
showiz = NOPE;
for(timenow=tabloid(Sdextab[woid1-1L],Stblfp);
    timenow!=0L;timenow=timewas)
    /* go back through source text and
determine
        instances of the ambiguous word and the
        correct target translation of each */
    {
        if (showiz)
            break;
        kook1 = 1L;

for(timewas=tabridges(Stblfp);timenow==timewas;)
    {
        timenow = timewas;
        timewas = tabridges(Stblfp);
        kook1++;
    }
    if (timenow<Lowlin || timenow>Hilin)
        continue;
    for(poof=0;poof<foople;poof++)
        zlincnt[poof] = 0L;
    for(oozaz=tabloid(Tlxtab[timenow-1L],Tlinfp);
        oozaz!=0L;
        oozaz=tabridges(Tlinfp))
    {
        for(poof=0;poof<foople;poof++)
        {
            if (Zonk[poof]==oozaz)
                zlincnt[poof]++;
        }
    }
    mountain = -1;
    for(poof=0;poof<foople;poof++)
    {

```

```

        if (zlincnt[poof]==kook1)
        {
            mountain = poof;
            tzaz = Zonk[poof];
            break;
        }
    }
    /* find a translation that occurred the
same      number of times as the ambiguous word */
    if (mountain==--1)
        continue;
    /* if there is none, skip the source line
*/
    for(poof=mountain+1;poof<foople;poof++)
    {
        if (zlincnt[poof]==kook1)
            zcnt[poof] -= kook1;
    }
    /* alter total frequencies by eliminating
       incorrect translations */
    for(linlen=0,Zazzoo[0]=
        tabloid(Slxtab[timenow-1L],Slinfp);
        Zazzoo[linlen]!=0L && linlen<MAXLNLN; )
    {
        linlen++;
        Zazzoo[linlen] = tabridges(Slinfp);
    }
    /* put the whole source line in memory */
    for(sentpos=0;sentpos<linlen;sentpos++)
    {
        if (Zazzoo[sentpos]==woid1)
        {
            if (Instop<(long )MAXNSTNC)
            {
                Instants[Instop].philpos =
                    Slxtab[timenow-1L];
                Instants[Instop].sentpos = sentpos;
                Instants[Instop++].answer = tzaz;
                /* record instances of the ambiguous
                   word, the file position of the
                   source line, and the correct
                   target translation */
            }
        }
        else
        {
            pos(21,2);
        }
    }

```



```

        printf("Skipping instances of '%s'",
            &Swdtab[Swxtab[woid1-1L].wdoff]);
        showiz = YUP;
        break;
    }
}
}
}
}
Devault = 0L;
if (Frumafil)
    weedit(woid1); /* read initial rules */
if (!Devault)
    Devault = Zonk[0];
/* if input file has not set a
default
        translation, choose the sense that
        is most frequent in the text */
tryit(foople, (Weisteki && !Frumafil), woid1);
/* train and test */
fprintf(Ofp, "\n%s    _    %s",
        &Swdtab[Swxtab[woid1].wdoff],
        &Twdtab[Twxtab[Zonk[0]].wdoff]);
for(kook1=1L; kook1<Swxtot; kook1++)
    /* write it out */
    {
        if ((tzaz=Weistabl[kook1])>0)
            fprintf(Ofp, "\n%s    %s    %s",
                &Swdtab[Swxtab[woid1].wdoff],
                &Swdtab[Swxtab[kook1].wdoff],
                &Twdtab[Twxtab[tzaz].wdoff]);
    }
pos(14,33);
printf("Errors %7ld", Blode);
pos(15,33);
printf("Trials %7ld", Tride);
if (Tride)
    {
        pos(16,33);
        printf("Correct %6f%c",
            (float ) (Tride-Blode)*100.0/(float
)Tride,37);
    }
}
if (Tride)
    fprintf(Nfp, "\n\nCorrect %6f%c",
        (float ) (Tride-Blode)*100.0/(float )Tride,37);
pos(24,1);
}

```

```

/*****
This routine trains and tests the rules. See
section 2.2.7 for the Weiss training method. The
immediate context of each ambiguous word is
examined. If in Weiss mode, the trigger word
nearest the ambiguous word determines the
hypothesized sense. Otherwise, all trigger words in
the context are counted, and the sense with the
highest score wins; if there is a tie, the sense
with the highest frequency among those that tied is
chosen.
*****/
/

tryit(foople,learn,woid)
long woid;
int foople,learn;
{
    int showiz,mistooks,poof,offit,direct;
    int linlen,topsy,toivy,sentpos,woidpos;
    long timenow,answ,answer;
    long oozzaz,zcnt[POLYSEEM],bescnt;
    showiz = YUP;
        /* keep training until no new rules can be
           added */
    while(showiz)
    {
        showiz = NOPE;
        mistooks = 0L;
        for(timenow=0L;timenow<Instop;timenow++)
        {
            if (!Weisteki)
                for(poof=0;poof<foople;poof++)
                    zcnt[poof] = 0L;
                /* initialize trigger count to 0 */
            for(linlen=0,Zazzoo[0]=

tabloid(Instants[timenow].philpos,Slinfp);
        Zazzoo[linlen]!=0L && linlen<MAXLINLN; )
        {
            linlen++;
            Zazzoo[linlen] = tabridges(Slinfp);
        }
        /* read source line into memory */
        sentpos = Instants[timenow].sentpos;

```

```

toivy = (Window)
        ? ((sentpos-Window>=0)
           ? (sentpos-Window)
           :0
        )
        : 0;
topsy = (Window)
        ? ((sentpos+Window<linlen)
           ? (sentpos+Window)
           :linlen-1
        )
        : linlen - 1;
/* determine limits of context */
answer = Instants[timenow].answer;
/* correct target sense */
for(offit=1;offit<MAXLNLN;offit++)
{
    if (sentpos-offit<toivy &&
        sentpos+offit>topsy)
        break;
    /* ignore words outside specified context
*/
    for(direct=-1;direct<2;direct+=2)
        /* look at context words in the order
           ... 9 7 5 3 1 A 2 4 6 8 10 ...
           where A is the ambiguous word,
beginning
           with context word 1 */
        {
            woidpos = sentpos+direct*offit;
            if (woidpos<toivy || woidpos>topsy)
                continue;
            if (oozzaz=Zazzoo[woidpos])
            {
                if ((answ=Weistabl[oozzaz])>0L)
                    /* used in a rule */
                {
                    if (!Weisteki)
                    {
                        for(poof=0;poof<foople;poof++)
                            if (Zonk[poof]==answ)
                                zcnt[poof]++;
                            /* if not in Weiss mode, simply
                               count trigger words */
                    }
                }
            }
        }
    }
}

```



```

        if (zcnt[poof]>bescnt)
        {
            bescnt = zcnt[poof];
            answ = Zonk[poof];
        }
    }
    /* find the sense with the most triggers
*/
    if (answ!=answer)
        mistakes++;
    /* record if an error */
}
else
{
    if (direct==3)
        answ = Devault;
    /* if no triggers were found, use default
*/
    if (direct!=2 && answ!=answer)
        mistakes++;
    /* record an error in Weiss mode */
}
}
}
Tride += Instop;
Blode += mistakes;
if (Instop)
    fprintf(Nfp, "\n%s      %ld/%ld  %c%f",
            &Swdtab[Swxtab[woid].wdoff],
            Instop-mistakes, Instop, 37,
            (float          )(Instop-mistakes)*100.0/(float
)Instop);
}

/*****
This routine reads in the context rule file.
The format is
    A <TAB> B <TAB> C <TAB>
where A is the ambiguous source word,
    B is context source word,
    and C is the target translation that is
    correct for A when near B.
*****/
/

weedit(woid)
long woid;
{

```

```

char *frog;
long gtndx();
if (Inkey==woid)
{
    Weistabl[Incon] = Intar;
    if (Incon==0L)
        Devault = Intar;
}
if (!Efil)
{
    while(fgets(Insent,NELFUB-1,Ifp)!=NULL)
    {
        if (Insent[strlen(Insent)-1]=='\n')
            Insent[strlen(Insent)-1] = '\0';
        if (strlen(Insent)<=0)
            continue;
        frog = strtok(Insent," ");
        if ((Inkey=gtndx(frog,Swxtot,Swxtab,Swdtab))
            ==Swxtot)
        {
            fprintf(Nfp,"\n Unknown key %s",frog);
            Inkey = -1L;
        }
        frog = strtok(NULL," ");
        if ((Incon=gtndx(frog,Swxtot,Swxtab,Swdtab))
            ==Swxtot)
        {
            fprintf(Nfp,
                "\n Unknown context word %s",frog);
            Incon = -1L;
        }
        frog = strtok(NULL," ");
        if ((Intar=gtndx(frog,Twxtot,Twxtab,Twdtab))
            ==Twxtot)
        {
            fprintf(Nfp,
                "\n Unknown translation %s",frog);
            Intar = -1L;
        }
    }
    if (Inkey==woid)
    {
        if (Incon>=0L && Intar>=0L)
        {
            Weistabl[Incon] = Intar;
            if (Incon==0L)
                Devault = Intar;
        }
    }
}

```

```

        else if (Inkey>woid)
            break;
    }
    if (feof(Ifp))
        Efil = YUP;
    else if (ferror(Ifp))
    {
        fprintf(Nfp, "\n\nRead error on input file");
        pos(8,11);
        printf("Read error on input file");
        queeut(1);
    }
}

