Case-Based Problem Solving with a Large Knowledge Base of Learned Cases [1]

Wendy G. Lehnert
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Abstract

Recent experiments indicate that a case-based approach to the problem of word pronunciation is effective as the basis for a system that learns to pronounce English words. More generally, the approach taken here illustrates how a case-based reasoner can access a large knowledge base containing hundreds of potentially relevant cases and consolidate these multiple knowledge sources using numerical relaxation over a structured network. In response to a test item, a search space is first generated and structured as a lateral inhibition network. Then a spreading activation algorithm is applied to this search space using activation levels derived from the case base. In this paper we describe the general design of our model and report preliminary test results based on a training vocabulary of 750 words. Our approach combines traditional heuristic methods for memory organization with connectionist-inspired techniques for network manipulation in an effort to exploit the best of both information-processing methodologies.

I. Introduction

While many researchers have proposed various reasoning mechanisms for case-based systems [Bain, 1986, Kolodner, Simpson, and Sycara-Cyranski, 1985, Hammond, 1986, Rissland and Ashley, 1986, Lebowitz, 1986], we have very little experience with truly large memories. A realistic memory for any case-based reasoner must contain hundreds or thousands of cases. With this many available cases we can expect to see significant competition among a large number of potentially relevant cases at any given time. It is therefore important to develop techniques for memory access and conflict resolution which will enable us to effectively arbitrate large numbers of contributing cases. It is all too easy to design ad hoc heuristics that apply to a limited set of examples operating in conjunction with a hand-coded memory. We have developed a more general strategy for handling case-based memories with a special concern for the difficulties of scaling up.

The problem of word pronunciation is an ideal task for experiments in case-based learning and reasoning. Symbolic pronunciations available in a standard dictionary grant us easy access to a large corpus of data which is completely free of the representational problems encountered in tasks like legal reasoning or medical diagnosis. By circumventing these otherwise important issues, we have been able to concentrate our efforts on the automatic construction of a large case-based memory, indexing techniques for that memory, and strategies for resolving competition among multiple cases.

To illustrate the effectiveness of our ideas we have implemented PRO, a system that learns to pronounce words using a knowledge base organized around cases. PRO creates its knowledge structures in response to supervised training items where each item contains a word and a sequence of phonemes representing that word's correct pronunciation. At any time we can interrupt PRO's learning mode to test PRO on arbitrary vocabulary items. We currently have an on-line data base of 850 word/pronunciation pairs which can be used as training items, test items, or both.

II. Automated Construction of the Case Base: Learning Mode

When PRO examines its training items it does not create one case per training item. Rather, PRO segments an input word into a partition of substrings which maps onto the targeted phoneme string in a credible fashion. We will refer to each mapping from a substring to a phoneme as a "hypothesis." A case is then a fixed-length subsequence of hypotheses contained in the mapping of a segmentation to a phoneme sequence.

To illustrate, suppose we have associated the segmentation (SH OW T I ME) with the phoneme sequence (*sh* $\bar{o}$ *t* $\bar{i}$ *m*). This produces five hypotheses (SH/*sh*, OW/$\bar{o}$, T/*t*, I/$\bar{i}$, and ME/*m*). In principle, we could design our cases to record sequences of either arbitrary or fixed lengths (although sequences of length 1 would fail to encode any con-

[1]This research was supported by NSF Presidential Young Investigators Award NSFIST-8351863 and DARPA grant N00014-85-K-0017.

textual information). The selection of a case length is a design decision that benefits from some experimentation: we have found that 3 is an effective length for the word pronunciation task. So the item "showtime" is then associated with the following seven cases:

1.	(START**/Ø	START*/Ø	SH/*sh*)
2.	(START*/Ø	SH/*sh*	OW/$\bar{o}$)
3.	(SH/*sh*	OW/$\bar{o}$	T/*t*)
4.	(OW/$\bar{o}$	T/*t*	I/$\bar{i}$)
5.	(T/*t*	I/$\bar{i}$	ME/*m*)
6.	(I/$\bar{i}$	ME/*m*	END*/Ø)
7.	(ME/*m*	END*/Ø	END**/Ø)

The null hypotheses serve only to mark places at the beginning and end of each word. These place markers are necessary if we want to maintain sequences of uniform length.

