Natural Language as a Robust Recovery Mechanism for Speech Recognition Errors

Greg Nelson

Introduction

In the field of speech recognition, there has been a long-standing competition between two different models of language. Early systems relied on a symbolic approach to language modeling, applying principles of linguistics to the problem of evaluating what combinations of words were valid or meaningful. These systems often used strict parsing frameworks such as LR and LALR to characterize the language. Later, the field developed statistical language models, whose evaluations rely on the statistical correlation between adjacent or proximal words. Although a majority of systems today favor statistical language models (LMs), each side has its advantages and disadvantages, and neither has clearly lost the war. Between the sounds of speech and the words being evaluated lies some sort of acoustic model. With the exception of the very earliest systems, the acoustic model (AM) used is usually stochastic in nature, whatever the underlying pattern matching technique might be.

The popularity of statistical LMs comes from two of their important advantages. First, the methodology is better aligned to the successful AM technologies, and the output from the two models can be readily combined to yield some notion of probability. Second, and more central to my research, is the robustness of the technique. No matter how improbable an utterance may be, it will always be assigned some score. If an unlikely utterance appears sufficiently certain to the AM, it may even be deemed the best parse. Although statistical language models do help speech recognition performance (refs?), they have two difficulties that we must overcome. One lies in the statistics themselves: they are only as good as the match between the training and the test set and often lack the necessary coverage. The other is the computational and storage cost inherent in the technique: once you have amassed an adequate training corpus, even the compiled probabilities may take more memory than one would find in a typical user's computer.

Symbolic language models have a contrasting set of advantages. They are built from a set of rules, typically written by hand as an implementation of a particular linguistic theory. These rules implicitly describe a fixed set of "correct" utterances. Though there is some engineering effort involved, the parser can be made small and fast, and can readily be generalized to new situations. The model can often be extended to cover new domains by the addition of a set of new vocabulary items to the lexicon, without significant changes to the parsing rules. Unfortunately, because the underlying linguistic theory is necessarily incomplete, the grammar will always be unable to parse some portion of utterances that human speakers would accept as correct, and often it will also accept some utterances we would reject. This makes symbolic models fragile in the face of input that does not conform to their expectations or rules.

The fundamental inability of symbolic models to handle the complexities of real spoken language has largely discouraged the speech community's use of this approach. Recently, however, progress has been made in the development of robust parsers such as GLR* (?) (and others?) that add flexibility to parsing. These parsers try to account for extragrammatical utterances by positing deviations (typically insertions and deletions) from an underlying grammatical input. Although the addition of deviations undoubtedly improves upon the rigid parsing of LR and LALR models, it does not eliminate the problem. The parser is only relaxing a set of rules that were never adequate in the first place, and it is throwing out potentially important information in the process.

I propose to build a symbolic parser that more accurately captures the regularities of both the language and the errors. My two primary goals are to detect and correct errors whenever possible, and to make progress toward a reasonable interpretation even in the event of a parsing failure. There are several capabilities I intend to develop to achieve these goals. The parser must be able to detect errors and determine their source. It should be able to correct errors in various ways, including the ability to interface directly with and provide feedback to the speech system's search process. This allows the parser to explore the hypothesis that a particular parsing difficulty is simply a recognition error. The parser should back off from the rigid grammar in controlled ways, using mechanisms such as type coercion and semantic inheritance before regarding an utterance as unparseable. Likewise, the parser should use as many knowledge sources as possible (including syntax, semantics, and discourse) to provide redundancy if one source fails to parse an utterance. If all else fails and no complete parse is possible, the parser must extract as much coherent semantic information as possible for later task-specific processing. Finally, the parser should be able to do this processing efficiently, especially for high frequency inputs, whether correct or erroneous.

Previous Work

The first generation of speech recognition systems (refs?) relied on symbolic knowledge for both language and acoustic modeling. However, experience with these systems suggested that their limited grammars rarely scaled to larger domains. In later systems (ref Hearsay, and whoever else?), stochastic models proved to be a more effective and powerful way of handling the acoustic phenomena. The symbolic LMs, however, did not incorporate a notion of likelihood that could be integrated into the stochastic search techniques being used at the level of the acoustics. The introduction of statistically based "N-gram" language models (LMs) solved both of these problems, but introduced a number of new ones.