/*****
Given a string, this routine determines its index in
a given sorted index list.
*****/

long gtndx(symbol, ndxtot, ndxtab, wdtab)
char *symbol, huge *wdtab;
long ndxtot;
museum huge *ndxtab;
{
    int cond;
    long low=0L, high=(ndxtot-1L), mid=0L;
    while (low<=high)
    {
        mid = low + (high-low)/2L;
        if ((cond=
strcmp(symbol, &wdtab[ndxtab[mid].wdoff]))<0)
            high = mid - 1L;
        else if (cond>0)
            low = mid + 1L;
        else
            return(mid);
    }
    return(ndxtot);
}

```

## Appendix C

**A PROGRAM FOR CREATING A SEMANTIC COOCCURRENCE  
NETWORK**

This program trains a neural network based on cooccurrence data. The number of input, hidden, and output units is specified on the command line. Training is based on two files, a pattern file, and a pattern index file.

The pattern file consists of a list of patterns. Each one is composed of two lists. The first list is a list of indexes of source words. The second list is a list of indexes of target words; correct target senses are positive, and incorrect ones are negative. Each list ends with a 0. The pattern index file points to the beginning of a pattern given its pattern number.

The network is trained by setting input units corresponding to the indexes of the source words to have an activation of 1. Errors are generated when the activations of output units corresponding to correct target senses are not equal to 1, and those corresponding to incorrect target senses are not equal



to 0. The algorithm is based on the back propagation program in the PDP software (McClelland 1988).

The network is trained once for each pattern during each epoch. The patterns are permuted before each epoch, so that the training is in random order during each epoch.

```
#include <stdio.h>
#include <dos.h>
#include <string.h>
#include <math.h>

#define YUP 1
#define NOPE 0

#if defined(OS2)
#define INCL_BASE
#include <OS2.H>
#endif

#define rnd() ((float)rand()*3.0518507e-5)

#define WRANGE 0.8
#define MOMENTUM 0.9
#define IHRATE 0.5
#define HORATE .08

char *Winnam,*Wtccnam,*Woutnam;
char *strdup(),*calloc(),*huge *halloc();

FILE *Owfp,*Otssfp,*Pxfp,*Patfp,*Lfp,*Iwfp,*fopen();

int Showit,filinit,shobad;

long Epoch,Nepochs,Pxtot,Pattot;
long john,silver,flipper,Imps,Hided,Oups;

float huge *Ihwts,huge *Howts,huge *Obias,huge
*Odbias,huge *Ihdw,huge *Hodw;
```

```

float *Hacts,*Hbias,*Hdbias,*Hdelta,*Herr,*Hnetin;
float
Ihlrate,Holrate,Obrate,Hbrate,Momentum,Tss,Pss;

long huge *Used;
long huge *Pxtab;
short huge *Pattab;

void srand();
int rand();
long time();

main(argc,argv)
int argc;
char *argv[];
{
#ifdef OS2
    VioSetAnsi(1,0);
#endif

    if (argc<14)
    {
        printf("\nwhinney <dbg><x><owt>");
        printf("<px><pat><#in><#hid><#out><#ep>");
        printf("<ih><ho><hb><ob>(<iwt>)");
        exit(1);
    }

    /* <dbg>  debug level
       <x>    no longer used
       <owt>  output file of connection weights and
              biases
       <px>   pattern index file
       <pat>  pattern file
       <#in>  number of input units
       <#hid> number of hidden units
       <#out> number of output units
       <#ep>  modulo of epochs on which to write
the
              output weights file
units
       <ih>   learning rate for input to hidden
units
       <ho>   learning rate for hidden to output
              units
       <hb>   learning rate for hidden biases
       <ob>   learning rate for output biases
       <iwt>  file name for initial connection
              weights and biases
    */
}

```

```

cls();
pos(1,1);
Showit = 0;
if (sscanf(argv[1], "%d", &Showit) == 0)
{
    printf("\n\nNo debug level in command line: ");
    printf("try again");
    exit(1);
}
if (Showit < 0 || Showit > 5)
    Showit = 0;
Wtccnam = strdup(argv[2]);
Woutnam = strdup(argv[3]);
indat(argc, argv);
    /* read in the pattern files */
makary();
    /* allocate and initialize arrays */
initwits();
    /* initialize weights of connections */
looop();
    /* train the network */
fcloseall();
}

/*****
This routine reads in the pattern and pattern index
files, and the command line parameters.
*****/
/

indat(numfil, filnams)
int numfil;
char *filnams[];
{
    FILE *riropen();
    long aboutnow;
    Pxfp = riropen(filnams[4], &aboutnow,
        &(long huge *)Pxtab);
    Pxtot = aboutnow / (sizeof(long));
    Patfp = riropen(filnams[5], &aboutnow,
        &(short huge *)Pattab);
    Pattot = aboutnow / (sizeof(short));
    Momentum = MOMENTUM;
    if (!sscanf(filnams[6], "%ld", &Imps) || Imps < 1L)
    {
        printf("\nNumber of input units is illegal");
        queueut(1);
    }
}

```

```

}
/* number of input units */
if (!sscanf(filnams[7], "%ld", &Hided) || Hided < 1L)
{
printf("\nHided unit number is illegal");
queueut(1);
}
/* number of hidden units */
if (!sscanf(filnams[8], "%ld", &Oups) || Oups < 1L)
{
printf("\nNumber of output units is illegal");
queueut(1);
}
/* number of output units */
if (!sscanf(filnams[9], "%ld", &Nepochs) ||
Nepochs < 1L)
{
printf("\nNepochs is illegal");
queueut(1);
}
/* modulo of epochs on which to output weights
*/
if (!sscanf(filnams[10], "%f", &Ihlrate) ||
Ihlrate < 0.0)
{
printf("\nIhlrate is illegal");
queueut(1);
}
/* learning rate for input to hidden units */
if (!sscanf(filnams[11], "%f", &Holrate) ||
Holrate < 0.0)
{
printf("\nHolrate is illegal");
queueut(1);
}
/* learning rate for hidden to output units */
if (!sscanf(filnams[12], "%f", &Hbrate) ||
Hbrate < 0.0)
{
printf("\nHbrate is illegal");
queueut(1);
}
/* learning rate for hidden biases */
if (!sscanf(filnams[13], "%f", &Obrate) ||
Obrate < 0.0) {
printf("\nObrate is illegal");
queueut(1);
}
}

```

```

        /* learning rate for output biases */
if (numfil>14)
{
    Winnam = strdup(filnams[14]);
    /* file name for input weights and biases */
    filinit = YUP;
}
else
    filinit = NOPE;
flipper = 100L;
}

/*****
This routine allocates and initializes arrays.
*****/
/

makary()
{
    if (filinit)
        bringiin(Winnam);
    silver = Imps*Hided;
    if ((Ihwts=(float huge *)(long huge *)
        halloc(silver,sizeof(float)))==NULL)
    {
        printf("\n\nOut of memory allocating ");
        printf("IH weights");
        queueut(1);
    }
    /* input to hidden weights */
    if ((Ihdw=(float huge *)(long huge *)
        halloc(silver,sizeof(float)))==NULL)
    {
        printf("\n\nOut of memory allocating ");
        printf("IH delta weights");
        queueut(1);
    }
    /* input to hidden delta terms */
    silver = Oups*Hided;

```

```

if ((Howts=(float huge *)(long huge *)
      halloc(silver,sizeof(float)))==NULL)
{
  printf("\n\nOut of memory allocating ");
  printf("HO weights");
  queueut(1);
}
/* hidden to output weights */
if ((Hodw=(float huge *)(long huge *)
      halloc(silver,sizeof(float)))==NULL)
{
  printf("\n\nOut of memory allocating ");
  printf("HO delta weights");
  queueut(1);
}
/* hidden to output delta terms */
if ((Obias=(float huge *)(long huge *)
      halloc(Oups,sizeof(float)))==NULL)
{
  printf("\n\nOut of memory allocating ");
  printf("output biases");
  queueut(1);
}
/* output biases */
if ((Odbias=(float huge *)(long huge *)
      halloc(Oups,sizeof(float)))==NULL)
{
  printf("\n\nOut of memory allocating ");
  printf("output biases");
  queueut(1);
}
/* output bias delta terms */
if ((Hacts=(float *)(long *))
      calloc((int)Hided,sizeof(float)))==NULL)
{
  printf("\n\nOut of memory allocating ");
  printf("hidden activations");
  queueut(1);
}
/* hidden unit activations */

```

```

if ((Hbias=(float *)(long *))
calloc((int)Hided,sizeof(float))!=NULL)
{
    printf("\n\nOut of memory allocating ");
        hidden biases");
    queeut(1);
}
    /* hidden unit biases */
if ((Hdbias=(float *)(long *))
        calloc((int)Hided,sizeof(float))!=NULL)
{
    printf("\n\nOut of memory allocating ");
    printf("hidden biases");
    queeut(1);
}
    /* hidden bias delta terms */
if ((Hdelta=(float *)(long *))
        calloc((int)Hided,sizeof(float))!=NULL)
{
    printf("\n\nOut of memory allocating ");
    printf("hidden biases");
    queeut(1);
}
    /* hidden activation delta terms */
if ((Herr=(float *)(long *))
        calloc((int)Hided,sizeof(float))!=NULL)
{
    printf("\n\nOut of memory allocating ");
    printf("hidden deltas");
    queeut(1);
}
    /* error terms for hidden units */
if ((Hnetin=(float *)(long *))
        calloc((int)Hided,sizeof(float))!=NULL)
{
    printf("\n\nOut of memory allocating ");
    printf("hidden net Imps");
    queeut(1);
}
    /* net input for hidden units */

```