The cases PRO identifies correspond to a moving "window" on the lexical item, although we cannot say that the window has a fixed length in terms of the letters spanned. Since each hypothesis may contain 1, 2, 3, or even 4 letters (as in OUGH/$\bar{o}$) from the input word, the size of this window can vary from 1 letter (at the beginning or end of the word) to as many as 12 letters (at least in principle) depending on the hypotheses involved.

Each case identified during training is indexed under the substring of its leading hypothesis and stored within a tree structure.[2] The indexing substring points to a separate tree for each hypothesis associated with that substring. For example, the substring "ow" could have two hypotheses associated with it: OW/$\bar{o}$ (as in "show") and OW/*ou* (as in "how"). To store the sequence (OW/$\bar{o}$ T/*t* I/$\bar{i}$) we would add this case to the tree headed by OW/$\bar{o}$. If T/*t* has never followed OW/$\bar{o}$ before, we must create a new branch for the tree. Otherwise, if T/*t* had been encountered after OW/$\bar{o}$ during training, we would traverse the branch already constructed and next check to see if I/$\bar{i}$ is present in the tree at the next level. Since we are operating PRO with a fixed sequence length of 3, each of our trees is limited to a depth of three hypotheses.

PRO updates its knowledge base by expanding tree structures as needed, and updating frequency data for each case encountered during training. Each node of a case tree is associated with a positive integer which indicates how many times this particular node has been visited during training. If T/*t* has followed OW/$\bar{o}$ 13 times before, we will now update that count to 14. If I/$\bar{i}$ has never followed (OW/$\bar{o}$ T/*t*) before, we create a new node for the tree and initialize its frequency count at 1. It follows that frequency counts can never increase as we traverse branches out from a root node: frequencies typically diminish as we move downward through a tree.

In general, there is more than one way to segment a character string and match those segments against the phoneme sequence encoding the string's pronunciation. PRO must therefore access its knowledge base during training in order to identify preferred segmentations with high degrees of credibility. Since segmentation errors during training encourage additional segmentation errors during subsequent training, as well as impaired performance in test mode, PRO is very conservative about segmentation judgments. If PRO cannot identify a preferred segmentation, PRO will ignore the training item and make no attempts to modify its knowledge base in response to that item. This strategy of "timid acquisition" is the only way to guarantee effective learning in the absence of negative examples [Berwick, 1986].

We have found that a very effective heuristic for filtering multiple segmentations can be devised by maximizing (1) known hypotheses, (2) new hypotheses which partially match some known hypothesis, and (3) known hypotheses with high frequency counts. However, these filters can still fail if PRO is subjected to an "unreasonable" training session at the start. To get PRO off on the right foot, we must begin with an initial training session that makes it easy for PRO to identify valid hypotheses.

At the beginning, when PRO's knowledge base is sparse, it is important to train PRO with training pairs that do not result in multiple segmentations. In general, three-letter words satisfy this constraint because most three-letter words map to a sequence of three phonemes. Once PRO has built a knowledge base in response to some such initial training session, we can move on to four and five-letter training words with confidence. Apart from this general restriction, PRO is not overly sensitive to the design of its training sequences. At worst, PRO will not learn anything from a poorly placed training item if it has not "worked itself up" to that item adequately.

III. Utilizing The Case Base: Test Mode

When PRO is in test mode it receives a lexical item and attempts to produce a unique phoneme sequence in response to that item. Because PRO's knowledge base does not remember training items in their entirety, there is no guarantee that PRO will produce a correct pronunciation for a word it previously encountered during training. However, PRO does tend to have a somewhat higher hit rate for items seen in training compared to novel test words. We will discuss PRO's performance in test mode at the end of this paper.

[2] In fact, we also index each case under the last hypothesis as well as the first. Indices using trailing hypotheses access trees that traverse case sequences backwards whereas cases indexed by leading hypotheses go forwards. This dual encoding becomes important when we use frequency data during test mode.

PRO begins its analysis of a test word by producing a search space of all possible hypothesis sequences it can associate with that word. Note that this search space will not, in general, contain all possible segmentations of the input word since most segmentations will not be associated with hypotheses recognized by the knowledge base. For example, any segmentation of "showtime" containing the substring "wti" will be rejected since PRO will not have any hypotheses in memory using the string "wti." The complexity of our search space is therefore limited by the knowledge base available to PRO. PRO also limits its search space by eliminating any segmentations which place hypothesis boundaries between letters that have never been divided between two hypotheses during training (PRO creates a small data base of this information during training in addition to the case-based memory described above). For the task of word pronunciation, the greatest sources of ambiguity result from vowels and vowel combinations, so those are places where the search space tends to "fan out."

