Probabilistic Parsing for Spoken Language Applications

Stephanie Seneff
Spoken Language Systems Group
Laboratory for Computer Science
MIT Cambridge, MA 02139

Abstract

A new natural language system, TINA, has been developed for applications involving spoken language tasks, which integrates key ideas from context free grammars, Augmented Transition Networks (ATN's) [6], and Lexical Functional Grammars (LFG's) [1]. The parser uses a best-first search strategy, with probability assignments on all arcs obtained automatically from a set of example sentences. An initial context-free grammar, derived from the example sentences, is first converted to a probabilistic network structure. Control includes both top-down and bottom-up cycles, and key parameters are passed among nodes to deal with long-distance movement, agreement, and semantic constraints. The probabilities provide a natural mechanism for exploring more common grammatical constructions first. One novel feature of TINA is that it provides an automatic sentence generation capability, which has been very effective for identifying overgeneration problems. A fully integrated spoken language system using this parser is under development.

1 Introduction

Most parsers have been designed with the assumption that the input word stream is deterministic: i.e., at any given point in the parse tree it is known with certainty what the next word is. As a consequence, these parsers generally cannot be used effectively, if at all, to provide linguistically directed constraint in the speech recognition component of a speech understanding system. In a fully integrated speech understanding system, the recognition component should only be allowed to propose partial word sequences that the natural language component can interpret; any word sequences that are syntactically or semantically anomalous should probably be pruned prior to the acoustic match, rather than examined for approval in a verification mode. To operate in such a fully integrated mode, a parser has to have the capability of considering a multitude of hypotheses simultaneously. The control strategy should have a sense of which of these hypotheses, considering both linguistic and acoustic evidence, is most likely to be correct at any given instant in time, and to pursue that hypothesis only incrementally before reexamining the evidence. The linguistic evidence should include probability assignments on proposed hypotheses; otherwise the perplexity of the task becomes too high for practical recognition applications.

This paper describes a natural language system, TINA, which addresses many of these issues. The grammar is constructed by converting a set of context-free rewrite rules to a form that merges common elements on the right-hand side (RHS) of all rules sharing the same left-hand side (LHS). Elements on the LHS become parent nodes in a family tree. Through example sentences, they acquire knowledge of who their children are and how they can interconnect. Such a transformation permits considerable structure sharing among the rules, as is done in typical shift-reduce parsers [5]. Probabilities are established on arcs connecting pairs of right siblings rather than on rule productions. This has several advantages, which will be discussed later. Context-dependent constraints to deal with agreement and gaps are realized through simple logical functions applied to flags or features passed among immediate relatives.

2 General Description

TINA is basically a context-free grammar, implemented by expansion at run-time into a network structure, and augmented with flags/parameters that activate filtering operations. The grammar is built from a set of training sentences, using a bootstrapping procedure. Initially, each sentence is translated by hand into a list of the rules invoked to parse it. After the grammar has built up a substantial knowledge of the language, many new sentences can be parsed automatically, or with minimal intervention to add a few new rules incrementally. The arc probabilities can be incrementally updated after the successful parse of each new sentence.

The process of converting the rules to a network form is straightforward. All rules with the same LHS are combined to form a structure describing possible interconnections among children of a parent node associated with the left-hand category. A probability matrix connecting each possible child with each other child is constructed by counting the number of times a particular sequence of two siblings occurred in the RHS's of the common rule set, and normalizing by counting all pairs from the particular left-sibling to *any* right sibling. Two distinguished nodes, a START node and an END node, are included among the children of every grammar node. A subset of the grammar nodes are terminal nodes whose children are a list of vocabulary words.

This process can be illustrated with the use of a simple example. Consider the following three rules:

NP ⇒ ARTICLE NOUN
NP ⇒ ARTICLE ADJECTIVE NOUN
NP ⇒ ARTICLE ADJECTIVE ADJECTIVE NOUN

These would be converted to a network as shown in Figure 1, which would be associated with a grammar node named NP. Since ADJECTIVE is followed twice by NOUN and once by ADJECTIVE, the network shows a probability of 1/3 for the self loop and 2/3 for the advance to NOUN. Notice that the system has now generalized to include any number of adjectives in a row.

