STATISTICAL LANGUAGE PROCESSING USING HIDDEN UNDERSTANDING MODELS

Scott Miller
College of Computer Science
Northeastern University
Boston, MA 02115

Robert Bobrow
BBN Systems and Technologies
70 Fawcett St.,
Cambridge, MA 02138

Richard Schwartz
BBN Systems and Technologies
70 Fawcett St.,
Cambridge, MA 02138

Robert Ingria
BBN Systems and Technologies
70 Fawcett St.,
Cambridge, MA 02138

ABSTRACT

This paper introduces a class of statistical mechanisms, called hidden understanding models, for natural language processing. Much of the framework for hidden understanding models derives from statistical models used in speech recognition, especially the use of hidden Markov models. These techniques are applied to the central problem of determining meaning directly from a sequence of spoken or written words. We present an overall description of the hidden understanding methodology, and discuss some of the critical implementation issues. Finally, we report on experimental results, including results of the December 1993 ARPA evaluation.

1 INTRODUCTION

Hidden understanding models are an innovative application 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 approaches by replacing hand-crafted grammars and rules with statistical models that are automatically learned from examples. Advantages of this approach include potential improvements in both robustness and portability of natural language systems.

Hidden understanding models were 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 segmenting a sentence into a sequence of concept relations [Pieraccini *et al.*, 91]. However, because of differences between language understanding and speech recognition, significant changes are required in the speech recognition 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.

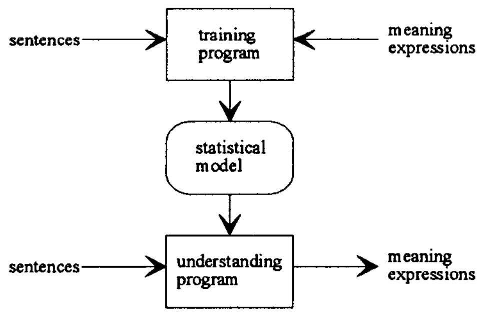

Figure 1: The main components of a hidden understanding system.

Below, we describe solutions for each of these requirements, and 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 expressive and appropriate for automatic learning techniques. Logical notations, such as the predicate calculus, are generally considered to possess sufficient expressive power. The difficulty lies in finding a meaning representation that can be readily aligned to the words of a sentence, and for which there is a tractable probability model for meanings. To satisfy these requirements, we have developed a family of representations which we call *tree structured meaning representations.*

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.* We require 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, the interpretation is identical. More formally, the meaning of a tree structured representation is invariant with respect to the left-to-right order of the component concept nodes.

At the leaves of a meaning tree are the words of the 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. In the current example, a *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.* These nodes can be thought of as elements in a *specialized sublanguage* for expressing meaning in the ATIS domain.

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.

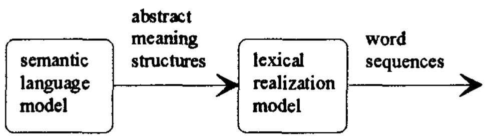

Figure 3: Language as a generative process.

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 $P(M|W)$ is maximized. By Bayes Rule, $P(M|W)$ can be rewritten as:

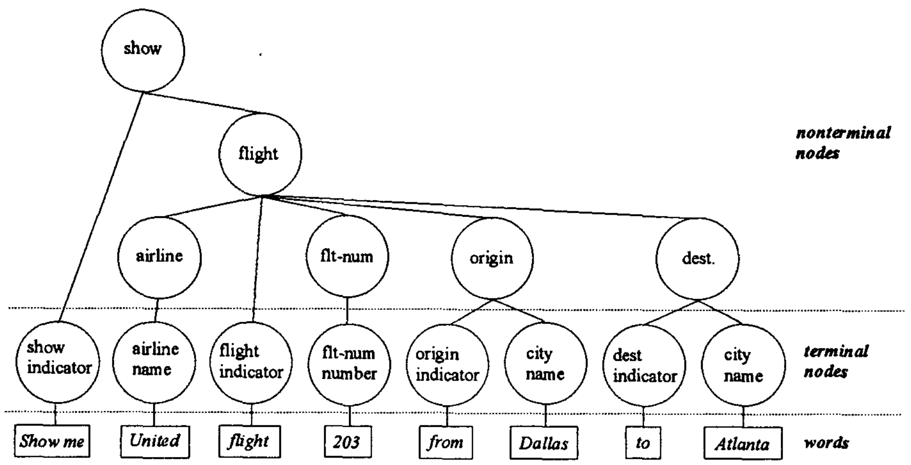

Figure 2: An example of a tree structure meaining representation.

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

Now, since $P(W)$ does not depend on M, maximizing $P(M|W)$ is equivalent to maximizing the product $P(W|M)\,P(M)$. However, $P(W|M)$ 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 most likely meaning M given word sequence W.

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 4 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 $P(State_n \,|\, State_{n-1}, Context)$, which is the probability of taking a transition from one state to another within a particular context. Thus, the arc from *origin* to *dest* has probability $P(dest \,|\, origin, flight)$, 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, $P(origin \,|\, dest, flight)$ 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 5 shows a meaning tree and its corresponding path through the state space.

