HIDDEN UNDERSTANDING MODELS OF NATURAL LANGUAGE

Scott Miller
College of Computer Science
Northeastern University
Boston, MA 02115
millers@ccs.neu.edu

Robert Bobrow, Robert Ingria, Richard Schwartz
BBN Systems and Technologies
70 Fawcett St.,
Cambridge, MA 02138
rusty, ingria, schwartz@BBN.COM

Abstract

We describe and evaluate hidden understanding models, a statistical learning approach to natural language understanding. Given a string of words, hidden understanding models determine the most likely meaning for the string. We discuss 1) the problem of representing meaning in this framework, 2) the structure of the statistical model, 3) the process of training the model, and 4) the process of understanding using the model. Finally, we give experimental results, including results on an ARPA evaluation.

1 Introduction

Hidden understanding models are an innovative class of statistical mechanisms that, given a string of words, determines the most likely meaning for the string. The overall approach represents a substantial departure from traditional techniques by replacing hand-crafted grammars and rules with statistical models that are automatically learned from examples. Hidden understanding models were primarily motivated by techniques that have been extremely successful in speech recognition, especially hidden Markov models [Baum, 72]. Related techniques have previously been applied to the problem of identifying concept sequences within a sentence [Pieraccini *et al.*, 91]. In addition, the approach contains elements of other natural language processing techniques including semantic grammars [Waltz, 78; Hendrix, 78], augmented transition networks (ATNs) [Woods, 70], probabilistic parsing [Fujisaki *et al.*, 89; Chitrao and Grishman, 90; Seneff, 92], and automatic grammar induction [Pereira and Schabes, 92].

Hidden understanding models are capable of learning a variety of meaning representations, ranging from simple domain-specific representations, to ones at a level of detail and sophistication comparable to current natural language systems. In fact, a hidden understanding model can be used to produce a representation with essentially the same information content as the semantic graph used by the Delphi system [Bobrow et al., 90], a general purpose NLP system, which utilizes a modified Definite Clause Grammar formalism. This fact made it possible to interface a hidden understanding system to the discourse processing and database retrieval components of Delphi to produce a complete "end to end" system. This hybrid system participated in the 1993 ATIS natural language evaluation. Although only four months old, the scores achieved by the combined system were quite respectable.

Because of differences between language understanding and speech recognition, significant changes are required in the hidden Markov model methodology. Unlike speech, where each phoneme results in a local sequence of spectra, the relation between the meaning of a sentence and the sequence of words is not a simple linear sequential model. Language is inherently nested, with subgroups of concepts within other concepts.

A statistical system for understanding language must take this and other differences into account in its overall design. In principle, we have the following requirements for a hidden understanding system:

- A notational system for expressing meanings.
- A statistical model that is capable of representing meanings and the association between meanings and words.
- An automatic training program which, given pairs of meanings and word sequences, can estimate the parameters of a statistical model.
- An understanding program that can search the statistical model to find the most likely meaning given a word sequence.

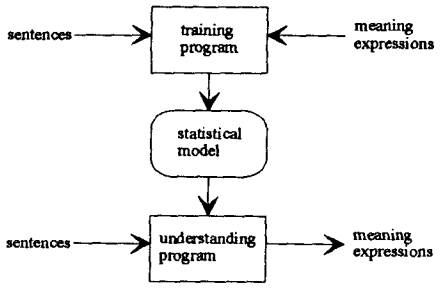

Figure 1. The Main Components of a Hidden Understanding System.

Below, we describe solutions for each of these requirements, and describe the relationship of these solutions to other work in stochastic grammars and probabilistic parsing. Finally, we will report on initial experiments with hidden understanding models.

2 Expressing Meanings

One of the key requirements for a hidden understanding model is that the meaning representation must be both precise and appropriate for automatic learning techniques. Specifically, we require a meaning representation that is:

- Expressive. It must be able to express meanings over the entire range of utterances that are likely to occur in an application.
- Annotatable. It must be possible to produce accurate annotations for a sufficiently large corpus with an acceptable level of human effort.
- Trainable. It must be possible to estimate the model parameters from a reasonable number of training examples.
- Tractable. There must be a computationally tractable algorithm capable of searching the meaning space.