The search space of possible word pronunciations which PRO generates must now be resolved to a single preferred pronunciation. This selection process is made on the basis of information available in PRO's case base. To access the case base, we transform our search space into a structured network utilizing lateral inhibition and spreading activation.

To begin, we create inhibitory links between any pair of hypotheses that share overlapping substrings from the input word. All such hypotheses are in competition with one another and must resolve to a preferred winner. These inhibitory links provide negative activation throughout the network, which is essential to the process of identifying a preferred path through the net. Figure 1 shows a sample search space for the word "showtime". (In actuality, this search space would be much larger if PRO had any substantial training behind it). All positive activation for the network comes from the case base by adding additional "context-nodes" to the network. A context node is added to the network wherever three consecutive hypotheses correspond to a complete branch of some tree in PRO's knowledge base. The context node is then connected to the three hypotheses that spawned it, and initialized at a positive level of activation. This level of activation is computed on the basis of frequency data available in the case trees.[3] Once all possible context nodes have been generated, connected up, and initialized, we are ready to relax the network.

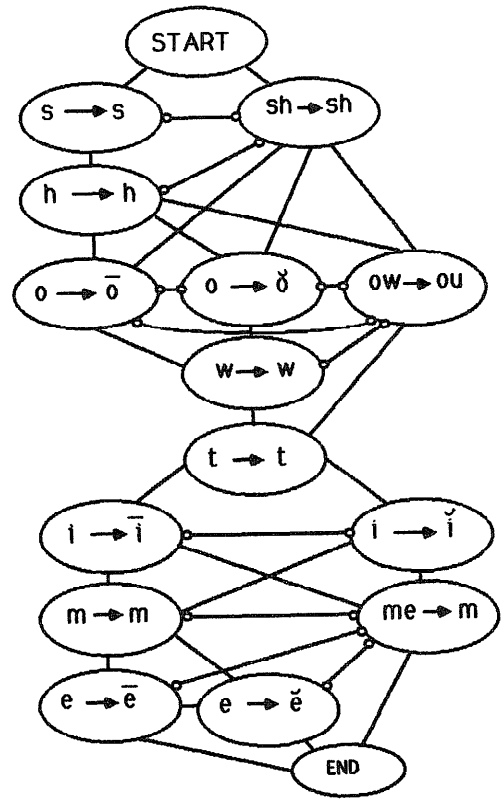

Figure 1: A search space network

A standard relaxation algorithm (see Feldman and Ballard 1982) is then applied to our network representation until all activation levels have stabilized or the number of iterations reaches 30. In general, the network stabilizes before 20 iterations, and we then evaluate the activation levels associated with each candidate pronunciation in the network. A path in the network receives as its activation level the lowest activation level found over all the hypothesis nodes contained in that path. Happily, most paths zero-out, leaving only a few with positive levels of activation. Of those with any positive activation, there is usually one with a maximal activation level. In the case of a unique maximal path, we have a strong preference for a single word pronunciation. In the case of multiple maximal paths, PRO picks one arbitrarily and returns that as the preferred pronunciation.

Once a test item has been resolved, PRO discards the network representation constructed for that item and moves on to the next test item with a clean slate. No modifications to PRO's knowledge base are made during test mode.

[3]The precise value is $f(x,y) = \text{rnd}\ (10^{(1-(1-x)(1-y))})$ where x and y represent the frequency of this particular hypothesis sequence relative to the first hypothesis in the sequence and the last hypothesis in the sequence. In other words, x tells us how often this sequence follows an instance of the first hypothesis and y tells us how often this sequence precedes an instance of the third hypothesis. $x,y \in (0,1]$, $f(x,y) \in (1,10]$ and $f(x,y) \longrightarrow 10$ as either x or $y \longrightarrow 1$.

IV. Test Results

At this time we have assembled a corpus of 850 training items consisting of words chosen at random from a dictionary and ranging in length from 3 to 8 letters. We have collected data for PRO's performance on a test set consisting of the first 200 words from a 750-word training session. This test set of "familiar" words contains 50 three-letter words, 100 four-letter words, and 50 five-letter words. We have also tested 100 novel words not found in the 750-word training session (50 four-letter words and 50 five-letter words). We ran PRO on both test sets at three different times during training: (1) after 250 words, (2) after 450 words, and (3) after all 750 words.