```

if ((Used=(long huge *)
      halloc(Pxtot,sizeof(long)))==NULL)
{
  printf("\n\nOut of memory allocating ");
  printf("permutation table");
  queueut(1);
}
/* permute array for pattern numbers */
}

/*****
This routine reads a pattern or pattern index file
into memory.
*****/
/

FILE *ripopen(filnam,siz,ptr)
char *filnam;
char huge **ptr;
long *siz;
{
  FILE *fp;
  long curptr=0L,ftell();
  if ((fp=fopen(filnam,"rb"))==NULL)
  {
    printf("Error opening input file %s",filnam);
    queueut(1);
  }
  if (fseek(fp,0L,SEEK_END)!=0)
  {
    printf("Error sizing input file %s",filnam);
    queueut(1);
  }
  if ((*siz=ftell(fp))== -1L)
  {
    printf("Error measuring input file %s",filnam);
    queueut(1);
  }
  if (fseek(fp,0L,SEEK_SET)!=0)
  {
    printf("Error positioning input file %s",
           filnam);
    queueut(1);
  }
}

```



```

if ((*ptr=(char huge *)(long huge * )
    halloc(*siz+2L,1))==NULL)
{
    printf("\nError allocating memory ");
    printf("for file '%s'",filnam);
    queueut(1);
}
clearerr(fp);
for(;!feof(fp) && !ferror(fp);curptr+=512L)
    fread((char * )&(*ptr)[curptr],512,1,fp);
if (ferror(fp)!=0)
{
    printf("Error reading input file %s",filnam);
    queueut(1);
}
return(fp);
}

/*****
This is a standard error exit routine.
*****/
/
queueut(oot)
int oot;
{
    fcloseall();
    exit(oot);
}

/*****
This routine pauses for user input.
*****/
/

contin()
{
    char dum;
    pos(23,77);
    printf("~");
    pos(23,77);
    printf(" ");
    dum = toupper((getch()&0xFF));
    if (dum=='x' || dum=='X') queueut(0);
}

```

```

/*****
This routine clears a line on the screen.
*****/
/

clreol(row,col)
int row,col;
{
    int i;
    if (row<0 || row>25 || col<1 || col>80)
        return;
    pos(row,col);
    i = 81 - col;
    while(i--)
        printf(" ");
    pos(row,col);
}

/*****
This routine initializes weights and biases and
delta terms. Delta terms are initialized to 0.
Either weights and biases are read in from the
weights input file, or biases are set to 0, and
weights are set to random numbers between -0.5 and
0.5.
*****/
/

initwits()
{
    int oogle;
    long nose;
    float alone;
    unsigned seed;
    printf("\nThis program is partly based on ");
    printf("software with the following note");
    printf("\n Copyright 1987 by James L. ");
    printf("McClelland and David E. Rumelhart.");
    cls();
    pos(9,31);
    printf("Initializing Weights");
    if (filinit)
        bringdat(Winnam);
    else
        Epoch = 1;
    srand((unsigned)time(NULL));
    seed = (unsigned)rand();
    srand(seed);
}

```

```

seed = (unsigned)rand();
srand(seed);
silver = Hided*Imps;
for(john=0L; john<silver; john++)
{
    Ihdw[john] = 0.0;
    if (john/1000L*1000L==john)
    {
        pos(11,37);
        printf("%6ld", john);
    }
}
silver = Hided*Oups;
for(john=0L; john<silver; john++)
{
    Hodw[john] = 0.0;
    if (john/1000L*1000L==john)
    {
        pos(11,37);
        printf("%6ld", john);
    }
}
if (!filinit)
{
    silver = Hided*Imps;
    alone = 0.0;
    nose = Hided;
    for(john=0L; john<silver; john++)
    {
        Ihwts[john] = ((float)(WRANGE)) * (rnd()-.5);
        alone += Ihwts[john];
        if (!--nose)
        {
            if (alone>1.0)
            {
                alone = (float )sqrt((double )alone);
                for(nose=john-Hided+1;nose<john;nose++)
                    Ihwts[nose] /= alone;
            }
            nose = Hided;
            alone = 0.0;
        }
        if (john/1000L*1000L==john)
        {
            pos(11,37);
            printf("%6ld", john);
        }
    }
}

```

```

silver = Hided*Oups;
for(john=0L; john<silver; john++)
{
    Howts[john] = (float)(WRANGE) * (rnd()-.5);
    if (john/1000L*1000L==john)
    {
        pos(11,37);
        printf("%6ld", john);
    }
}
silver = Oups;
for(john=0L; john<silver; john++)
{
    Odbias[john] = 0.0;
    if (john/100L*100L==john)
    {
        pos(13,37);
        printf("%6ld", john);
    }
}
if (!filinit)
{
    for(john=0L; john<silver; john++)
    {
        Obias[john] = 0.0;
        if (john/1000L*1000L==john)
        {
            pos(13,37);
            printf("%6ld", john);
        }
    }
}
for(oogle=0; oogle<(int)Hided; oogle++)
{
    Hdbias[oogle] = 0.0;
    pos(15,37);
    printf("%6ld", john);
}
if (!filinit)
{
    for(oogle=0; oogle<(int)Hided; oogle++)
    {
        Hbias[oogle] = 0.0;
        pos(15,37);
        printf("%6ld", john);
    }
}

```

```

}

/*****
This routine randomly permutes the order in which
patterns will be used in the next epoch to train the
network.
*****/

/

permute()
{
    long feller,yeller,beller;
    for(feller=0L;feller<Pxtot;feller++)
        Used[feller] = feller;
    for(feller=0L;feller<Pxtot;feller++)
    {
        yeller = (long)(rnd() * (Pxtot-feller) +
feller);
        beller = Used[feller];
        Used[feller] = Used[yeller];
        Used[yeller] = beller;
    }
}

/*****
This routine computes the activation of a unit,
based on the net input to that unit.
*****/

/

float logistic (x)
/* copied from PDP bp.c */
float x;
{
    double exp ();
    if (x > 11.5129)
        return(.99999);
    else
        if (x < -11.5129)
            return(.00001);
    else
        return(1.0 / (1.0 +
(float) exp( (double) ((-1.0) * x))));
}

/*****

```

This is the main training routine. During each epoch, use each pattern to update weights and biases.

```

*****
/

looop()
{
    float impetus,acticus,logistic();
    float oerr,delta,dwight,scalar,gutter;
    long curin,curout,curpat,cur,row,goseek;
    long absout,windex,gindex;
    cls();
    pos(6,31);
    printf("Quiet! I'm thinking!");
    shobad = NOPE;
    row = Hided;
    Tss = 0.0;
    if (Showit>2 && (Lfp=fopen("hors.lst","w"))
        ==NULL)
    {
        printf("Error opening output file hors.lst");
        queueut(1);
    }
    for(;;Epoch++)
    {
        pos(12,37);
        printf("%6ld",Epoch);
        Tss = 0.0;
        permute();
        /* randomly order patterns */
        for(john=0L;john<Pxtot;john++)
        {
            /* for each pattern */
            Pss = 0.0;
            memset(Hacts,0,((int)Hided)*sizeof(float));
            memset(Herr,0,((int)Hided)*sizeof(float));
            memset(Hdelta,0,((int)Hided)*sizeof(float));
            /* initialize hidden activations, errors,
               and delta terms */

            memcpy(Hnetin,Hbias,((int)Hided)*sizeof(float));
            /* the hidden biases */
            cur = curpat = Pxtab[Used[john]]/2;
            /* current pattern number */

```

```

for(curin=(long)Pattab[cur++];
    curin!=0L;curin=(long)Pattab[cur++])
{
    if (curin>Imps || curin<=0L)
        continue;
    gindex = (curin-1L)*row;
    for(goseek=0L;goseek<row;goseek++)
        Hnetin[goseek] += Ihwts[gindex++];
}
    /* calculate inputs to hidden units
*/
for(goseek=0L;goseek<row;goseek++)
    Hacts[goseek] = logistic(Hnetin[goseek]);
    /* calculate hidden activations */
for(curout=(long)Pattab[cur++];
    curout!=0L;curout=(long)Pattab[cur++])
{
    absout = (curout>0L) ? curout : -curout;
    absout -= 1L;
    if (absout>=Oups)
        continue;
    impetus = Obias[absout];
    /* initialize output unit net input
to
        current value of output bias */
    gindex = windex = absout*row;
    for(goseek=0;goseek<row;goseek++)
        impetus      +=      Howts[gindex++]      *
Hacts[goseek];          /* accumulate input to
output unit */
    acticus = logistic(impetus);
    /* output unit activation */
    oerr = ((curout>0L) ? 1.0 : 0.0) - acticus;
    /* output unit error */
    if (Showit>2 && (oerr>0.5 || oerr<-0.5))
        fprintf(Lfp,"%4ld %ld      %9.8g\n",
            Used[john],curout,oerr);
    Pss += oerr*oerr;
    /* add squared error to partial sum
of
        squares */
    delta = oerr*acticus*(1.0-acticus);
    /* output delta */
    dwight = Odbias[absout] =
        Obrate*delta+Momentum*Odbias[absout];
    /* delta weight */
    Obias[absout] += dwight;
    /* change output bias */

```