In order to facilitate annotation of a training corpus, meaning expressions should be as simple as possible. Frame based representations, such as the example shown in figure 2, have the advantage that they are relatively simple to understand.

A difficulty with this style of representation is that the frames do not align directly to the words of the sentences. In particular, a meaning frame contains few explicit clues as to how the words of a sentence imply the structural characteristics of the frame. Tree structured meaning representations, discussed in the next section, have the advantage that they can be fully aligned to the words of a sentence. The cost is that these tree structured representations are more detailed than their frame based counterparts, thereby requiring greater annotation effort. Fortunately, the techniques developed for tree structured representations can be extended to simpler frame representations as well.

SHOW:
  FLIGHTS:
    TIME:
      PART-OF-DAY: *morning*
    ORIGIN:
      CITY: *Boston*
    DEST:
      CITY: *San Francisco*
    DATE:
      DAY-OF-WEEK: *Tuesday*

Please show me morning flights from Boston to San Francisco on Tuesday.

Figure 2. A Frame Based Meaning Representation.

2.1 Tree Structured Meaning Representations

The central characteristic of a tree structured representation is that individual concepts appear as nodes in a tree, with component concepts appearing as nodes attached directly below them. For example, the concept of a *flight* in the ATIS domain has component concepts including *airline*, *flight number*, *origin*, and *destination*. These could then form part of the representation for the phrase: *United flight 203 from Dallas to Atlanta*. The use of a hierarchical representation is one characteristic that distinguishes hidden understanding models from earlier work in which meaning is represented by a linear sequence of concepts [Pieraccini *et al.*, 91].

A requirement for tree structured representations is that the order of the component concepts must match the order of the words they correspond to. Thus, the representation of the phrase *flight 203 to Atlanta from Dallas on United* includes the same nodes as the earlier example, but in a different order. For both examples, however, the interpretation is identical.

At the leaves of a meaning tree are the words of the

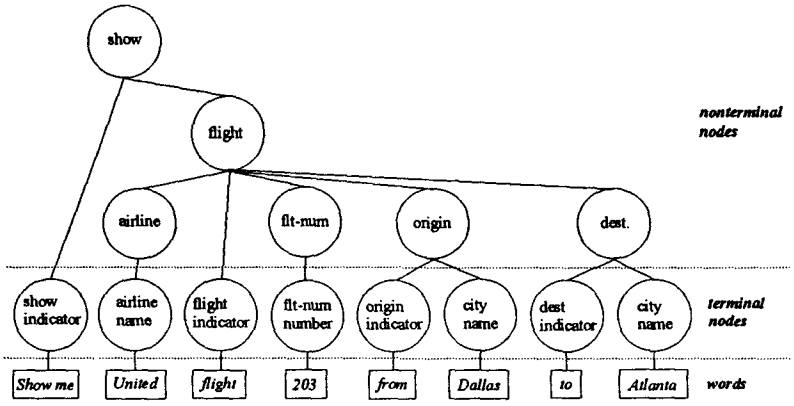

Figure 3. A Tree Structured Meaning Representation.

sentence. We distinguish between nodes that appear above other nodes, and those that appear directly above the words. These will be referred to as nonterminal nodes and terminal nodes respectively, forming two disjoint sets. No node has both words and other nodes appearing directly below it. Figure 3 shows an example of a typical meaning tree. In this example, the *flight* node represents the abstract concept of a flight, which is a structured entity that may contain an origin, a destination, and other component concepts. Appearing directly above the word "flight" is a terminal node, which we call a *flight indicator*. This name is chosen to distinguish it from the *flight* node, and also because the word *flight*, in some sense, indicates the presence of a flight concept. Similarly, there are *airline indicators*, *origin indicators*, and *destination indicators*.