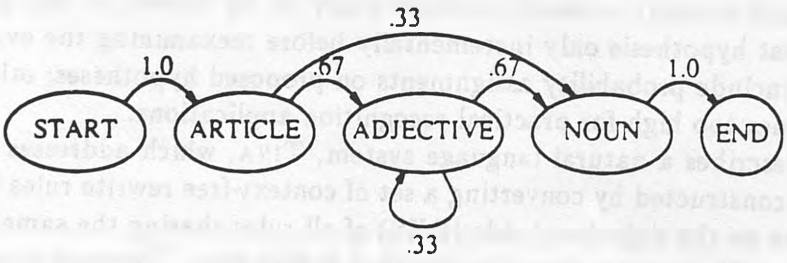

Figure 1: Probablistic Network Resulting from three Context-Free Rules given in Text.

A functional block diagram of the control strategy is given in Figure 2. At any given time, a set of active *parse nodes* are arranged on a priority queue. Each parse node contains a pointer to a corresponding grammar node, and has access to all the information needed to pursue its partial theory. The top node is popped from the queue, and it then creates a number of new nodes (either children or right siblings depending on its state), and inserts them into the queue according to their probabilities. If the node is an END node, it collects up all subparses from its sequence of left siblings, back to the START node, and passes the information up to the parent node, giving that node a completed subparse. The process can terminate on the first successful completion of a sentence, or the Nth successful completion if more than one hypothesis is desired.

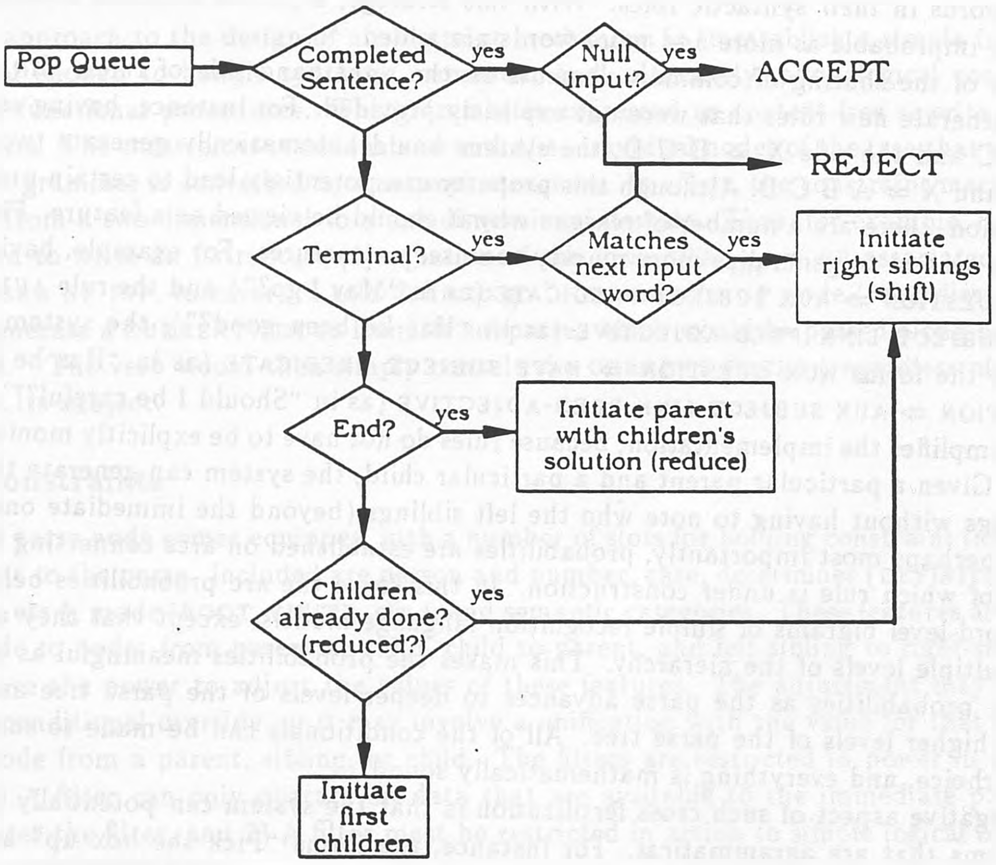

Figure 2: Functional Block Diagram of Control Strategy.