A typical architecture for a modern continuous speech recognition system uses hidden Markov models (HMMs) to evaluate the probabilities of particular word strings accounting for the observed acoustics. These probabilities are combined with those produced by the language model in a larger HMM which is evaluated through a dynamic programming search, using heuristics to limit the cost of the search. The most common type of LM in use today is a trigram LM with some sort of "backoff" mechanism. (ref Sphinx, Sphinx-II, Dragon, HTK, ...?) In this model, many words are represented as having a particular probability *in* *the context of two other words.* In cases where this trigram (three words and a probability) is not available, there is a mechanism for substituting ("backing off" to) a bigram probability, which only uses one word of context. Finally, if this fails, it is possible to back off to a unigram, which is essentially the base probability of the word's occurrence. All of the probabilities are determined by frequency estimates gathered from a large corpus of data in the appropriate domain.

Coverage is more easily extended to larger domains than it was with symbolic models. Rather than tedious knowledge engineering by human programmers, the generation of a new language model becomes a simple matter of running the training data through the computer software that builds the LM. Furthermore, moving to a slightly different domain may be possible as long as the words are present in the LM and their probabilities are not radically different. The first problem we encounter, though, lies in the acquisition of the training data itself. Adequate training for a large vocabulary task requires millions (?) of words of text. And unfortunately, what may seem like a minor domain difference to a human language user (the difference between written financial reports and ones broadcast over the radio or the television, for example) may in fact be large enough to seriously degrade LM performance.

For some domains, it is hard to find an exact match in a form that is readily available for machine processing: although we can get the text of the Wall Street Journal or of ClariNet electronically, it is still costly to get accurate transcriptions of radio or television broadcasts. For other domains, no large corpus of any sort is readily available. How does an American company or research group get a million words of machine-readable Serbo-Croatian? Finally, even the best training corpus yields a LM that is static in nature. A LM built from the news of a particular month or year would reflect the current events of that time, and would do a poor job estimating probabilities for many parts of today's news. Techniques for adjusting or adapting the language model "on the fly" as a function of the context, which would help with these difficulties, are still open research issues (refs? Roni?).

Another major difficulty lies in the computational and storage costs of the models. A trigram LM in a relatively limited domain (such as ATIS, the Air Travel Information System) can require tens of megabytes of storage. A comparable bigram-only model might take only a tenth of that amount of space. To run efficiently with current speech decoder technology, much if not all of this data must reside in physical memory rather than on disk. These constraints (combined with the finite quantities of available training data) prevent us from realistically using contexts larger than trigrams. Yet many regularities in language, such as subject-verb and verb-object agreement occur over longer distances than this.[1] Furthermore, much of the computational cost of the search performed in a typical continuous speech recognition system can be shown to lie in the cost of branching out from a possible word endpoint to a potentially huge set of following words. Here even the cost with just unigrams is high, but the addition of bigrams compounds the problem immensely.

Using symbolic knowledge instead of (or in addition to) statistical knowledge, we can

[1] Most of us would cringe at "The horse with the white spot win the race," or, "I persuaded my friend the issue." But **persuaded** and **the** are four words apart, and **horse** and **win** would require a 6-gram language model to relate them. A trigram LM could never detect these errors, much less help to correct them.

begin to describe patterns of usage that vary in the details of the actual words and the number of words of separation. One partially symbolic approach is the use of class N-grams (refs?), which can help capture some generalizations by placing groups of words with similar linguistic behaviors into classes. These classes may be lingustically based, using features like part of speech (ref Roni?) or statistically based, using clustering techniques such as maximum entropy (ref Lafferty?). Many of the problems cited earlier remain. The statistical techniques require large volumes of data that must resemble the desired domain. The linguistic approaches rely on a set of fixed, predefined classes, and furthermore require a way to place on the training data class tags which are reliable in the face of polysemy and other distractors.