It is important to note that the complexity of a given test item changes as PRO processes additional training items and increases its knowledge base. There are three factors that contribute to the complexity of PRO's task during test mode: (1) the number of hypotheses in memory, (2) the number of cases (hypothesis sequences) in memory, and (3) the frequency data associated with each case in memory. We will refer to these three factors as the "Hypothesis Base," the "Case Base," and the "Statistical Base."

As the Hypothesis Base grows, we will see the search space for a given test item increase since additional segmentations may be possible and more hypotheses may be associated with each plausible segmentation. As the Case Base grows, we see more context nodes generated in response to a given test item since there are more hypothesis sequences available to reinforce the search space. As the Statistical Base grows, we do not see additional complexity in the structures we generate, but we may see some effects on the time required to stabilize the network during the relaxation process. On the basis of only 750 training items, it is not possible to say whether effects from a growing Statistical Base will alter the number of iterations required during network relaxation.

We can easily plot growth curves for the Hypothesis Base as a function of training items processed. Table 1 shows how the number of hypotheses increases during training. Note that a sharp growth rate during the first 200 items (630 phonemes) drops off to a much slower growth rate during the remaining 550 items. During the initial growth spurt we average about 1 new hypothesis for every 2 training items. After the initial growth spurt, we pick up roughly 1 new hypothesis for every 10 training items. By the end of this training session PRO has identified 180 hypotheses. While the growth rate during the last 550 items appears to be linear, we must assume that it will become increasingly harder to identify new hypotheses as more training items are processed. Since we have not trained enough to see our curve level off, it is difficult to say where the ceiling on hypotheses might be. However, it is safe to say that this growth process will eventually reach an asymptote, at which point the Hypothesis Base cannot increase the complexity of our search spaces any further.

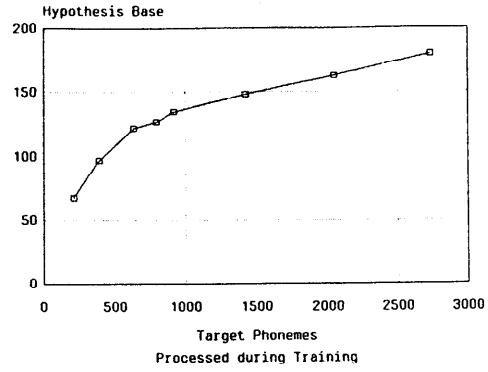

Table 1. Hypothesis Base Growth Curve

A similar growth curve for the Case Base tells a different story. Table 2 shows how the number of cases in PRO's memory increase as PRO processes the same 750 training items. Now we see a largely linear function throughout: PRO generates roughly 2.1 new cases for each training item it processes. By the end of this training session PRO has generated a knowledge base containing 1591 cases.

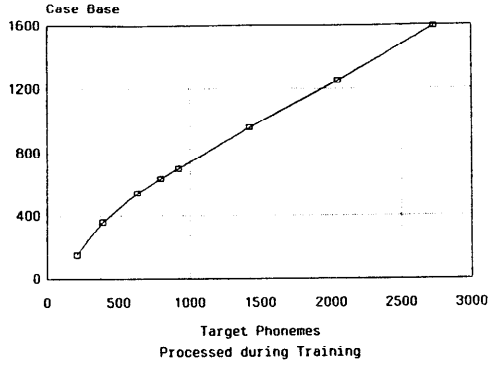

Table 2. Case Base Growth Curve

As with the Hypothesis Base, we must assume that the Case Base will eventually saturate and cease to acquire new cases. Unfortunately, our limited training experience does not allow us to speculate about how many training items might be required before saturation sets in. At the very least, we can say that saturation in the Hypothesis Base necessarily precedes saturation in the Case Base, and it will probably take the Case Base a while to settle down after the Hypothesis Base has stabilized.

Unlike the Hypothesis Base and the Case Base, the Statistical Base will continue to change as long as training continues. If training items are repeated, we would not see any operational differences in the Statistical Base, as long as all training items are repeated with the same frequency. However, interesting effects would be derived from the Statistical Base if some segment of the training corpus were repeated heavily in the absence of compensating

repetitions throughout the entire training corpus. Words encountered often would then influence the relaxation process more heavily than words seen less frequently. This is an important feature of our model as far as psychological validity is concerned. Since the frequency distribution for words in everyday use is not uniform, one could argue that not all words are equal in the mind of a lexicon-processing human. Any psychological account of phonetic interpretation should therefore be responsive to this question of frequency distributions and predict behaviors that vary in response to manipulations of word distributions.