A parse in TINA begins with a single parse node linked to the grammar node SENTENCE, which is entered on the queue with probability 1.0. This node creates new parse nodes with categories like STATEMENT, QUESTION, and REQUEST, and places them on the queue, prioritized. If STATEMENT is the most likely child, it gets popped from the queue, and returns nodes indicating SUBJECT, IT, etc., to the queue. When SUBJECT reaches the top of the queue, it activates units such as NOUN-GROUP (for noun phrases and associated post-modifiers), GERUND, and NOUN-CLAUSE. Each node, after instantiating first-children, becomes inactive, pending the return of a successful subparse from a sequence of children. Eventually, the cascade of first-children reaches the terminal-node ARTICLE, which proposes the words "the," "a," and "an," testing these hypotheses against the input stream. If a match with "the" is found, then the ARTICLE node fills its subparse slot with the entry (ARTICLE "the"), and activates all of its possible right-siblings.

Whenever a terminal node has successfully matched an input word, the path probability is reset to 1.0. Thus the probabilities that are used to prioritize the queue represent not the *total* path probability but rather the probability *given* the partial word sequence. Each path climbs up from a terminal node and back down to a next terminal node, with each new node adjusting the path probability by multiplying by a new conditional probability. The resulting conditional path probability for a next word represents the probability of that word in its syntactic role given all preceding words in *their* syntactic roles. With this strategy, a partial sentence does not become increasingly improbable as more and more words are added. [1].

Because of the sharing of common elements on the right hand side of rules, TINA can automatically generate new rules that were not explicitly provided. For instance, having seen the rule X ⇒ A B C and the rule X ⇒ B C D, the system would automatically generate two new rules, X ⇒ B C, and X ⇒ A B C D. Although this property can potentialy lead to certain problems with overgeneration, there are a number of reasons why it should be viewed as a feature. First of all, it permits the system to generalize more quickly to unseen structures. For example, having seen the rule AUX-QUESTION ⇒ AUX SUBJECT PREDICATE (as in "May I go?") and the rule AUX-QUESTION ⇒ HAVE SUBJECT LINK PRED-ADJECTIVE (as in "Has he been good?"), the system would also understand the forms AUX-QUESTION ⇒ HAVE SUBJECT PREDICATE (as in "Has he left?") and AUX-QUESTION ⇒ AUX SUBJECT LINK PRED-ADJECTIVE (as in "Should I be careful?").[2] Secondly it greatly simplifies the implementation, because rules do not have to be explicitly monitored during the parse. Given a particular parent and a particular child, the system can generate the allowable right siblings without having to note who the left siblings (beyond the immediate one) were. Finally, and perhaps most importantly, probabilities are established on arcs connecting sibling pairs regardless of which rule is under construction. In this sense the arc probabilities behave like the familiar word-level bigrams of simple recognition language models, except that they apply to siblings at multiple levels of the hierarchy. This makes the probabilities meaningful as a product of conditional probabilities as the parse advances to deeper levels of the parse tree and also as it returns to higher levels of the parse tree. All of the conditionals can be made to sum to one for any given choice, and everything is mathematically sound.

One negative aspect of such cross fertilization is that the system can potentially generalize to include forms that are agrammatical. For instance, the forms "Pick the box up" and "Pick up the box," if defined by the same LHS name, would allow the system to include rules producing forms such as "Pick up the box up" and "Pick up the box up the box!" This problem can be overcome either by giving the two structures different LHS names or by grouping "up the box" and "the box up" into distinct parent nodes, adding another layer to the hierarchy on the RHS. A third alternative is to include a PARTICLE slot among the features which, once filled, cannot be refilled. In fact, there were only a few situations where such problems arose, and they were always correctable.

3 Constraints and Gaps

This section describes how TINA handles several issues that are often considered to be part of the task of a parser. These include agreement constraints, semantic restrictions, subject-tagging for verbs, and long distance movement (often referred to as *gaps*, or the *trace*, as in "(which article)$_i$

[1]Some modification of this scheme will be necessary when the input stream is not deterministic. See [4] for a discussion of these very important issues regarding scoring in a best-first search.

[2]The auxiliary verb sets the mode of the main verb to be root or past participle as appropriate.

do you think I should read (t_i)?"). TINA is particulary effective in handling gaps. Complex cases of nested or chained gaps are handled correctly, and appropriately ill-formed gaps are rejected. The mechanism resembles the *Hold* register idea of ATN's [6] and the treatment of bounded domination metavariables in LFG's ([1], p. 235 ff), but I believe it is more straightforward than both of these.