One view of these tree structured representations is that they are parse trees produced according to a semantic grammar. In this view, the dominance relations of the grammar are predetermined by the annotation schema, while the precedence relations are learned from the training examples.

2.2 Alternative Tree Representations

Tree structured meaning expressions can range in complexity from simple special purpose sublanguage representations to the structural equivalent of detailed syntactic parse trees. The possibilities are limited only by two fundamental requirements: (1) semantic concepts must be hierarchically nested within a tree structure, and (2) the sets of terminal and nonterminal nodes must remain disjoint. Both of these requirements can be satisfied by trees possessing most of the structural characteristics of conventional syntactic parse trees. Since our objective is to model meaning, the nodes must still be labeled to reflect semantic categories. However, additional and augmented labels may be introduced to reflect syntactic categories as well.

Representations of this form contain significantly more internal structure than specialized sublanguage models. This can be seen in the example in figure 4. The specialized sublanguage representation requires only seven nodes, while a full syntactically motivated analysis requires fifteen. The additional nodes are used to distinguish what is being shown to whom, to reflect the fact that the stopover phrase is part of a relative clause, and to determine the internal structure of the relative clause.

One interesting characteristic of these more elaborate trees is their similarity to those produced by classical, linguistically motivated, natural language systems. Thus, a hidden understanding model can serve to replace the part-of-speech tagger, parser, and semantic interpreter of a classical system. Instead of writing grammar and semantic interpretation rules by hand, the training program automatically constructs a statistical model from examples of meaning trees.

Regardless of the details of the tree structure and labels, the components comprising a hidden understanding system remain unchanged. The only difference is in how the system is trained.

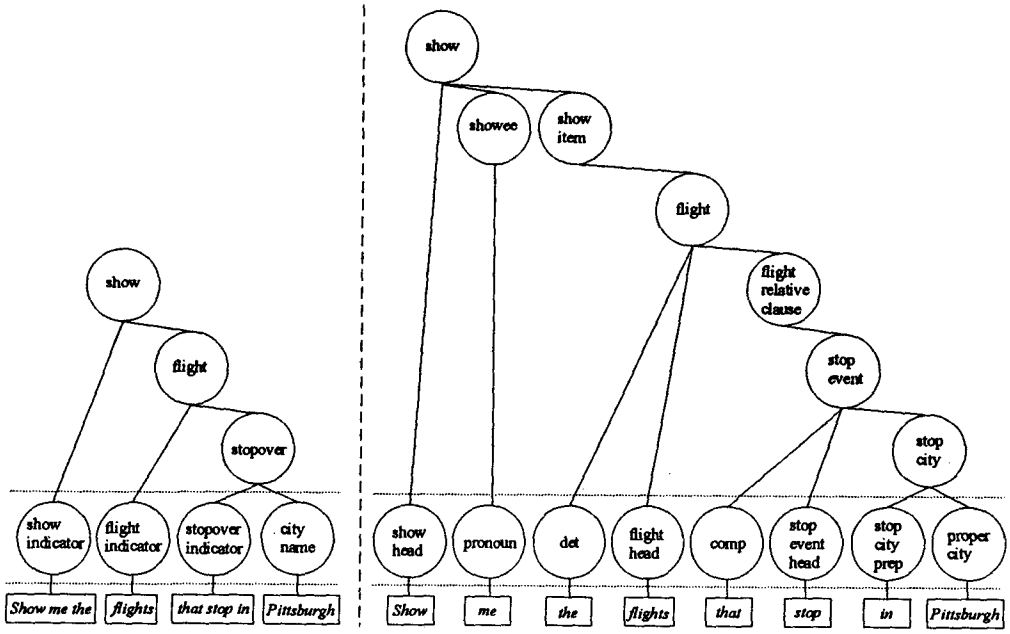

Figure 4. A Specialized Sublanguage Analysis and a Full Syntactic Analysis.