```
gindex = windex;
```



```

for(goseek=0L;goseek<row;goseek++)
{
    Herr[goseek] += delta*Howts[gindex];
    dwight = Hodw[gindex] =
        Holrate*delta*Hacts[goseek]+
        Momentum*Hodw[gindex];
    Howts[gindex++] += dwight;
    /* change hidden to output weights */
}
}
for(goseek=0L;goseek<row;goseek++)
{
    Hdelta[goseek] = Herr[goseek]*Hacts[goseek]*
        (1.0-Hacts[goseek]);
    dwight =Hdbias[goseek] =
        Hbrate*Hdelta[goseek]+
        Momentum*Hdbias[goseek];
    Hbias[goseek] += dwight;
}
    /* change hidden biases */
cur = curpat;
for(curin=(long)Pattab[cur++];
    curin!=0L;curin=(long)Pattab[cur++])
{
    if (curin>Imps || curin<=0L)
        continue;
    gindex = (curin-1L)*row;
    gutter = 0.0;
    for(goseek=0L;goseek<row;goseek++)
    {
        dwight = Ihdw[gindex] =
            Ihlrate*Hdelta[goseek]+
            Momentum*Ihdw[gindex];
        Ihwts[gindex] += dwight;
        gutter += Ihwts[gindex]*Ihwts[gindex];
        gindex++;
    }
    /* change input to hidden weights */
}

```

```

if (gutter>1.0)
{
  gutter = (float )sqrt((double )gutter);
  gindex = (curin-1L)*row;
  for(goseek=0L;goseek<row;goseek++)
    Ihwts[goseek] /= gutter;
}
/* normalize connecting weights from input
to hidden units so that the weights
leading out of each input unit form a
vector of length 1.0: note that this is
different from the PDP software */
}
Tss += Pss;
/* update total sum of squares */
if (john/flipper*flipper==john)
{
  pos(15,36);
  printf("%4ld %4ld", john+1L,Used[john]+1L);
  pos(18,36);
  printf("%9.8g      ",Tss);
  if (Showit==2)
    contin();
}
}
if (shobad)
{
  pos(17,31);
  printf("                ");
}
pos(19,36);
printf("%9.8g      ",Tss);
if (!shobad && Tss<(float)Pxtot)
  shobad = YUP;
if ((Epoch-((Epoch/Nepochs)*Nepochs))==0L)
  poodiout();
if (Showit>2)
  break;
}
}
}

```

```

/*****
This routine writes weights and biases to an output
file so that training can continue at a later time
using this file as input.
*****/
/

poodiout()
{
    if ((Owfp=fopen(Woutnam,"wb"))==NULL)
    {
        printf("Error opening output file %s",Woutnam);
        queueut(1);
    }
    fwrite(&Epoch,sizeof(long),1,Owfp);
    fwrite(&Tss,sizeof(float),1,Owfp);
    fwrite(&Imps,sizeof(long),1,Owfp);
    fwrite(&Hided,sizeof(long),1,Owfp);
    fwrite(&Oups,sizeof(long),1,Owfp);
    silver = Hided*Imps;
    for(john=0L;john<silver;john++)
        fwrite(&Ihwts[john],sizeof(float),1,Owfp);
    silver = Hided*Oups;
    for(john=0L;john<silver;john++)
        fwrite(&Howts[john],sizeof(float),1,Owfp);
    silver = Hided;
    for(john=0L;john<silver;john++)
        fwrite(&Hbias[john],sizeof(float),1,Owfp);
    silver = Oups;
    for(john=0L;john<silver;john++)
        fwrite(&Obias[john],sizeof(float),1,Owfp);
    fclose(Owfp);
    pos(22,37);
    printf("(%6ld)",Epoch);
}

/*****
This routine reads network configuration and epoch
data from a weights input file.
*****/
/

bringiin(phil)
char *phil;
{
    if ((Iwfp=fopen(phil,"rb"))==NULL)
    {
        printf("Error opening input file %s",phil);
    }
}

```

```

    queueut(1);
}
fread(&Epoch,sizeof(long),1,Iwfp);
if (ferror(Iwfp)!=0)
{
    printf("Error reading input file %s",phil);
    queueut(1);
}
fread(&Tss,sizeof(float),1,Iwfp);
if (ferror(Iwfp)!=0)
{
    printf("Error reading input file %s",phil);
    queueut(1);
}
pos(19,28);
printf("Epoch %ld  Tss %9.8g",Epoch,Tss);
Epoch++;
fread(&Imps,sizeof(long),1,Iwfp);
if (ferror(Iwfp)!=0 || Imps<=0L)
{
    printf("Error reading number of input ");
    printf("units from input file %s",phil);
    queueut(1);
}
fread(&Hided,sizeof(long),1,Iwfp);
if (ferror(Iwfp)!=0 || Hided<=0L)
{
    printf("Error reading number of hidden ");
    printf("units from input file %s",phil);
    queueut(1);
}
fread(&Oups,sizeof(long),1,Iwfp);
if (ferror(Iwfp)!=0 || Oups<=0L)
{
    printf("Error reading number of output units ");
    printf("from input file %s",phil);
    queueut(1);
}
}
}

```

```

/*****
This routine reads weights and biases from an input
file.
*****/
/

bringdat(phil)
char *phil;
{
    silver = Hided*Imps;
    for(john=0L; john<silver; john++)
    {
        fread(&Ihwts[john], sizeof(float), 1, Iwfp);
        if (john/1000L*1000L==john)
        {
            pos(11, 37);
            printf("%6ld", john);
        }
        if (ferror(Iwfp)!=0)
        {
            printf("Error reading input file %s", phil);
            queeut(1);
        }
    }
    silver = Hided*Oups;
    for(john=0L; john<silver; john++)
    {
        fread(&Howts[john], sizeof(float), 1, Iwfp);
        if (john/1000L*1000L==john)
        {
            pos(11, 37);
            printf("%6ld", john);
        }
        if (ferror(Iwfp)!=0)
        {
            printf("Error reading input file %s", phil);
            queeut(1);
        }
    }
    silver = Hided;
    for(john=0L; john<silver; john++)
    {
        fread(&Hbias[john], sizeof(float), 1, Iwfp);
        pos(15, 37);
        printf("%6ld", john);
        if (ferror(Iwfp)!=0)
        {
            printf("Error reading input file %s", phil);

```

```
        queeut(1);
    }
}
silver = Oups;
for(john=0L; john<silver; john++)
{
    fread(&Obias[john], sizeof(float), 1, Iwfp);
    if (john/100L*100L==john)
    {
        pos(13, 37);
        printf("%6ld", john);
    }
    if (ferror(Iwfp)!=0)
    {
        printf("Error reading input file %s", phil);
        queeut(1);
    }
}
fclose(Iwfp);
Iwfp = NULL;
}
```

## Appendix D

**A PROGRAM FOR EVALUATING SEMANTIC COOCCURRENCE  
NETWORKS**

This program evaluates the ambiguity resolution power of a trained neural network. It takes as input an ASCII file in groups of three or four lines. Each group is as follows:

```
S
C1 C2 C3 C4 C5 C6 ...
T1 T2 ...
TC
```

where *S* is the ambiguous source word, *C<sub>i</sub>* are the source context words, *T<sub>i</sub>* are the possible target translations, and *TC* is the correct target translation. Normally, the ambiguous word should not be one of the words listed as context words. In test mode, the *TC* line is required, but if the program is run in resolution mode, the *TC* line is not allowed and the network simply resolves the ambiguity without checking against any predetermined standard.

Inputs to the program are the file just discussed, the weights file created by the program in Appendix C, and the word index files described in Appendix B.

The program can be run with two algorithms. The first, called default mode, presents the entire context to the network as inputs at once. The second, called pair mode, presents the ambiguous word and one context word to the network, and counts the context word as a trigger word if the activation of the most activated target word output unit exceeds a certain threshold (a command line parameter); each context word is presented in this manner, the target word counts are then tallied, and the most triggered sense is chosen. If none of the context words are known, the algorithm picks T1 as the correct default translation.