3.1 Design Philosophy

Our approach to the design of a constraint mechanism is to establish a simple framework that is general enough to handle syntactic, semantic, and, ultimately, phonological constraints using identical functional procedures. The grammar is expressed as context-free rewrite rules without constraints. The constraints reside instead with the individual nodes of the tree that are established when the grammar is converted to a network structure. In effect, the constraint mechanism is thus reduced from a two-dimensional to a one-dimensional domain. Thus, for example, it would not be permitted to write an f-structure [1] equation of the form $\text{SUBJ}_{\text{Inf}} \Rightarrow$ NP associated with the rule VP $\Rightarrow$ VERB NP INF, to cover, "I told John to go." Instead, the NP node (regardless of its parent) would generate a CURRENT-FOCUS from its subparse, which would be passed along passively to the verb "go." The verb would then simply consult the CURRENT-FOCUS (regardless of its source) to establish its subject.

3.2 Constraints

Each parse node comes equipped with a number of slots for holding constraint information that is relevant to the parse. Included are person and number, case, determiner (DEFINITE, INDEFINITE, PROPER, etc.), mode (ROOT, FINITE, etc.), and semantic categories. These features are passed along from node to node: from parent to child, child to parent, and left-sibling to right-sibling. Certain nodes have the power to adjust the values of these features. The adjustment may take the form of an unconditional override, or it may involve a unification with the value for that feature passed to the node from a parent, sibling, or child. The filters are restricted in power in two important ways: 1) A filter can only operate on data that are available to the immediate parse node that instantiates the filter, and 2) A filter must be restricted in action to simple logical operations such as AND, SET, RESET, etc.

Some specific examples of constraint implementations will help explain how this works. Certain nodes specify person/number/determiner restrictions which then propagate up to higher levels and back down to later terminal nodes. Thus, for example, A NOUN-PL node sets the number to PLURAL, but only if the left sibling passes to it a description for number that includes PLURAL as a possibility (otherwise it dies, as in "each boats"). This value then propagates up to the SUBJECT node, across to the PREDICATE node, and down to the verb, which then must agree with PLURAL, unless its MODE is marked as non-finite. Any non-auxilliary verb node blocks the transfer of any predecessor person/number information to its right siblings, reflecting the fact that verbs agree in person/number with their subject but not their object.

A more complex example is a compound noun phrase, as in "Both John and Mary have decided to go." Here, each individual noun is singular, but the subject requires the plural form of "have." TINA deals with this by making use of a node category AND-NOUN-PHRASE, which sets the number constraint to PLURAL *for its parents*, and blocks the transfer of number information *to its children*. Some nodes also have special powers to set the mode of the verb either for their children or for their right-siblings. Thus, for example, "have" as an auxilliary verb sets mode to PAST-PARTICIPLE for its *right-siblings.* The category GERUND sets the mode to PRESENT-PARTICIPLE for its *children.* Whenever a PREDICATE node is invoked, the verb's mode has always been set by a predecessor.