Thus, although there is much promising research currently being done with statistical LMs, a successful robust symbolic parser would present an alternative means of addressing a number of important problems. In the next section, I will describe the techniques I plan to apply in building such a parser. Along the way I will point out some difficulties with earlier methodologies and show how I plan to address them. Some of the proposals will be supported by examples from a prototype of the robust parser that I have built within the framework of NL-Soar.

Goals and Issues

As I stated before, there are two key goals for the parser. First, it should detect and correct recognition and parsing errors whenever possible. Second, we would like it to make progress towards a reasonable interpretation even in those situations it cannot correct. Unlike the heuristic of minimal edit distance used in some earlier robust parsers, my approach favors an explicit decision between correcting the speech input and proceeding after a parsing failure. As we will see, the potential independence of the choice to correct and the choice to ignore a parsing problem will force a tradeoff in what may be deemed sensitivity and specificity. For example, making error correct so general that correct inputs are corrupted will inhibit our ability to get a valid interpretation of these inputs. In contrast, allowing potentially correctable recognition errors to be treated a parsing failures misses the opportunity to get a complete, correct interpretation. I will present a number of these issues and describe a proposed mechanism to tackle each one. Although some of the individual solutions may not work in practice, the goal of this thesis is to explore a constellation of techniques and demonstrate a set that do improve the robustness of a running system.

The first step in addressing an error is its detection. As soon as we have found a problem, we face one of the familar but difficult problems in speech recognition: the assignment of blame or responsibility for the error. The type of correction we should apply depends on the source of the error. Some recent research has looked at the problem of determining which recognition components are responsible for various error classes or even individual errors. Chase (1996?) has investigated the distinction between LM and acoustic model (AM) failures and even studied which sub-phonetic features (duration, voicing, etc.) are associated with the highest error rates. We are not changing the recognition system, and so are not concerned with what part of the recognizer is responsible; instead we would like to be able to determine whether the error was made by the speaker or the recognizer (*early* errors), or the grammar (*late* errors).

Early errors fall into two categories: errors made by human speakers and errors introduced by the AM in the recognizer. These types overlap, but each set also has some unique representatives. For example, restarts by speakers tend to include content words: "I think... uh... thought I had a good example." In contrast, a number of recognizer problems produce illegal combinations of short, high-frequency, closed-class words, including determiners and prepositions: "Yes of a in has the map." If we take the human speaker's words as the ultimate truth, we arrive at the counterintuitive conclusion that the apparently erroneous input from the recognizer can't be made any more accurate, even if it could be made more parsable. On the other hand, if we are interested in acquiring as much semantic information as possible, a more parsable version of the erroneous input may be useful, even if it is not what the speaker said. So, depending on the goals of the system, the value in being able to distinguish different kinds of early errors may vary.

What it more important is to distinguish early errors from late errors. A late error is typically a case of overspecificity or overgenerality of the parser's rules, rather than a fault of the speech recognizer. Parser errors are certainly easier for the system builder to correct than the subtle errors that come from variations in the human voice, but it is difficult to make the parser correct itself. There are some possibilities, however. If overspecific rules give us parsing failures, we can force the system to search for generalizations.

In some cases, even a well formed utterance is inappropriate to the situation at hand, violating "pragmatic" constraints. Although this may not show up as a parsing breakdown, this is another form of error. Separating cases in which the speech recognition system has failed and produced an anomalous utterance from cases in which the speaker has strayed from the topic or interjected something unexpected requires substantially more sophistication than is available from most parsers. If it is possible to isolate a few common classes of anomalous utterances, it would be valuable to build mechanisms that can detect these anomalies and categorize them. The system can then make an informed decision about the relative likelihood of the anomaly arising from the speaker or the speech recognition process.

We have already addressed some of the kinds of error correction that are possible. The most powerful and general mechanism for dealing with early errors is a mechanism by which the parser can request alternative inputs from the recognizer. I have developed a general version of this which will allow for a variety of close interactions between the two systems. Two important features of the interface design are its generality for a variety of speech systems, and its relative similarity to the recognizer's own search process. The second feature ensures the possibility of moving the parsing technology inside the recognizer once its success has been demonstrated.