```
#include <stdio.h>
#include <dos.h>
#include <string.h>

#define YUP 1
#define NOPE 0

#define TIMIL 500
/* number of context words allowed */
#define TNETXE 20
```



```

        /* number of target translations allowed */
#define NELFUB 2000
        /* input buffer length */
#define NELMANF 80
        /* screen input buffer length */

#if defined(OS2)
#define INCL_BASE
#include <OS2.H>
#endif

char *Wonnam,*Winnam,*Woutnam,*Waitnam;
char *strdup(),*calloc(),huge *halloc();

FILE
*Foup,*Iwfp,*Swxfp,*Swdfp,*Twxfp,*Twdfp,*fopen();

float Tss,Accrate,huge *Ihwts,huge *Howts,huge
*Obias;

int Showit;
        /* screen debug level */
int Filit;
        /* file information level */
int Checkit;
        /* true if checking against known value */
int Defaultit;
        /* true if presenting whole context at once */
int Quickit;
        /* true if doing only one command line file */

long Swxtot,Twxtot,Swdtot,Twdtot;

long Epoch,Nepochs,John,silver,Imps,Hided,Oups;

float huge *Ihwts,huge *Howts,huge *Obias;
float *Hacts,*Hbias,*Hnetin;

typedef struct wax {
    long freq;
    long wdoff;
} museum;

museum huge *Swxtab,huge *Twxtab;
char huge *Swdtab,huge *Twdtab;

main(argc,argv)
int argc;

```

```

char *argv[];
{
#ifdef OS2
    VioSetAnsi(1,0);
#endif

    if (argc<7)
    {
        printf("\nwindext <dbug><wts><swx><swd>");
        printf("<twx><twd><rate>( <infil>");
        exit(1);
    }
    cls();
    pos(1,1);
    Showit = 0;
    if (sscanf(argv[1], "%d", &Showit) == 0)
    {
        printf("\n\nNo debug level in command line: ");
        printf("try again");
        exit(1);
    }
    /* Debug level WXYZ means
        W      1 if all context words are to be
              presented to the network at
once
              0 if context words should be
              presented one-by-one with the
              ambiguous word and tallied
        X      1 if input files contain the TC
              line to check results against
              0 if no checking is to be done
        Y      file level information
              0 writes word, context, sense
              chosen and statistics if
Checkit
              1 0-level info plus trigger
words
              2 1-level info plus activations
              3 2-level info plus tallies
              4 3-level info plus hidden units
              9 0-level without contexts
        Z      screen debug level
    */
    Defaultit = Showit/1000;
    if (Defaultit != 1)
        Defaultit = 0;
    Showit %= 1000;
    Checkit = Showit/100;

```

```

if (Checkit!=1)
    Checkit = 0;
Showit %= 100;
Filit = Showit/10;
Quickit = (Filit==9) ? YUP : NOPE;
if (Filit<0 || Filit>5)
    Filit = 0;
Showit %= 10;
if (Showit<0 || Showit>5)
    Showit = 0;
Winnam = NULL;
indat(argc,argv);
    /* read in indexes */
bringiin(Waitnam);
    /* read in weight file info */
makary();
    /* allocate arrays */
initwits();
    /* read in weights */
ovrnover();
    /* evaluate the neural network */
fcloseall();
}

/*****
This routine reads in the index files and command
line parameters.
*****/
/

indat(numfil,filnams)
int numfil;
char *filnams[];
{
    FILE *ripopen();
    long aboutnow;
    Waitnam = strdup(filnams[2]);
    Swxftp = ripopen(filnams[3],&aboutnow,
        &(char huge *)Swxtab);
    Swxtot = aboutnow/(sizeof(museum));
    Swdfp = ripopen(filnams[4],&aboutnow,&Swdtab);
    Swdtot = aboutnow/(sizeof(char));
    Twxftp = ripopen(filnams[5],&aboutnow,
        &(char huge *)Twxtab);
    Twxtot = aboutnow/(sizeof(museum));
    Twdfp = ripopen(filnams[6],&aboutnow,&Twdtab);
    Twdtot = aboutnow/(sizeof(char));
    if (numfil>7)

```

```

    {
        if (sscanf(filnams[7], "%f", &Accrate) == 0)
        {
            printf("\n\nNo debug level in command line:
");
            printf("try again");
            exit(1);
        }
    }
else
    Accrate = 0.8;
    /* trigger threshold in pair mode */
if (numfil > 8)
    Wonnam = filnams[8];
else
    Wonnam = NULL;
    /* ASCII input data for resolution */
}

/*****
This routine allocates the arrays.
*****/

makary()
{
    silver = Imps * Hided;
    if ((Ihwts = (float huge *) (long huge *)
        halloc(silver, sizeof(float))) == NULL)
    {
        printf("\n\nOut of memory allocating IH
weights");
        queueut(1);
    }
    silver = Oups * Hided;
    if ((Howts = (float huge *) (long huge *)
        halloc(silver, sizeof(float))) == NULL)
    {
        printf("\n\nOut of memory allocating HO
weights");
        queueut(1);
    }
}

```

```

if ((Obias=(float huge *)(long huge *)
      halloc(Oups,sizeof(float)))==NULL)
{
    printf("\n\nOut of memory allocating ");
    printf("output biases");
    queeut(1);
}
if ((Hacts=(float *)(long *)

calloc((int)Hided,sizeof(float))==NULL)
{
    printf("\n\nOut of memory allocating ");
    printf("hidden activations");
    queeut(1);
}
if ((Hbias=(float *)(long *)

calloc((int)Hided,sizeof(float))==NULL)
{
    printf("\n\nOut of memory allocating ");
    printf("hidden biases");
    queeut(1);
}
if ((Hnetin=(float *)(long *)

calloc((int)Hided,sizeof(float))==NULL)
{
    printf("\n\nOut of memory allocating hidden ");
    printf("net inputs");
    queeut(1);
}
}

/*****
This routine reads an input file into memory.
*****/

FILE *ripopen(filnam,siz,ptr)
char *filnam;
char huge **ptr;
long *siz;
{
    FILE *fp;
    long curptr=0L,ftell();
    if ((fp=fopen(filnam,"rb"))==NULL)
    {
        printf("Error opening input file %s",filnam);
    }
}

```

```

    queueut(1);
}
if (fseek(fp,0L,SEEK_END)!=0)
{
    printf("Error sizing input file %s",filnam);
    queueut(1);
}
if ((*siz=ftell(fp))== -1L)
{
    printf("Error measuring input file %s",filnam);
    queueut(1);
}
if (fseek(fp,0L,SEEK_SET)!=0)
{
    printf("Error      positioning      input      file
%s",filnam);
    queueut(1);
}
if ((*ptr=(char huge * )(long huge * )
    halloc(*siz+2L,1))==NULL)
{
    printf("\nError allocating memory for file
'%s' ",
        filnam);
    queueut(1);
}
clearerr(fp);
for( ;!feof(fp) && !ferror(fp); curptr+=512L)
    fread((char * )&(*ptr)[curptr],512,1,fp);
if (ferror(fp)!=0)
{
    printf("Error reading input file %s",filnam);
    queueut(1);
}
return(fp);
}

/*****
This is the standard error exit routine.
*****/

/

queueut(oot)
int oot;
{
    fcloseall();
    exit(oot);
}

```



```

/*****
This routine pauses for user input.
*****/
/

contin()
{
    char dum;
    pos(23,77);
    printf("~");
    pos(23,77);
    printf(" ");
    dum = toupper((getch()&0xFF));
    if (dum=='x' || dum=='X') queeut(0);
}

/*****
This routine clears a line on the screen.
*****/
/

clreol(row,col)
int row,col;
{
    int i;
    if (row<0 || row>25 || col<1 || col>80)
        return;
    pos(row,col);
    i = 81 - col;
    while(i--)
        printf(" ");
    pos(row,col);
}

/*****
This routine initializes weights and biases in the
neural network based on the weights file.
*****/
/

initwits()
{
    int oogle;
    unsigned seed;
    printf("\nThis program is partly based on ");
    printf("software with the following note");
    printf("\n Copyright 1987 by James L. ");
    printf("McClelland and David E. Rumelhart.");
}

```



```

    cls();
    pos(9,31);
    printf("Initializing Weights");
    bringdat(Waitnam);
}

/*****
This routine calculates activations based on net
inputs.
*****/

float logistic (x)
    /* copied from PDP bp.c */
float x;
{
    double exp ();
    if (x > 11.5129)
        return(.99999);
    else
        if (x < -11.5129)
            return(.00001);
    else
        return(1.0 / (1.0 + (float) exp( (double)
((-1.0) * x))));
}

/*****
This is the main routine for evaluation. In a loop,
it asks for a filename; it expects the ASCII input
data to be in filename.won in Checkit mode, and
filename.win otherwise; output is to filename.wot.
It reads each group, finds the indices of the
ambiguous word, source context words, target
translations, and correct translation. It calls
figrdout to choose a sense, and reports the results.
*****/

ovrnover()
{
    FILE *finp;
    char philenam[NELMANF],philein[NELMANF];
    char phileout[NELMANF],**trw;
    char ambwoid[NELMANF],chkwoid[NELMANF];
    char curwrđ[NELMANF],*insent,*frog;
    long errnum,totnum,*srccon,*targposs,gtndx();
    short *targscr,fndout;

```

```

unsigned short srcnum, trgnum, bestran, figrdout();
if ((srccon=
    (long *)calloc(TIMIL, sizeof(long)))==NULL)
{
    printf("\n\nOut of memory on allocation ");
    printf("of source array!");
    queueut(1);
}
/* source context array */
if ((targposs=(long *)
    calloc(TNETXE, sizeof(long)))==NULL)
{
    printf("\n\nOut of memory on allocation ");
    printf("of target array!");
    queueut(1);
}
/* target possible translations array */
if ((trw=(char **)
    calloc(TNETXE, sizeof(char *)))==NULL)
{
    printf("\n\nTRW is too big!");
    queueut(1);
}
/* array pointing to target strings */
if ((targscr=(short *)
    calloc(TNETXE, sizeof(short)))==NULL)
{
    printf("\n\nOut of memory on allocation ");
    printf("of target score array!");
    queueut(1);
}
/* array for tallies */
if ((insent=calloc(NELFUB, sizeof(char )))==NULL)
{
    printf("\n\nPassed out making insents!");
    queueut(1);
}
/* input buffer from ASCII file */
Winnam = &philein[0];
Woutnam = &phileout[0];
while(1)
{
    if (!Wonnam)
    {
        cls();
        pos(1,31);
        printf("THE CONTEXT MACHINE");
        pos(3,27);
    }
}

```