2.3 Frame Based Representations

One way to think of a frame based meaning is as a partially specified tree in which some words are not accounted for. Nevertheless, a frame representation is a complete meaning representation in the sense that it fully specifies the concepts and structure comprising the meaning. In terms of a tree structured representation, the set of nonterminal nodes is fully specified, while some of the terminal nodes may be omitted.

The missing terminal nodes are said to be hidden, in the sense that every word is required to align to some terminal node, but the alignment is not necessarily given by the meaning frame. These hidden nodes must later be aligned as part of the training process. The general idea is to assign a small number of free terminal nodes (typically one or two) beneath every nonterminal node. These are then free to align to any unassigned words, provided that the overall tree structure is not violated. An EM algorithm (Estimate-Maximize) is used to organize the unassigned terminal nodes into classes that correspond to individual words and phrases, and that bind to particular abstract concepts. Figure 5 shows the complete meaning tree with hidden nodes corresponding to the frame in figure 2.

If we consider tree structured meaning expressions as parse trees which are generated according to some incompletely specified grammar, then the problem of aligning the hidden nodes can be considered as a grammar induction problem. In this way, the problem of aligning the hidden nodes given only a partially specified set of trees is analogous to the problem of fully parsing a training corpus given only a partial bracketing. The difference is that while a partial bracketing determines constituent boundaries that cannot be crossed, a partially specified tree determines structure that must be preserved.

3 The Statistical Model

One central characteristic of hidden understanding models is that they are *generative*. From this viewpoint, language is produced by a two component statistical process. The first component chooses the meaning to be expressed, effectively deciding "what to say". The second component selects word sequences to express that meaning, effectively deciding "how to say it". The first phase is referred to as the *semantic language model*, and can be thought of as a stochastic process that produces meaning expressions selected from a universe of meanings. The second phase is referred to as the *lexical realization model*, and can be thought of as a stochastic process that generates words once a meaning is given.

By analogy with hidden Markov models, we refer to the combination of these two models as a hidden understanding model. The word "hidden" refers to the fact that only words can be observed. The internal states of each of the two models are unseen and must be inferred from the words. The problem of language understanding, then, is to recover the most likely meaning structure given a sequence of words. More formally, understanding a word sequence W is accomplished by searching among all possible meanings for some meaning M such that $P(M \mid W)$ is maximized. By Bayes Rule, $P(M \mid W)$ can be rewritten as:

$$P(M|W) = \frac{P(W|M)P(M)}{P(W)}$$

Now, since $P(W)$ does not depend on M, maximizing $P(M \mid W)$ is equivalent to maximizing the product $P(W \mid M)$ $P(M)$. However, $P(M \mid W)$ is simply our lexical realization model, and $P(M)$ is simply our semantic language model. Thus, by searching a combination of these models it is possible to find the maximum likelihood meaning M given word sequence W. Considering the statistical model as a stochastic grammar, the problem of determining M given W is analogous to the problem of finding the most likely derivation for W according to that grammar.

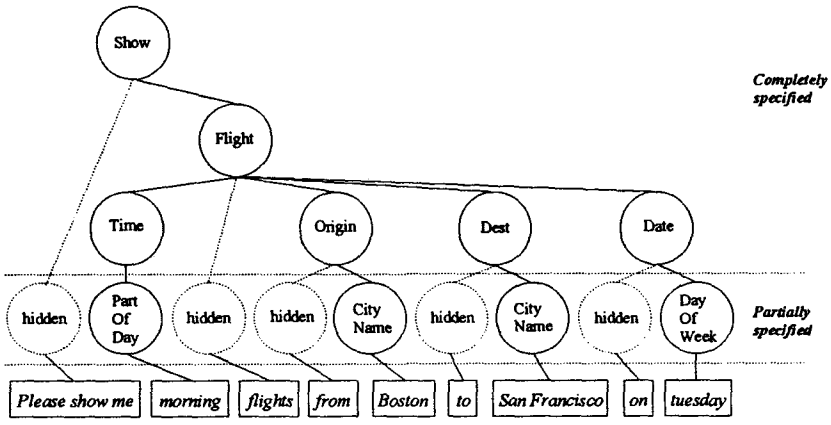

Figure 5. A Tree Structure Corresponding to a Frame Representation.