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 probabilities between states, while lexical realization networks specify transition probabilities between words. Lexical realization

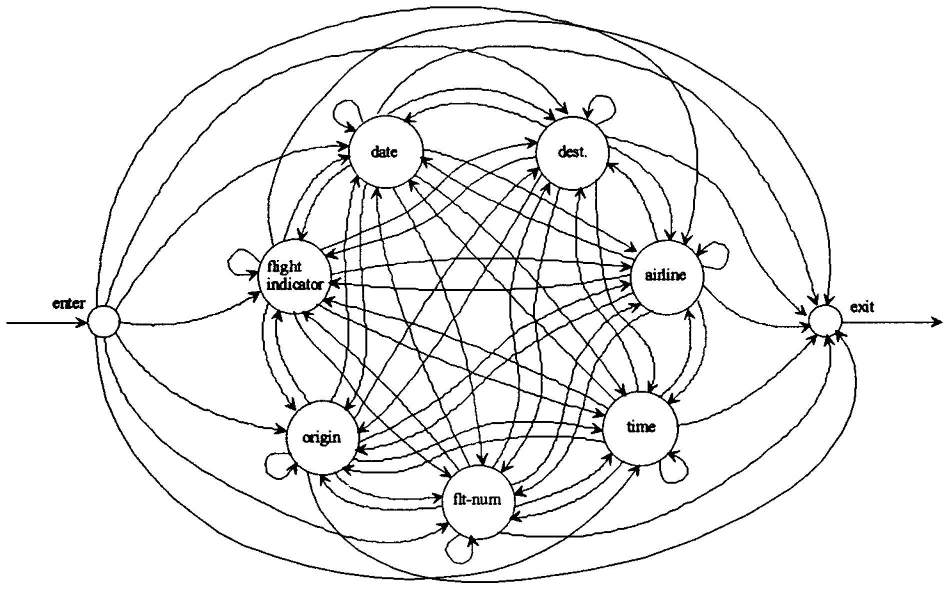

Figure 4: A partial network corresponding to the *flight* concept.

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 $P(W|M)\,P(M)$ is maximized. Since, the semantic language model and the lexical realization model are both probabilistic networks, $P(W|M)\,P(M)$ 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.

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

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.

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 $C(state_n|state_m, context_s)$ denote the number of times that this condition has led to a transition to state $state_n$. Similarly, define counts $C(word_m, context_t)$ and $C(word_n|word_m, context_t)$. Then, a direct estimate of the probabilities is given by:

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

and

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

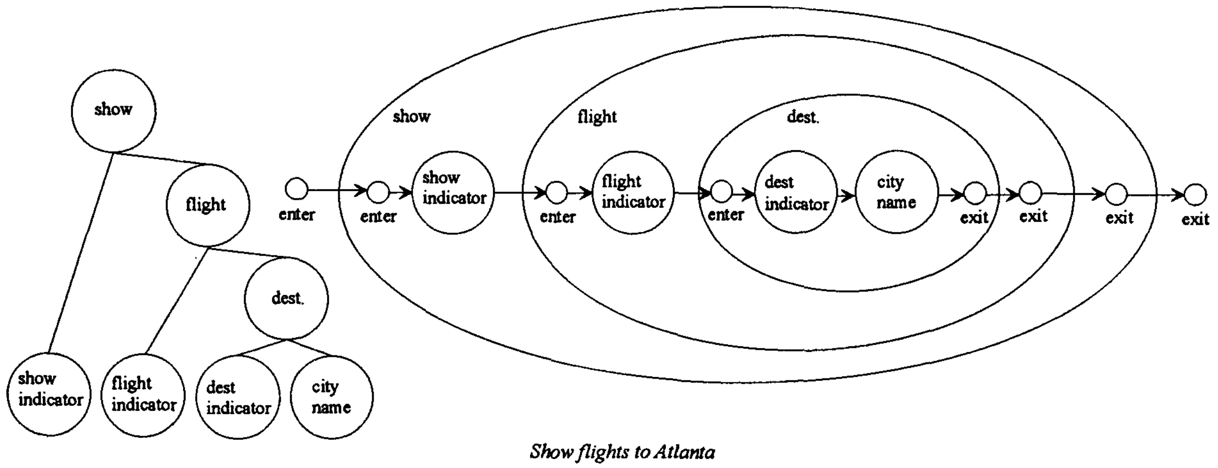

Figure 5: A meaning tree and its corresponding path through state space.

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, $\hat{P}(state_n|state_m,context)$ is smoothed with $\hat{P}(state_n|,context)$, and $\hat{P}(word_n|word_m,context)$ is smoothed with $\hat{P}(word_n|context)$. Robustness is further increased through word classes. For example, *Boston* and *San Francisco* are both members of the class of cities.

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. 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. 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 system. Using the entire 6000 sentence corpus described above as training data, the system produced a score of 23% 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.

ACKNOWLEDGEMENTS

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

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

3. S.M. 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

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

5. R. Pieraccini, E. Levin, C.H. Lee, "Stochastic Representation of Conceptual Structure in the ATIS Task," Proceedings of the Speech and Natural Language Workshop, pp. 121-124, Morgan Kaufmann Publishers, Feb. 1991

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

7. A. 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