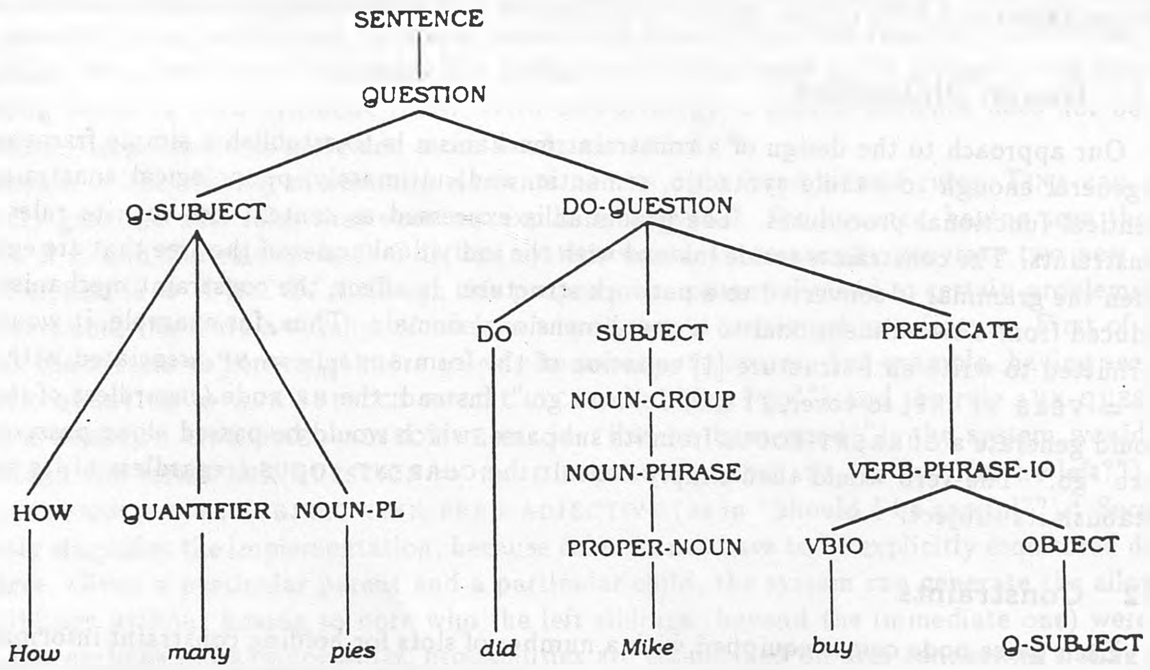

Figure 3: Example of a Parse Tree Illustrating a Gap.

3.3 Gaps

The mechanism to deal with gaps resembles in certain respects the *Hold* register idea of ATN's, but with an important difference, reflecting the design philosophy that no node can have access to information outside of its immediate domain. The process of getting into the *Hold* register (or the FLOAT-OBJECT slot, using my terminology) requires two steps, executed independently by two different nodes. The first node, the *generator*, fills the CURRENT-FOCUS slot with the subparse returned to it by its children. The second node, the *activator*, moves the CURRENT-FOCUS into the FLOAT-OBJECT position, for its children. It also requires that the FLOAT-OBJECT be absorbed somewhere among its descendants by a designated *absorber* node. The CURRENT-FOCUS only gets passed along to siblings and their descendants, and hence is unavailable to activators at higher levels of the parse tree. Finally, certain (*blocker*) nodes block the transfer of the FLOAT-OBJECT to their children.

A simple example will help explain how this works. For the sentence "(How many pies)$_i$ did Mike buy (t_i)?" as illustrated by the parse tree in Figure 3, the Q-SUBJECT "how many pies" is a generator, so it fills the CURRENT-FOCUS with its subparse. The DO-QUESTION is an activator; it moves the CURRENT-FOCUS into the FLOAT-OBJECT position. Finally, the object of "buy," an absorber, takes the Q-SUBJECT, as its subparse. The DO-QUESTION refuses to accept any solutions from its children if the FLOAT-OBJECT has not been absorbed. Thus, the sentence "How many pies did Mike buy the pies?" would be rejected. Furthermore, the same DO-QUESTION node deals with the yes/no question "Did Mike buy the pies?," except in this case there is no CURRENT-FOCUS and hence no gap.

More complicated sentences involving nested or chained traces, are handled staightforwardly by this scheme. For instance, the phrase, "(the violin)$_i$ that (these Sonatas)$_j$ are easy to play (t_j) on (t_i)" can be parsed correctly by TINA, identifying "Sonatas" as the object of "play" and "violin" as the object of "on." This works because the VERB-PHRASE-P-O, an activator, writes over the FLOAT-OBJECT "violin" with the new entry "Sonatas," but only for its children. The original FLOAT-OBJECT is still available to fill the OBJECT slot in the following prepositional phrase.

The example used to illustrate the power of ATN's [6], "John was believed to have been shot," also parses correctly, because the OBJECT node following the verb "believed" acts as both an absorber and a (re)generator. Cases of crossed traces are automatically blocked because the second CURRENT-FOCUS gets moved into the FLOAT-OBJECT position at the time of the second activator, overriding the preexisting FLOAT-OBJECT set up by the earlier activator. The wrong FLOAT-OBJECT is available at the position of the first trace, and the parse dies:

*(Which books)$_i$ did you ask John (where)$_j$ Bill bought (t_i) (t_j)?