To correct late errors, a number of standard techniques are available. Many utterances will fail on syntactic grounds but be recoverable on semantic grounds. When semantic parsing fails, some problems may be solvable through semantic inheritance – knowing that a something is a member of a class for which a semantic relation would succeed, we can posit that the relation should succeed for the class member as well. Both syntax and semantics can fall back on certain type coercions which generalize the fixed rules in particular ways. A more sophisticated late correction that may be needed is the automatic extension of the lexicon. When there are mismatches between the recognition lexicon and the parsing lexicon, either one may be at fault. Extra words in the parsing lexicon pose no problem, but how does the parser handle an unknown word the recognizer produces? When our system has a clear sense of what it is expecting, it can attempt to process the word appropriately. If, after all of these options have been explored, there is still no adequate parse, the system must gracefully move on while retaining as much of the the semantic information as possible for later use.

In the end, it is important that the system perform all of these tasks efficiently. The system clearly must spend time on every word, taking more time for longer inputs. Optimally, we would like the system to run in time proportional to the real length of the spoken utterance. However, if we can show that the system takes no more than a constant time for every word considered, and we can limit the number of words we consider adequately, we will be doing well. Obviously, it would also be nice if the system ran *fast*, but this is an implementational detail and not a priority of this research.

The two domains I have looked at so far are the ATIS (Air Travel Information System) domain and the bidirectional, machine-assisted speech-to-speech translation of English and Serbo-Croatian in the "Bosnia" domain. The latter domain is particularly interesting for a number of reasons. It shows the potential difficulty of acquiring a large body of training text; even though I am working with the English language side, there is an obvious isomorphism with the Serbo-Croatian side, for which appropriate corpora will be difficult to construct. It is also a domain in which we expect to encounter out-of-vocabulary words (such as names of individuals), which will prove to be an interesting challenge for the speech system. Furthermore, generating a valid translation will require some level of semantic processing, so our efforts to this end are not wasted, even if there is no error in a given utterance.

Plans and Preliminary Results

In this section, I will give an overview of specific plans for achieving the goals I have set out. Along the way, I will give some examples and details based on a prototype system that I have implemented using the Sphinx-II recognizer and the NL-Soar parser. Each of these systems is a research project in its own right, and will certainly come with its own set of issues and unsolved problems. This thesis is not intended to fix those problems, *per se*. For the most part, the unsolved problems in Sphinx will remain that way. But because the parser is the focus of this research, it is more likely that progress made by this thesis will involve lasting improvements to NL-Soar. The overall architecture of the system is shown in Figure 1.

The speech recognition system I am using is Sphinx-II (refs?), which is much like the prototypical system described earlier. It uses vector quantization and semi-continuous hidden Markov models to evaluate a set of triphone acoustic models. The system is typically used with a trigram backoff language model. Like almost every powerful continuous speech recog-

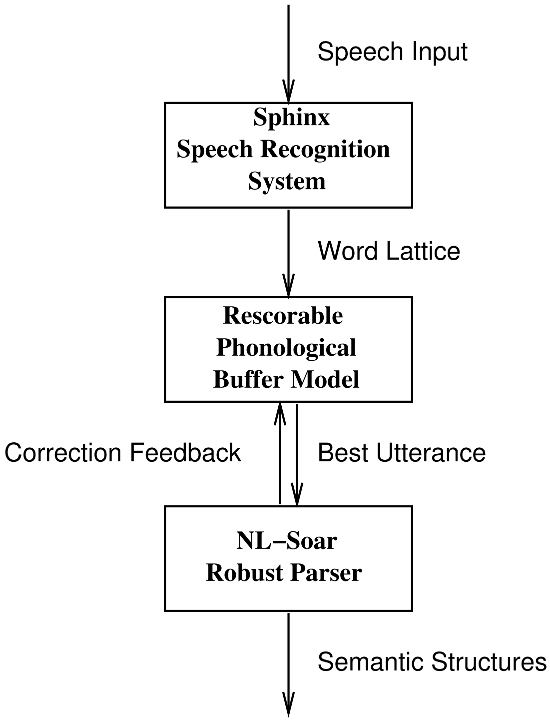

Figure 1: A schematic of the overall architecture of the thesis system.