```

printf("File of contexts:  ");
philenam[0] = 0;
if (scanf("%s",philenam)<=0 ||
    philenam[0]=='x' ||
    philenam[0]=='X')
    break;
strcpy(Winnam,philenam);
strcpy(Woutnam,philenam);
}
else
{
    strcpy(Winnam,Wonnam);
    strcpy(Woutnam,Wonnam);
}
if (Checkit)
    strcat(Winnam, ".won");
else
    strcat(Winnam, ".win");
strcat(Woutnam, ".wot");
if ((Foup=fopen(Woutnam, "w"))==NULL)
{
    pos(8,11);
    printf("\n\nError opening output file %s",
        Woutnam);
    contin();
    continue;
}
if ((finp=fopen(Winnam, "r"))==NULL)
{
    pos(8,11);
    printf("Error opening output file %s",Winnam);
    contin();
    fclose(Foup);
    continue;
}
fprintf(Foup, "Processing contextual patterns ");
fprintf(Foup, "using file %s, Epoch %ld, rate
%f.",
    Waitnam, Epoch-1L, Accrate);
if (Default)
    fprintf(Foup, "\nDefault processing");
fprintf(Foup, "\n\n=====");
fprintf(Foup,
    "=====\n");
fprintf(Foup, "\nContext analysis of patterns ");
fprintf(Foup, "from file %s.",Winnam);
errnum = 0L;
totnum = 0L;

```

```

while(1)
{
    fprintf(Foup, "\n\n-----");
    fprintf(Foup,
        "-----");
    if (Showit>1)
    {
        cls();
        pos(1,31);
        printf("THE CONTEXT MACHINE");
        pos(3,40-strlen(philein)/2);
        printf("%s",philein);
    }
    if (fgets(ambwoid,NELMANF-1,finp)==NULL)
        /* read ambiguous word */
    {
        if (feof(finp))
            break;
        if (ferror(finp))
        {
            fprintf(Foup, "\n\nRead error on input ");
            fprintf(Foup, "file %s",Winnam);
            if (Showit)
            {
                pos(8,11);
                printf("Read error on input ");
                printf("file %s",Winnam);
                if (Showit>1)
                {
                    contin();
                    clreol(8,11);
                }
            }
            break;
        }
    }
    if (ambwoid[strlen(ambwoid)-1]=='\n')
        ambwoid[strlen(ambwoid)-1] = '\0';
    srcnum = 0;
    if ((srccon[srcnum++]=
        gtndx(ambwoid,Swxtot,Swxtab,Swdtab))
        ==Swxtot)
    {
        fprintf(Foup, "\n\n UNKNOWN keyword ");
        fprintf(Foup, '%s', ambwoid);
        if (Showit)
        {
            pos(5,31-strlen(ambwoid)/2);

```

```

        printf("UNKNOWN                                keyword
'%s' ",&ambwoid[0]);
        if (Showit>3)
            contin();
        clreol(7,1);
    }
    if (fgets(insent,NELFUB-1,finp)==NULL)
    {
        if (feof(finp))
            break;
        if (ferror(finp))
        {
            fprintf(Foup,"\n\nRead error on input
");
            fprintf(Foup,"file %s",Winnam);
            if (Showit)
            {
                pos(8,11);
                printf("Read error on input ");
                printf("file %s",Winnam);
                if (Showit>1)
                {
                    contin();
                    clreol(8,11);
                }
            }
            break;
        }
    }
    if (fgets(insent,NELFUB-1,finp)==NULL)
    {
        if (feof(finp))
            break;
        if (ferror(finp))
        {
            fprintf(Foup,"\n\nRead error on input
");
            fprintf(Foup,"file %s",Winnam);
            if (Showit)
            {
                pos(8,11);
                printf("Read error on input ");
                printf("file %s",Winnam);

```

```

        if (Showit>1)
        {
            contin();
            clreol(8,11);
        }
    }
    break;
}
}
continue;
}
else
{
    fprintf(Foup, "\n\nKeyword '%s'", ambwoid);
    if (Showit)
    {
        pos(5, 35-strlen(ambwoid)/2);
        printf("Keyword '%s'", ambwoid);
    }
}
if (fgets(insent, NELFUB-1, finp)==NULL)
    /* read context line */
{
    if (feof(finp))
    {
        fprintf(Foup, "\n\nUnexpected end of ");
        fprintf(Foup, "input file %s", Winnam);
        if (Showit)
        {
            pos(8,11);
            printf("Unexpected end of input ");
            printf("file %s", Winnam);
            if (Showit>1)
            {
                contin();
                clreol(8,11);
            }
        }
    }
    break;
}
if (ferror(finp))
{
    fprintf(Foup, "\n\nRead error on input ");
    fprintf(Foup, "file %s", Winnam);
}

```

```

    if (Showit)
    {
        pos(8,11);
        printf("Read error on input ");
        printf("file %s",Winnam);
        if (Showit>1)
        {
            contin();
            clreol(8,1);
        }
    }
    break;
}
}
if (insert[strlen(insert)-1]!='\n')
    insert[strlen(insert)-1] = '\0';
if (!Quickit)
    fprintf(Foup, "\nContext: '%s'",insert);
if (Showit>1)
{
    pos(9,1);
    printf("Context: '%s'",insert);
}
frog = strtok(insert, " ");
fndout = NOPE;
while(frog!=NULL)
{
    if ((srcon[srcnum]=
        gtndx(frog,Swxtot,Swxtab,Swdtab))
        ==Swxtot)
    {
        if (!fndout)
        {
            if (!Quickit)
            {
                fprintf(Foup, "\n UNKNOWN words: ");
                fprintf("%s",frog);
            }
            fndout = YUP;
        }
    }
    else if (!Quickit)
        fprintf(Foup, " %s",frog);
    if (Showit)
    {
        pos(8,30-strlen(frog)/2);
        printf("UNKNOWN source word %s",frog);
        if (Showit>3)
            contin();
    }
}

```

```

        clreol(8,1);
    }
    frog = strtok(NULL," ");
    continue;
}
if (++srcnum>=TIMIL)
{
    fprintf(Foup,"Out of space in context ");
    fprintf(Foup," array; ignoring remainder
");
    fprintf(Foup,"of context");
    if (Showit)
    {
        pos(8,11);
        printf("Out of space in context
array;");
        printf(" ignoring remainder of
context");
        if (Showit>1)
        {
            contin();
            clreol(8,1);
        }
    }
    break;
}
frog = strtok(NULL," ");
}
if (fgets(insent,NELFUB-1,finp)==NULL)
/* read target translation line */
{
    if (feof(finp))
    {
        fprintf(Foup,"\nUnexpected end of input
");
        fprintf(Foup,"file %s",Winnam);
        if (Showit)
        {
            pos(8,11);
            printf("Unexpected end of input ");
            printf("file %s",Winnam);
            if (Showit>1)
                contin();
        }
        break;
    }
}

```



```

if (ferror(finp))
{
    fprintf(Foup, "\nRead error on input file
");
    fprintf(Foup, " %s", Winnam);
    if (Showit)
    {
        printf("\n\nRead error on input ");
        printf("file %s", Winnam);
        if (Showit>1)
            contin();
    }
    break;
}
}
if (insert[strlen(insert)-1]!='\n')
    insert[strlen(insert)-1] = '\0';
trgnum = 0;
fprintf(Foup, "\nSenses: '%s'", insert);
if (Showit>1)
{
    pos(12,1);
    printf("Senses: '%s'", insert);
}
fndout = NOPE;
frog = strtok(insert, " ");
while(frog!=NULL)
{
    if ((targposs[trgnum]=
        gtndx(frog, Twxtot, Twxtab, Twdtab))
        ==Twxtot)
    {
        if (!fndout)
        {
            fprintf(Foup, "\n UNKNOWN words: ");
            fprintf(Foup, " %s", frog);
            fndout = YUP;
        }
        else
            fprintf(Foup, " %s", frog);
        if (Showit)
        {
            pos(8,30-strlen(frog)/2);
            printf("UNKNOWN target word %s", frog);
            if (Showit>3)
                contin();
        }
    }
}

```

```

        clreol(8,1);
    }
    frog = strtok(NULL," ");
    continue;
}
trw[trgnum] = frog;
if (++trgnum>=TNETXE)
{
    fprintf(Foup,"Out of space in sense ");
    fprintf(Foup,"array; ignoring remainder
");
    fprintf(Foup,"of context");
    if (Showit)
    {
        pos(8,11);
        printf("Out of space in sense ");
        printf("array; ignoring remainder ");
        printf("of context");
        if (Showit>1)
        {
            contin();
            clreol(8,1);
        }
    }
    break;
}
frog = strtok(NULL," ");
}
if (trgnum<=0)
{
    fprintf(Foup,"\nNo targets were specified");
    if (Showit>1)
    {
        pos(7,28);
        printf("No targets were specified");
        contin();
    }
    continue;
}
if (Checkit)
{
    if (fgets(chkwoid,NELMANF-1,finp)==NULL)
        /* read correct translation line */
    {
        if (feof(finp))
        {
            fprintf(Foup,"\n\nUnexpected end of ");

```