The CURRENT-FOCUS slot is not restricted to nodes that represent nouns. Some of the generators are adverbial or adjectival parts-of-speech (POS). An absorber checks for agreement in POS before it can accept the FLOAT-OBJECT as its subparse. As an example, the question, "(How oily)$_i$ do you like your salad dressing (t_i)?" contains a Q-SUBJECT "how oily" that is an adjective. The absorber PRED-ADJECTIVE accepts the available float-object as its subparse, but only after confirming that POS is ADJECTIVE.

The CURRENT-FOCUS has a number of other uses besides its role in movement. It always contains the subject whenever a verb is proposed, including verbs that are predicative objects of another verb, as in "I want to go to China." In the case of passive voice, it contains 'NIL at the time of instantiation of the verb. It has also been found to be very effective for passing semantic information to be constrained by a future node, and it plays an integral role in pronoun-reference. These issues are addressed more fully in [4]

3.4 Semantic Filtering

In the most recent version of the parser, we implemented a number of semantic constraints using procedures that were very similar to those used for syntactic constraints. We found it effective to filter on the ACTIVE-NOUN's semantic category, as well as to constrain absorbers in the gap mechanism to require a match on semantics before they could accept a FLOAT-OBJECT. Semantic categories were implemented in a hierarchy such that, for example, RESTAURANT automatically inherits the more general properties BUILDING and PLACE. We also introduced semantically-loaded categories at the low levels of the parse tree. It seems that, as in syntax, there is a trade-off between the number of unique node-types and the number of constraint filtering operations. At low levels of the parse tree it seems more efficient to label the categories, whereas information that must pass through higher levels of the hierarchy is better done through constraint filters.

4 Practical Issues

Two unique practical aspects of TINA's design are its generation-mode capability and its ability to build a grammar automatically from a set of parsable sentences. We have found generation mode to be an essential tool for identifying overgeneration problems in the grammar. The ability to automatically provide a subset grammar for a set of sentences makes it easy to design a very specific, well constrained grammar, leading to improved performance in restricted-domain spoken language tasks.

Generation mode uses the same low-level routines as those used by the parser, but chooses only a single path based on the outcome of a random-number generator. Since all of the arcs have assigned probabilities, the parse tree is traversed by generating a random number at each node and deciding which arc to take based on the outcome, using the arc probabilities to weight the alternatives. Occasionally, the generator chooses a path which leads to a dead end, due to unanticipated constraints. In this case, it can back up and try again. Table 1 contains five examples of consecutively generated sentences. Since these were not selectively drawn from a larger set, they accurately reflect the current performance level. Because a number of semantic filtering operations have been applied within this task, most of the generated sentences are semantically as well as syntactically sound.

It is a two-step procedure to acquire a grammar from a specific set of sentences. The rule set is first built up gradually, by parsing the sentences one-by-one, adding rules and/or constraints as needed. Once a full set of sentences has been parsed in this fashion, the parse trees from the sentences are automatically converted to the set of rules used to parse each sentence. The training of both the rule set and the probability assignments is established directly from the provided set of parsed sentences; i.e. the parsed sentences *are* the grammar.

Another useful feature of TINA is that, as in LFG's, all unifications are nondestructive, and as a consequence explicit back-tracking is never necessary. Every hypothesis on the queue is independent of every other one, in the sense that activities performed by pursuing one lead do not disturb the other active nodes. This feature makes TINA an excellent candidate for parallel implementation. The control strategy would simply ship off the most probable node to an available processor.

Table 1: Sample sentences generated consecutively by the most recent version of TINA.

Do you know the most direct route to Broadway Avenue from here?
Can I get Chinese cuisine at Legal's?
I would like to walk to the subway stop from any hospital.
Locate a T-stop in Inman Square.
What kind of restaurant is located around Mount Auburn in Kendall Square of East Cambridge?

5 Discussion

This paper describes a new grammar formalism that addresses issues of concern in building a fully integrated speech understanding system. The grammar includes arc probabilities reflecting the frequency of occurrence of the syntactic structures within the domain. These probabilities are used to control the order in which hypotheses are considered, and are trained automatically from a set of parsed sentences, which makes it straightforward to tailor the grammar to a particular need. Ultimately, one could imagine the existence of a very large grammar that could parse almost anything, which would be subsetted for a particular task by simply providing it with a set of example sentences within that task.