Given the increased complexity of PRO's pronunciation task as its knowledge base grows, it is not surprising to see some degradation in PRO's test mode performance as we move through the training corpus. After training on 250 words, PRO returns a hit rate of 94.7% on the number of phonemes it can correctly identify in the test corpus of familiar words. By the time PRO has processed 750 training words, this hit rate has dropped to 89.9% — the error rate has doubled. At the same time, the Hypothesis Base has grown by a factor of 1.4 and the Case Base has expanded by a factor of 2.5. It is interesting to note that PRO's performance is not significantly correlated with word lengths: shorter words do not necessarily fare better than longer words.

While PRO's performance drops slightly over time for familiar words, we see an increase in performance levels for novel words. After 250 words PRO correctly identifies 66% of the phonemes for our test group of 100 words not present in the training corpus. By the time PRO has processed 750 training words, this success rate has risen to 75%. It is not surprising to see PRO behave differently for these two groups of test items in its early stages of training. Further experimentation is needed to determine whether these two performance curves eventually converge and stabilize.

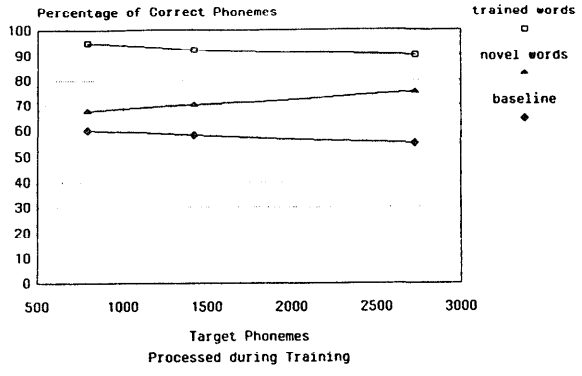

Table 3. Comparative Hit Rates in Test Mode

Table 3 shows the two performance curves for familiar and novel words along with a baseline curve designed to factor out the effects of the Case Base. The percentages contained in the baseline result from test runs that substitute random guesses in place of PRO's relaxation algorithm. For these baseline hit rates we generate search spaces derived from the Hypothesis Base just as PRO does. But instead of adding context nodes and relaxing the network, we simply pick a path at random from the search space. After 250 words, the random algorithm exhibits a hit rate of 60%. After 750 words, this hit rate has dropped to 55% due to the larger search spaces that result from a growing Hypothesis Base.

V. Conclusions

The network representation generated by PRO during test mode provides a simple formalism for defining a search space of possible responses to a given test item. This search space is further influenced by the addition of context nodes derived from PRO's Case Base. The relaxation algorithm applied to this network representation provides us with a powerful strategy for integrating the relative strengths of relevant cases. A large number of competing cases and mutually-supportive cases influence the contributions of one another and eventually stabilize in a global consensus.

When evaluating PRO, we must remember that the techniques used here will be most effective in a domain characterized by a tendency toward regularities. In spite of its idiosyncrasies, the phonetic patterns of English do satisfy this requirement to a large extent. Even so, exceptions abound and PRO must maintain a delicate balance between its ability to recognize special "one shot" cases and its responsiveness to general patterns.

For example, "move" "love" and "cove" each require a different pronunciation for "o". However arbitrary these may be, PRO can learn to favor the correct pronunciation based on the contextual influence of an "m", "l", or "c" before the segment "ov." But now consider what happens when we add an "r" to the end of each word. "Mover" and "lover" retain the vowel sounds present in "move" and "love", but "cover" is not consistent with the sound from "cove." On the other hand, "over" is consistent with "cove." Using PRO's limited case length of 3 hypotheses, it is possible for PRO's Case Base to miss essential discriminating features when arbitrary conventions of this sort arise. Failings of this kind do not imply that PRO is seriously flawed. We could easily increase PRO's case length to 4 and handle the above instances without difficulty. If we increased the case length we would necessarily increase the size of the Case Base, but we would also decrease the number of context nodes we generate since it is harder to match a sequence of 4 hypotheses than a sequence of 3. A design modification of this sort could only result in im-

proved performance, but at the cost of greater memory requirements. If anything, we should be surprised that a case length of 3 is as effective as it is.