```

        fprintf(Foup,"file          on          input
file",Winnam);
        if (Showit)
        {
            pos(8,11);
            printf("Unexpected end of file ");
            printf("on input file %s",Winnam);
            if (Showit>1)
            {
                contin();
                clreol(8,11);
            }
        }
        break;
    }
    if (ferror(finp))
    {
        fprintf(Foup,"\n\nRead  error  on  input
");
        fprintf(Foup,"file %s",Winnam);
        if (Showit)
        {
            pos(8,11);
            printf("Read error on input ");
            printf("file %s",Winnam);
            if (Showit>1)
            {
                contin();
                clreol(8,11);
            }
        }
        break;
    }
}
if (chkwoid[strlen(chkwoid)-1]=='\n')
    chkwoid[strlen(chkwoid)-1] = '\0';
}
bestran = figrdout(srccon,srcnum,
targposs,trgnum,trw,targscr);
/* figure out best translation */
if (bestran==trgnum)
    bestran = 0;
/* if none chosen, use first translation
on the translation line */
frog = trw[bestran];
fprintf(Foup,
"\n\nBest translation is '%s'",frog);

```

```

if (Checkit)
{
    totnum++;
    if (chkwoid && strlen(chkwoid)>0 &&
        strcmp(trw[besttran],chkwoid)==0)
        fprintf(Foup," (correct)");
    else
    {
        fprintf(Foup," { # '%s' }",chkwoid);
        errnum++;
    }
}
if (Showit>1)
{
    pos(7,30-strlen(frog)/2);
    printf("Best translation is '%s'",frog);
    contin();
}
}
fclose(finp);
fprintf(Foup,"\n");
if (Checkit && totnum)
{
    fprintf(Foup,"\n%d/%d = %f%c correct\n\n",
        totnum-errnum,totnum,
        ((float )(((float )(totnum-errnum)))/
            ((float )totnum))*100.0,37);
}
fclose(Foup);
if (Wonnam)
    break;
}
}

```

```

/*****
This is the routine that uses the neural network to
determine the correct translation.  In default mode,
it simply calls wipadout.  In pair mode, the
ambiguous word and one context word are presented as
input to the network.  If the activation of the most
activated target translation exceeds the threshold,
the context word is counted as a trigger word and is
tallied.
*****/
/

unsigned short figrdout(srcary,srcnum,trgary,
                        trgnum,wort,tally)
unsigned short srcnum,trgnum;
long *srcary,*trgary;
char **wort;
short *tally;
{
    short nap,very,relaxing,too;
    float axe;
    unsigned short grin,wipadout();
    long twoary[2];
    if (!Defaultit)
    {
        if (srcnum>1)
        {
            too = 2;
            for(nap=0;nap<trgnum;nap++)
                tally[nap] = 0;
            /* initialize tallies */
            twoary[0] = srcary[0];
            if (Showit==1)
                pos(12,1);
            for(nap=1;nap<srcnum;nap++)
            {
                /* use each context word in turn */
                twoary[1] = srcary[nap];
                if (Filit>3)
                    fprintf(Foup,"\n\n>> %s %s",
                        &Swdtab[Swxtab[twoary[0]].wdoff],
                        &Swdtab[Swxtab[twoary[1]].wdoff]);
            }
        }
    }
}

```

```

if (Showit>1)
{
    pos(11,1);
    printf("< %s %s > ",
        &Swdtab[Swxtab[twoary[0]].wdoff],
        &Swdtab[Swxtab[twoary[1]].wdoff]);
}
grin = wipadout(twoary,too,
                trgary,trgnum,wort,&axe);
/* determine most activated translation */
if (grin!=trgnum && axe>=Accrate)
/* if it exceeds threshold, tally it */
{
    tally[grin]++;
    if (Showit==1)
        printf("\n '%s' triggers '%s' (%f)",
            &Swdtab[Swxtab[twoary[1]].wdoff],
            wort[grin],axe);
    if (Filit)
    {
        fprintf(Foup,"\n '%s' triggers '%s'",
            &Swdtab[Swxtab[twoary[1]].wdoff],
            wort[grin]);
        if (Filit>1)
            fprintf(Foup," (%f)",axe);
    }
}
}
}
very = NOPE;
relaxing = 0;
if (Filit>2)
    fprintf(Foup,"\n\nTallies:");
for(nap=0;nap<trgnum;nap++)
{
    if (Filit>2)
        fprintf(Foup,"\n %s = %hd",
            wort[nap],tally[nap]);
    if (tally[nap]>tally[relaxing])
    {
        relaxing = nap;
        very = NOPE;
    }
    else if (nap!=0 &&
            tally[nap]==tally[relaxing])
        very = YUP;
}
if (!very && tally[relaxing]!=0)

```

```

        return(relaxing);
    }
    fprintf(Foup, "\n\nDefault Used");
}
return(wipadout(srcary,srcnum,
                trgary,trgnum,wort,&axe));
}

/*****
This routine uses the neural network to determine
the most activated translation.
*****/

unsigned short wipadout(srcary,srcnum,
                       trgary,trgnum,wort,indyact)
unsigned short srcnum,trgnum;
long *srcary,*trgary;
char **wort;
float *indyact;
{
    float impetus,acticus,logistic(),nummal;
    short rowum,colum;
    unsigned short beswoid,cur;
    long curin,curout,row,goseek,absout,gindex;
    row = Hided;
    memset(Hacts,0,((int)Hided)*sizeof(float));
    memcpy(Hnetin,Hbias,((int)Hided)*sizeof(float));
    for(cur=0;cur<srcnum;cur++)
    {
        curin = srcary[cur];
        if (curin>Imps || curin<=0L)
            continue;
        gindex = (curin-1)*row;
        for(goseek=0L;goseek<row;goseek++)
            Hnetin[goseek] += Ihwts[gindex++];
        /* calculate inputs to hidden units */
    }
    if (Filit>3)
        fprintf(Foup, "\n");
    if (Showit>1)
        pos(24,40-Hided/2);
    for(goseek=0L;goseek<row;goseek++)
    {
        Hacts[goseek] = logistic(Hnetin[goseek]);
        /* and hidden activations */
        if (Filit>3)
            fprintf(Foup, "%c",

```

```

        (Hacts[goseek]>0.5) ? '1' : '0');
    if (Showit>1)
        printf("%c", (Hacts[goseek]>0.5) ? '1' : '0');
}
beswoid = trgnum;
nummal = -1.0;
for(cur=0;cur<trgnum;cur++)
{
    absout = trgary[cur]-1L;
    if (absout>=Oups || absout<0L)
        continue;
    impetus = Obias[absout];
    gindex = absout*row;
    for(goseek=0;goseek<row;goseek++)
        impetus += Howts[gindex++] * Hacts[goseek];
    /* inputs to output unit */
    acticus = logistic(impetus);
    /* output unit activation */
    if (acticus>nummal)
    {
        beswoid = cur;
        nummal = acticus;
    }
    /* find the most activated translation */
    if (Filit>3)
        fprintf(Foup, "\n%9.8g
        %s", acticus, wort[cur]);
    if (Showit>1)
    {
        rowum = 13+cur-(cur/7)*7;
        colum = 1+(cur/7)*26;
        pos(rowum, colum);
        printf("%9.8g", acticus);
        pos(rowum, colum+12);
        printf("%s", wort[cur]);
    }
}
*indyact = nummal;
return(beswoid);
}

```



```

/*****
This routine reads network information.
*****/
/

bringiin(phil)
char *phil;
{
    if ((Iwfp=fopen(phil,"rb"))==NULL)
    {
        printf("Error opening input file %s",phil);
        queeut(1);
    }
    fread(&Epoch,sizeof(long),1,Iwfp);
    if (ferror(Iwfp)!=0)
    {
        printf("Error reading input file %s",phil);
        queeut(1);
    }
    fread(&Tss,sizeof(float),1,Iwfp);
    if (ferror(Iwfp)!=0)
    {
        printf("Error reading input file %s",phil);
        queeut(1);
    }
    pos(19,28);
    printf("Epoch %ld  Tss %9.8g",Epoch,Tss);
    Epoch++;
    fread(&Imps,sizeof(long),1,Iwfp);
    if (ferror(Iwfp)!=0 || Imps<=0L)
    {
        printf("Error reading number of input units ");
        printf("from input file %s",phil);
        queeut(1);
    }
    fread(&Hided,sizeof(long),1,Iwfp);
    if (ferror(Iwfp)!=0 || Hided<=0L)
    {
        printf("Error reading number of hidden units ");
        printf("from input file %s",phil);
        queeut(1);
    }
    fread(&Oups,sizeof(long),1,Iwfp);
    if (ferror(Iwfp)!=0 || Oups<=0L)
    {
        printf("Error reading number of output units ");
        printf("from input file %s",phil);
        queeut(1);
    }
}

```