nition system it has at its core some sort of search, in this case a Viterbi beam search, over a lattice of possible interpretations of the input. Scores for the words in this lattice represent combined acoustic and language model information. The highest scoring path through this lattice of words is what we usually see as the "final" output of the system, but it is possible to extract the lattice itself. This lattice represents not only the best or most likely interpretation of the input, but other good alternatives that were considered. A common technique for improving recognition performance is to run a more sophisticated rescoring pass over this lattice, and in essence this is exactly what we wish to do. In principle, any technique which proved fruitful in this simple form could be moved inside of the recognizer and into the search process at a later point. A key feature of this lattice is that the recognizer is adjusted to ensure a very high likelihood that the correct utterance falls within the lattice, even if it is not selected as the best path. We see this in the fact that the lattice word-error rate or LWER (the number of correct words that fail to appear anywhere in the lattice) is kept very low. (Number, ref?)

The parser we use [Description of NL-Soar] (Lewis, 1993; Lehman, Lewis, & Newell, 1991; Rubinoff & Lehman, 1994).

Between the two system lies and interface I call the *Rescorable Phonological Buffer*, or RPB. Although the implementation I have built is designed to work with the Sphinx-II recognition system[2] it relies on data structures that should be available from other systems as well. The lattice is the primary input to the RPB model, and its output is a single "best"

[2]Actually, with two different incarnations of Sphinx-II known as FBS6 and FBS8.

interpretation of its inputs. Thus the RPB aligns the output data structures of Sphinx with the input structures of NL-Soar. In addition to the lattice, the other inputs to the RPB include a dictionary and a static language model, as shown in Figure 2. Here the dictionary provides a mapping from word IDs to actual lexemes, and the language model scores (which may not be present in the lattice, depending on the recognition system) can be recovered from the stored language model tables. A small amount of additional knowledge, and some general algorithms, are applied to handle such tasks as mapping multiple lexemes (sometimes represented by a single word ID in Sphinx) into separate tokens.

Along the way, it performs a number of services. First, because NL-Soar expects a single utterance rather than a lattice, it must select a particular path to present. Second, NL-Soar is expecting to hear and process the utterances in the order they are spoken – by analogy to the written page, we say "left-to-right." Wit the RPB interface, the speech recognition system need not have the same left-to-right temporal conception as NL-Soar. At any given point in time (characterized by the number of frames of speech sound already processed) NL-Soar sees a segment of a particular path through the lattice as its input. The extent of this segment is limited (as it was in earlier NL-Soar models) by the *capacity* of the phonological buffer, which holds approximately two seconds worth of speech. Interactions between NL-Soar and this buffer allow feedback from higher level language representations to guide the search process.

In this model, the connection between the two systems occurs at the word level. The current path through the lattice is selected on the basis of per-word acoustic scores, language model scores[3], and rescoring information. If we present no feedback from NL-Soar, we would expect to see and process only the best path through the lattice. However, we may present feedback at any point which will change the relative rankings. If we know that the right word is present in the lattice (because the LWER is low), and we push the wrong word out of the way, we have a good chance of getting the right word to take its place.

From the perspective of the natural language system, the rescorable phonological buffer is neither a lattice nor a single, simple string. Instead, it is a considered a pool of potential words. At any particular time, only one interpretation for a segment of acoustic input is available to the language processing facilities of NL-Soar. However, NL-Soar has feedback mechanisms available that can be used to modify what this segment appears to be. The interpretation presented combines the speech system's acoustic and language model scores with scores passed back from the NL-Soar system, which represent a combination of syntactic, semantic, and pragmatic constraints. On the basis of each of these factors, or a combination of them, NL-Soar can post a request to the phonological buffer to raise or lower the score of a word within a particular time frame, or possibly to reject it outright.

Turning to our attention to our goals and the mechanisms needed to implement them, we look again at error detection. This is one of the simplest components of the system, because we primarily rely on the presence of a parsing failure to signal an error. It should also be possible, as mentioned before, to use pragmatic knowledge to detect certain problems that

[3] In the current model, these are unigram-only scores, but the methodology could be applied equally well to the complete trigram model.