I believe that, at the time a set of word candidates is proposed to the acoustic matcher of a recognizer, all of the constraint available from the restrictive influence of syntax, semantics, and phonology should have already been applied. The parse tree of TINA can be used to express various constraints ranging from acoustic-phonetic to semantic and pragmatic. Each parse node would contain slots for all kinds of constraint information – syntactic filters such as person, number and mode, semantic filters such as the permissible semantic categories for the subject/object of the hypothesized verb, and acoustic-phonetic filters (for instance, restricting the word to begin with a vowel if the preceding word ended in a flap, as in "Wha*t* is"). As the parse tree advances, it accumulates additional constraint filters that further restrict the number of possible next-word candidates. Thus the task of the predictive component is formulated as follows: given a sequence of words that has been interpreted to the fullest capability of the syntactic/semantic/phonological components, what are the likely words to follow, and what are their associated a priori probabilities?

While TINA's terminal nodes are lexical words, I believe that the nodes should continue down below the word level. Prefixes and suffixes alter the meaning/part-of-speech in predictable ways, and therefore should be represented as separate subword grammar units that can take certain specified actions. Below this level would be syllabic units, whose children are subsyllabic units such as onset and rhyme, finally terminating in phoneme-like units. Acoustic evidence would enter at several stages. Important spectral matches would take place at the terminal nodes, but duration and intonation patterns would contribute to scores at many higher levels of the hierarchy.

Three different task-specific versions of TINA have been implemented. The first one was designed to handle the 450 "phonetically rich" sentences of the TIMIT database [2]. The system was then ported to the DARPA Resource Management domain. A number of evaluation measures have been applied for these tasks, as described in [3]. Little else will be said here, except to note that perplexity was reduced nine-fold for the Resource Management task when arc probabilities established from the training data were incorporated, instead of using the equal-probability scheme. The latest version has been tailored to the new VOYAGER task, under development at MIT. This task involves navigational assistance within a geographical region. Our goal is to utilize constraints offered by both syntax and semantics so as to reduce perplexity as much as possible without sacrificing coverage. The parser is implemented on the Symbolics Lisp machine and runs quite efficiently. A sentence, entered in text form, is typically processed in a fraction of a second.

An effort to integrate the VOYAGER version of TINA with the SUMMIT speech recognition system [7] is currently underway. Two important issues are 1) how to combine the scores for the recognition component and the predictive component of the grammar, and 2) how to take advantage of appropriate pruning strategies to prevent an explosive search problem. The fully integrated spoken language system will use TINA both to constrain the recognition space and to provide an input to the back-end. Our current approach is to link together all words and all start-times that are equivalent within the parse, letting them proceed at a pace in accordance with the best-scoring word/time for the set. Viterbi pruning can take place within the recognizer, by having each terminal node initialize the recognizer with all the active phonetic nodes provided by its set of active hypotheses.

6 Acknowledgements

In developing this parser, I have benefitted from interactions with Lynette Hirschman, Mark Johnston, and Victor Zue. This research was supported by DARPA under Contract N00014-89-j-1332, monitored through the Office of Naval Research.

References

[1] Bresnan, J., ed., *The Mental Representation of Grammatical Relations*, MIT Press, 1982.

[2] Lamel, L., R.H. Kassel, and S. Seneff, "Speech Database Development: Design and Analysis of the Acoustic-Phonetic Corpus," DARPA Speech Recognition Workshop Proceedings, Palo Alto, CA, Feb 19-20, 1986.

[3] Seneff, S. "TINA: A Probablistic Syntactic Parser for Speech Understanding Systems," *Darpa Speech and Natural Language Workshop Proceedings*, Feb.1989.

[4] Seneff, S. "TINA: A Probablistic Syntactic Parser for Speech Understanding Systems," Laboratory for Computer Science Technical Report, forthcoming.

[5] Tomita, M., *Efficient Parsing for Natural Language*, Kluwer Academic Publishers, Boston, MA, 1986.

[6] Woods, W.A., "Transition Network Grammars for Natural Language Analysis," Commun. of the ACM 13, 591-606, 1970.

[7] Zue, V., J. Glass, M. Phillips, and S. Seneff, "The MIT Summit Speech Recognition System: A Progress Report," *DARPA Speech and Natural Language Workshop Proceedings*, Feb.1989.