If we did enough training to see the Hypothesis Base and Case Base approach saturation, we would see the effects of the Statistical Base come into play. It is conceivable that continued enhancements in the Statistical Base might reverse any negative effects that appear during growth periods for the Hypothesis Base and the Case Base. We would speculate that the chances of this happening are greater the sooner the Case Base settles down. If the Case Base continued to grow at a significant rate after we had exhausted half the words of the language, and performance continued to degrade as the Case Base grew, then it is unlikely that statistical effects would have enough impact to do any good that far along. For this reason, it might be desirable to minimize the size of the Case Base. At the current time too little is known about PRO's long term performance to say much about these tradeoffs.

Although we have characterized PRO as a case-based reasoning system, it may be more narrowly described as a memory-based reasoning (MBR) system [Stanfill and Waltz, 1986]. Within the MBR paradigm, PRO is very similar to MBRtalk [op. cit.] in its overall goals despite major differences between the two approaches. In terms of performance, MBRtalk attains a hit rate of 86%, but this is after training on 4438 words (PRO attained 76% after 750 words). It is also the case the MBRtalk requires a massively parallel architecture (the Connection Machine) while PRO runs reasonably in a Common Lisp environment with 4M of memory.

Apart from further experimentation with PRO by expanding its training corpus, we see two general directions for future research. On the one hand, we would like to identify other problem areas where numerical relaxation is an effective strategy for accessing large knowledge bases organized around cases. Any domain where frequency data tends to correspond with generalizations is a good candidate for these investigations. On the other hand, we also want to investigate methods for generalizing numerical relaxation to symbolic processes of constraint propagation. Symbolic constraint propagation is a richer and more powerful technique than numerical relaxation. In domains where numerical data is inappropriate or unobtainable, we would still like to pursue the notion of network stabilization as an effective means for mediating competing cases in a large knowledge base of available cases.

We therefore view PRO as a single application illustrating a general framework for case-based reasoning systems. The general utility of these methods can be determined only by extensive experimentation. For example, the ideas behind PRO are now being applied to the task of conceptual sentence analysis [Lehnert, 1986, Lehnert, 1987]. As we gain more experience with PRO's approach to case-based reasoning and memory organization, we will be in a better position to characterize the tasks and knowledge domains best suited to these techniques.

References

[Bain, 1986] Bain, W. A Case-based Reasoning System for Subjective Assessment, *Proceedings of the Fifth National Conference on Artificial Intelligence* pp. 523-527, 1986.

[Berwick, 1986] Berwick, R.C. Learning from positive-only examples: the subset principle and three case studies in *Machine Learning* vol. 2. (eds. Michalski, Carbonell and Mitchell). pp. 625-645. Morgan Kaufmann, 1986.

[Feldman and Ballard, 1982] Feldman, J.A., and Ballard, D.H. Connectionist models and their properties *Cognitive Science* Vol. 6, no. 3. pp.205-254, 1982.

[Hammond, 1986] Hammond, K. CHEF: A Model of Case-Based Planning, *Proceedings of the Fifth National Conference on Artificial Intelligence* pp. 267-271, 1986.

[Kolodner, Simpson, and Sycara-Cyranski, 1985] Kolodner, J., Simpson, R., and Sycara-Cyranski, K. A Process Model of Case-Based Reasoning in Problem Solving, in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* pp. 284-290, 1985.

[Lebowitz, 1986] Lebowitz, M. Not the Path to Perdition: The Utility of Similarity-Based Learning, *Proceedings of the Fifth National Conference on Artificial Intelligence* pp. 533-537, 1986.

[Lehnert, 1986] Lehnert, W.G. Utilizing episodic memory for the integration of syntax and semantics CPTM #15. Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1986.

[Lehnert, 1987] Lehnert, W.G. (in press) Learning to Integrate Syntax and Semantics, Machine Learning Workshop, 1987.

[Rissland and Ashley, 1986] Rissland. E. and Ashley, K. Hypotheticals as Heuristic Device *Proceedings of the Fifth National Conference on Artificial Intelligence* pp. 289-297, 1986.

[Stanfill and Waltz, 1986] Stanfill, C., and Waltz, D. Toward memory-based reasoning. *Communications of the ACM*, vol. 29, no.12. pp.1213-28, 1986.