3.1 Semantic Language Model

For tree structured meaning representations, individual nonterminal nodes determine particular abstract semantic concepts. In the semantic language model, each abstract concept corresponds to *a probabilistic state transition network*. All such networks are then combined into a single *probabilistic recursive transition network*, forming the entire semantic language model.

The network corresponding to a particular abstract concept consists of states for each of its component concepts, together with two extra states that define the entry and exit points. Every component concept is fully connected to every other component concept, with additional paths leading from the entry state to each component concept, and from each component concept to the exit state. Figure 6 shows a sample network corresponding to the *flight* concept. Of course, there are many more flight component concepts in the ATIS domain than actually appear in this example.

Associated with each arc is a probability value, in a similar fashion to the TINA system [Seneff, 92]. These probabilities have the form $P(State_n \mid State_{n-1}, Context)$, which is the probability of a taking transition from one state to another within a particular context. Thus, the arc from *origin* to *dest* has probability $P(dest \mid origin, flight)$, meaning the probability of entering *dest* from *origin* within the context of the *flight* network. Presumably, this probability is relatively high, since people usually mention the destination of a flight directly after mentioning its origin. Conversely, $P(origin \mid dest, flight)$ is probably low because people don't usually express concepts in that order. Thus, while all paths through the state space are possible, some have much higher probabilities than others.

Within a concept network, component concept states exist for both nonterminal concepts, such as *origin*, as well as terminal concepts, such as *flight indicator*. Arrows pointing into nonterminal states indicate entries into other networks, while arrows pointing away indicate exits out of those networks. Terminal states correspond to networks as well, although these are determined by the lexical realization model and have a different internal structure. Thus, every meaning tree directly corresponds directly to some particular path through the state space. Figure 7 shows a meaning tree and its corresponding path through state space.

Viewed as a grammar, the semantic language model is expressed directly as a collection of networks rather than as a collection of production rules. These networks represent grammatical constraints in a somewhat different fashion than do grammars based on production rules. In this model, constituents may appear beneath nonterminal nodes in any arbitrary order, while preferences for some orderings are determined through the use of probabilities. By contrast, most grammars limit the ordering of constituents to an explicit set which is specified by the grammar rules. The approach taken in the TINA system eliminates many ordering constraints while retaining the local state transition constraints determined by its grammar. We believe that an unconstrained ordering of constraints increases parsing robustness, while the preferences determined by the arc probabilities help minimize overgeneration.

3.2 Lexical Realization Model

Just as nonterminal tree nodes correspond to networks in the semantic language model, terminal nodes correspond to networks in the lexical realization model. The difference is that semantic language networks specify transition

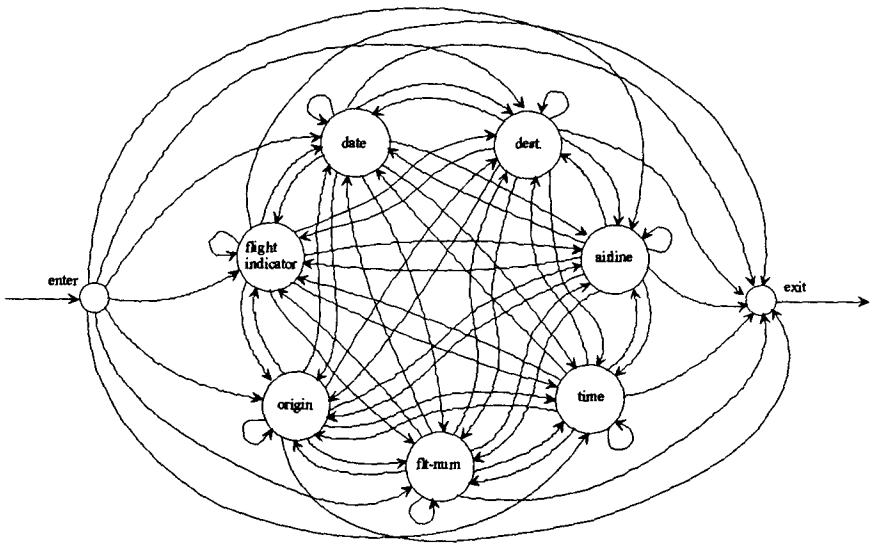

Figure 6. A Partial Network Corresponding to the ATIS *Flight* Concept.