```

    }
}

/*****
This routine reads weights and biases.
*****/

bringdat(phil)
char *phil;
{
    silver = Hided*Imps;
    for(john=0L; john<silver; john++)
    {
        fread(&Ihwts[john], sizeof(float), 1, Iwfp);
        if (john/1000L*1000L==john)
        {
            pos(11, 37);
            printf("%6ld", john);
        }
        if (ferror(Iwfp)!=0)
        {
            printf("Error reading input file %s", phil);
            queeut(1);
        }
    }
    silver = Hided*Oups;
    for(john=0L; john<silver; john++)
    {
        fread(&Howts[john], sizeof(float), 1, Iwfp);
        if (john/1000L*1000L==john)
        {
            pos(11, 37);
            printf("%6ld", john);
        }
        if (ferror(Iwfp)!=0)
        {
            printf("Error reading input file %s", phil);
            queeut(1);
        }
    }
    silver = Hided;
    for(john=0L; john<silver; john++)
    {
        fread(&Hbias[john], sizeof(float), 1, Iwfp);
        pos(15, 37);
        printf("%6ld", john);
        if (ferror(Iwfp)!=0)

```

```

    {
        printf("Error reading input file %s",phil);
        queeut(1);
    }
}
silver = Oups;
for(john=0L; john<silver; john++)
{
    fread(&Obias[john], sizeof(float), 1, Iwfp);
    if (john/100L*100L==john)
    {
        pos(13,37);
        printf("%6ld", john);
    }
    if (ferror(Iwfp)!=0)
    {
        printf("Error reading input file %s",phil);
        queeut(1);
    }
}
fclose(Iwfp);
Iwfp = NULL;
}

/*****
Given a string, this routine returns an index into a
sorted list.
*****/

long gtndx(symbol, ndxtot, ndxtab, wdtab)
char *symbol, huge *wdtab;
long ndxtot;
museum huge *ndxtab;
{
    int cond;
    long low=0L, high=(ndxtot-1L), mid=0L;
    while (low<=high)
    {
        mid = low + (high-low)/2L;
        if ((cond=strcmp(symbol,
            &wdtab[ndxtab[mid].wdoff]))<0)
            high = mid - 1L;
        else if (cond>0)
            low = mid + 1L;
        else
            return(mid);
    }
}

```

```
    return(ndxtot);  
}
```

**BIBLIOGRAPHY**

Adriens, Geert, and Steven L. Small. 1988. "Word Expert Parsing in a Cognitive Science Perspective," Lexical Ambiguity Resolution, eds. Steven L. Small, Garrison W. Cottrell, and Michael K. Tannenhaus. San Mateo, California: Morgan Kaufman.

Brady, John W. 1990. "ICSS (Interlingual Communication Support Systems) and a Wittgensteinian Language Game," Proceedings of the 13th Annual European Studies Conference (1988). In Press.

Charniak, Eugene. 1983. "Passing Markers: A Theory of Contextual Influence in Language Comprehension," Technical Report CS-80, Department of Computer Science, Brown University.

Church, Kenneth, and William Gale, Patrick Hanks,  
and

Donald Hindle. 1989. "Parsing, Word  
Associations, and Typical Predicate-Argument  
Relations," Parsing Technologies Conference  
Proceedings, Carnegie Mellon University.

Cottrell, Garrison. 1988. "A Model of Lexical  
Access

of Ambiguous Words," Lexical Ambiguity  
Resolution, eds. Steven L. Small, Garrison W.  
Cottrell, and Michael K. Tannenhaus. San  
Mateo, California: Morgan Kaufman.

The Daily Herald, Provo, Utah. October 20, 1988.

Dreyfus, Hubert L. 1985. "From Micro-Worlds to  
Knowledge Representation: AI at an Impasse," in  
Readings in Knowledge Representation, eds.  
Ronald J. Brachman, Hector Levesque. Los Altos,  
California: Morgan Kaufman.

Fass, Dan. 1988. "Collative Semantics: A Semantics for Natural Language Processing," Report MCCS-88-118, Computing Research Laboratory, New Mexico State University.

Golden Common LISP Reference Manual, Version 1.01.  
1983. Gold Hill Computers, Cambridge, Massachusetts, pp. 20-21.

Gould, Roderick. 1957. "Multiple Correspondence," Mechanical Translation. Cambridge, Massachusetts: MIT Press. 4:14-27.

Hirst, Graeme. 1987. Semantic Interpretation and the Resolution of Ambiguity. Cambridge: Cambridge University Press.

Kaplan, Abraham. 1955. "An Experimental Study of Ambiguity and Context," Mechanical Translation. Cambridge, Massachusetts: MIT Press. 2:39-47.

Kawamoto, Alan H. 1988. "Distributed Representations

of Ambiguous Words and Their Resolution in a Connectionist Network," Lexical Ambiguity Resolution, eds. Steven L. Small, Garrison W. Cottrell, and Michael K. Tannenhaus. San Mateo, California: Morgan Kaufman.

King, Gilbert W. 1956. "Stochastic Methods of Mechanical Translation," Mechanical Translation. Cambridge, Massachusetts: MIT Press. 3:38-39.

Liberman, Mark. 1989. "How Many Words Do People Know?" unpublished talk given at the 27th Annual Meeting of the Association for Computational Linguistics.

Madhu, Swaminathan, and Dean W. Lytle. 1965. "A Figure of Merit Technique for the Resolution of Non-Grammatical Ambiguity," Mechanical Translation. Cambridge, Massachusetts: MIT Press. 8:9-13.



Masterman, Margaret. 1957. "The Thesaurus in Syntax

and Semantics," *Mechanical Translation*.  
Cambridge, Massachusetts: MIT Press. 4:35-43.

McClelland, James L. and David E. Rumelhart. 1988.

Explorations in Parallel Distributed Processing. 1988. Cambridge, Massachusetts: MIT Press.

Papegaaïj, B. C. 1986. Word Expert Semantics, an

Interlingual Knowledge-Based Approach, V. Sadler and A. P. M. Witkam, editors. Dordrecht, Holland: Foris Publications.

Perry, James W. 1955. "Translation of Russian

Technical Literature by Machine," *Mechanical Translation*. Cambridge, Massachusetts: MIT Press. 2:15-26.

Pimsleur, Paul. 1957. "Semantic Frequency Counts,"

*Mechanical Translation*. Cambridge, Massachusetts: MIT Press. 4:11-13.

Piron, Claude. 1988. "Learning from Translation Mistakes", Conference Proceedings of New Directions in Machine Translation. Dordrecht, Holland: Foris Publishers.

Sedelow, Sally Yeates and Walter A. Sedelow, Jr. 1986. "Thesaural Knowledge Representation," Proceedings, Advances in Lexicology, Second Annual Conference of the UW Centre for the New Oxford English Dictionary. Waterloo, Canada: University of Waterloo.

Sedelow, Sally Yeates and Walter A. Sedelow, Jr. 1990. "Conceptual Primitives," ms. University of Arkansas at Little Rock.

Slator, Brian M. 1988. "Lexical Semantics and a Preference Semantics Parser," Report MCCS-88-116, Computing Research Laboratory, New Mexico State University.

Smith, Joseph. 1978. "Joseph Smith History",  
The Pearl of Great Price. Salt Lake City,  
 Utah: The Church of Jesus Christ of Latter-day  
 Saints.

Sparck-Jones, Karen. 1965. "Experiments in  
 Semantic  
 Classification," Mechanical Translation.  
 Cambridge, Massachusetts: MIT Press. 8:97-  
 112.

Tarlburt, John R. and Donna K. Mooney. 1990. ms.  
 University of Arkansas at Little Rock.

Walker, Donald E. and Robert A. Amsler. 1986. "The  
 Use of Machine-Readable Dictionaries in  
 Sublanguage Analysis," Sublanguage: Description  
 and Processing, eds. R. Grishman and R.  
 Kittredge. Lawrence Erlbaum.

Weaver, Warren. 1967. "Selected Papers of Warren  
 Weaver", in Science and Imagination. New York:  
 Basic Books, Inc.

Weiss, Stephen. 1973. "Learning to Disambiguate,"  
Storage Information and Retrieval. Pergamon  
Press. 9:33-41.

Wilks, Yorick. 1968. "On-Line Semantic Analysis of  
English Texts," Mechanical Translation.  
Cambridge, Massachusetts: MIT Press. 11:59-  
72.

Wilks, Yorick. 1975. "Preference Semantics,"  
Formal  
Semantics of Natural Language, ed. Edward L.  
Keenan. Cambridge: Cambridge University Press.

Wilks, Yorick, and Dan Fass, Cheng-Ming Guo, James  
E.

McDonald, Tony Plate, and Brian M. Slator.  
1987. "A Tractable Machine Dictionary as a  
Resource for Computational Semantics," Report  
MCCS-87-105, Computing Research Laboratory, New  
Mexico State University.

**VITA**

Dan Walter Higinbotham was born May 10, 1954, in Denver, Colorado, the son of Oliver A. Higinbotham, Jr., and Birdie Jane Epstein Higinbotham. He received the degrees of Bachelor of Science and Master of Science in Mathematics at Brigham Young University in 1978 and 1980, with the Master's thesis Binary Structures on a Finite String. As a software engineer for Weidner Communications, Inc., he directed a team producing software to translate Japanese to English, before entering The University of Texas in 1981. Following classwork in 1984, he became Director of Research at Executive Communication Systems, Inc., in Provo, Utah, where he designed and implemented a machine translation system based on Lexical Functional Grammar. On December 1, 1989, he married Barbara Ann Morrell, and lived happily ever after.

Permanent address: 756 N. 400 E.

Orem, Utah

84057

This dissertation was typed by the author.