probabilities between states, while lexical realization networks specify transition probabilities between words. Lexical realization probabilities have the form $P(word_n \mid word_{n-1}, context)$, which is the probability of taking a transition from one word to another given a particular context. Thus, $P(show \mid please, show\text{-}indicator)$ is the probability that the word *show* follows the word *please* within the context of a *show indicator* phrase. In addition, there are two pseudo-words, ***begin**** and ****end****, which indicate the beginning and ending of phrases. Thus, we have probabilities such as $P(please \mid *begin*, show\text{-}indicator)$, which is the probability that *please* is the first word of a *show indicator* phrase, and $P(*end* \mid me, show\text{-}indicator)$, which is the probability of exiting a *show indicator* phrase given that the previous word was *me*.

4 The Understanding Component

As we have seen, understanding a word string W requires finding a meaning M such that the probability $P(W \mid M)$ $P(M)$ is maximized. Since, the semantic language model and the lexical realization model are both probabilistic networks, $P(W \mid M)$ $P(M)$ is the probability of a particular path through the combined network. Thus, the problem of understanding is to find the highest probability path among all possible paths, where the probability of a path is the product of all the transition probabilities along that path.

$$P(Path) = \prod_{t \in Path} \begin{bmatrix} P(state_n \mid state_{n-1}, context) & \text{if } t \text{ in Semantic Language Model} \\ P(word_n \mid word_{n-1}, context) & \text{if } t \text{ in Lexical Realization Model} \end{bmatrix}$$

Thus far, we have discussed the need to search among all meanings for one with a maximal probability. In fact, if it were necessary to search every path through the combined network individually, the algorithm would require exponential time with respect to sentence length. Fortunately, this can be drastically reduced by combining the probability computation of common subpaths through dynamic programming. In particular, because our meaning representation aligns to the words, the search can be efficiently performed using the well-known Viterbi [Viterbi, 67] algorithm.

Since our underlying model is a recursive transition network, the states for the Viterbi search must be allocated dynamically as the search proceeds. In addition, it is necessary to prune very low probability paths in order to keep the computation tractable. We have developed an elegant algorithm that integrates state allocation, Viterbi search, and pruning all within a single traversal of a tree-like data structure. In this algorithm, each of the set of currently active states is represented as a node in a tree. New nodes are added to the tree as the computation pushes into new subnetworks that are not currently active. Stored at each node is the probability of the most likely path reaching that state, together with a backpointer sufficient to recreate the path later if needed. Whenever the probability of all states in a subtree falls below the threshold specified by the beam width, the entire subtree is pruned away.

5 The Training Component

In order to train the statistical model, we must estimate transition probabilities for the semantic language model and lexical realization model. In the case of fully specified meaning trees, each meaning tree can be straightforwardly converted into a path through state space. Then, by counting occurrence and transition frequencies along those paths, it is possible to form simple estimates of the transition probabilities. Let $C(state_m, context_s)$ denote the number of times $state_m$ has occurred in $context_s$, and let $C(state_n \mid state_m, context_s)$ denote the number of times that this condition has led to a transition to state $state_n$. Similarly, define counts $C(word_m, context_l)$ and $C(word_n \mid word_m, context_l)$. Then, a direct estimate of the probabilities is given by:

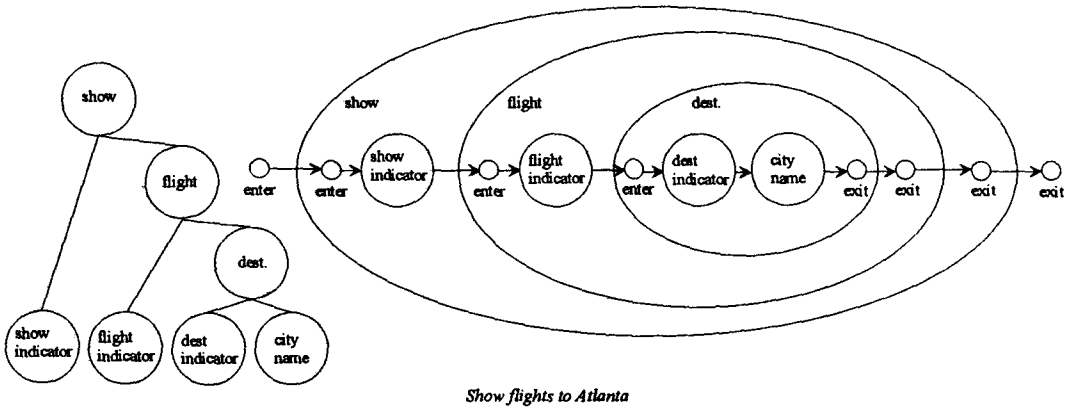

Figure 7. A Meaning Tree and its Corresponding Path Through State Space.

$$\hat{P}(state_n | state_m, context) = \frac{C(state_n | state_m, context)}{C(state_m, context)},$$

and

$$\hat{P}(word_n | word_m, context) = \frac{C(word_n | word_m, context)}{C(word_m, context)}.$$

In order to obtain robust estimates, these simple estimates are smoothed with *backed-off* estimates [Good, 53], using techniques similar to those used in speech recognition [Katz, 87; Placeway *et al.*, 93]. Thus, $\hat{P}(state_n \mid state_m, context)$ is smoothed with $\hat{P}(state_n \mid context)$, and $\hat{P}(word_n \mid word_m, context)$ is smoothed with $\hat{P}(word_n \mid context)$. Robustness is further increased through word classes. For example, *Boston* and *San Francisco* are both members of the class of cities.

In the case of frame based representations, it is not always possible to construct an exact path through the state space corresponding to a meaning representation. Nevertheless, since frames are treated as partially specified trees, most of the path can be reconstructed, with some portions undetermined. Then, the partial path can be used to constrain a gradient descent search, called the forward-backward algorithm [Baum, 72] for estimating the model parameters. This algorithm is an iterative procedure for adjusting the model parameters so as to increase the likelihood of generating the training data, and is an instance of the well-known class called EM (Estimate-Maximize) algorithms.

6 Experimental Results

We have implemented a hidden understanding system and performed a variety of experiments. In addition, we participated in the 1993 ARPA ATIS NL evaluation.

One experiment involved a 1000 sentence ATIS corpus, annotated according to a simple specialized sublanguage model. The annotation effort was split between two annotators, one of whom was a system developer, while the other was not. To annotate the training data, we used a bootstrapping process in which only the first 100 sentences were annotated strictly by hand.

Thereafter, we worked in cycles of:

1. Running the training program using all available annotated data.
2. Running the understanding component to annotate new sentences.
3. Hand correcting the new annotations.

Annotating in this way, we found that a single annotator could produce 200 sentences per day. We then extracted the first 100 sentences as a test set, and trained the system on the remaining 900 sentences. The results were as follows:

- 61% matched exactly.
- 21% had correct meanings, but did not match exactly.
- 28% had the wrong meaning.

Another experiment involved a 6000 sentence ATIS corpus, annotated according to a more sophisticated meaning model. In this experiment, the Delphi system automatically produced the annotation by printing out its own internal representation for each sentence, converted into a more readable form. In order to maintain high quality annotations, we used only sentences for which Delphi produced a complete parse, and for which it also retrieved a correct answer from the database. We then removed 300 sentences as a test set, and trained the system on the remaining 5700. The results were as follows:

- 85% matched exactly.
- 8% had correct meanings, but did not match exactly.
- 7% had the wrong meaning.

For the ARPA evaluation, we coupled our hidden understanding system to the discourse and backend components of the Delphi. Using the entire 6000 sentence corpus described above as training data, the system produced a score of 26% simple error on the ATIS NL evaluation. By examining the errors, we have reached the conclusion that nearly half are due to simple programming issues, especially in the interface between Delphi and the hidden understanding system. In fact, the interface was still incomplete at the time of the evaluation.

We have just begun a series of experiments using frame based annotations, and are continuing to refine our techniques. In a preliminary test involving a small corpus of 588 ATIS sentences, the system correctly aligned the hidden states for over 95% of the sentences in the corpus.

7 Limitations

Several limitations to our current approach are worth noting. In a small number of cases, linguistic movement phenomena make it difficult to align the words of a sentence to any tree structured meaning expression without introducing crossings. In most cases, we have been able to work around this problem by introducing minor changes in our annotation such that the tree structure is maintained. A second limitation, due to the local nature of the model, is an inability to handle nonlocal phenomena such as coreference. Finally, in some cases the meaning of a sentence depends strongly upon the discourse state, which is beyond the scope of the current model.

8 Conclusions

We have demonstrated the possibility of automatically learning semantic representations directly from a training corpus through the application of statistical techniques. Empirical results, including the results of an ARPA

evaluation, indicate that these techniques are capable of relatively high levels of performance.

While hidden understanding models are based primarily on the concepts of hidden Markov models, we have also shown their relationship to other work in stochastic grammars and probabilistic parsing.

Finally, we have noted some limitations to our current approach. We view each of these limitations as opportunities for further research and exploration.

Acknowledgments

The work reported here was supported in part by the Defense Advanced Research Projects Agency under ARPA Contract No. N00014-92-C-0035. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

References

1. E. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes," Inequalities 3:1-8, 1972

2. Bobrow, R. Ingria, and D. Stallard, "Syntactic and Semantic Knowledge in the DELPHI Unification Grammar," *Proceedings, Speech and Natural Language Workshop*, pp. 230-236, June 1990

3. Chitrao, and R. Grishman, "Statistical Parsing of Messages," *Proceedings, Speech and Natural Language Workshop*, pp. 263-276, Morgan Kaufmann Publishers, June 1990

4. Fujisaki, F. Jelinek, J. Cocke, E. Black, T. Nishino, "A Probabilistic Parsing Method for Sentence Disambiguation," *International Parsing Workshop*, pp. 85-90, 1989

5. Good, "The Population Frequencies of Species and the Estimation of Population Parameters," Biometrika 40, pp.237-264, 1953

6. G.G Hendrix, "Semantic Aspects of Translation," Understanding Spoken Language, pp. 193-226, New York, Elsevier, 1978

7. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-35, pp. 400-401, 1987

8. Pereira and Y. Schabes, "Inside-Outside Reestimation from Partially Bracketed Corpora," *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pp.128-135, Newark, Delaware, 1992

9. R. Pieraccini, E. Levin, and C. Lee, *Stochastic Representation of Conceptual Structure in the ATIS Task: DARPA Speech and Natural Language Workshop*, pp. 121-124, Feb. 1991.

10. Placeway, R. Schwartz, P. Fung, L. Nguyen, "The Estimation of Powerful Language Models from Small and Large Corpora," IEEE ICASSP, II:33-36

11. Seneff, "TINA: A Natural Language System for Spoken Language Applications," *Computational Linguistics* Vol. 18, Number 1, pp. 61-86, March 1992

12. J. Viterbi, "Error Bounds for Convolutional Codes and an Asympotically Optimum Decoding Algorithm," IEEE Transactions on Information Theory IT-13(2):260-269, April 1967

13. D.L Waltz, "An English Language Question Answering System for a Large Relational Database," *Communications of the ACM* 21(7):526-39, 1978.

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