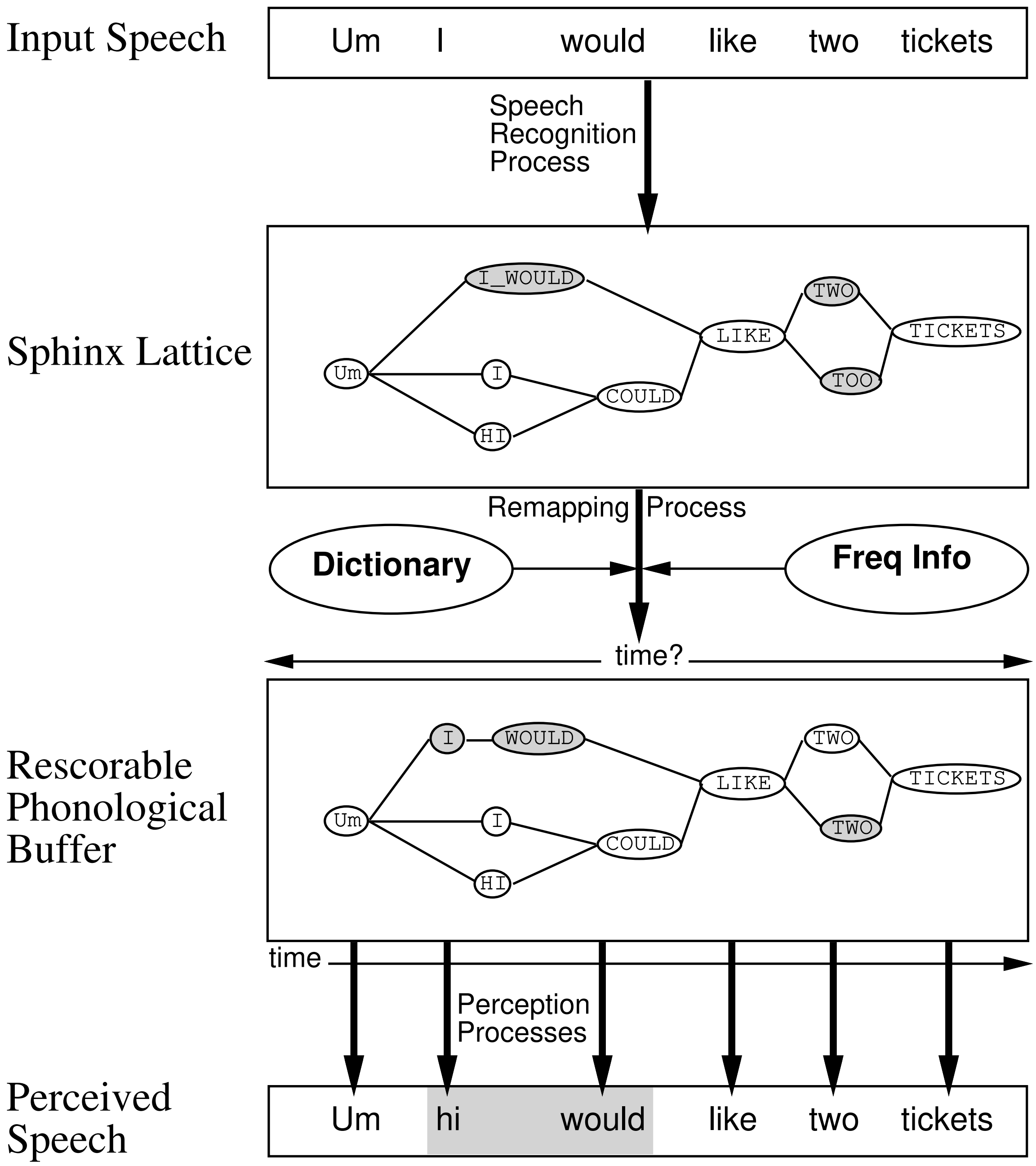

Figure 2: A diagram of the data sources and mapping processes of the Rescorable Phonological Buffer.

```
PROPOSE-CORRECTION*DETERMINER-PAIR

If the most recent word is unattached and it is a determiner
   and the word preceding it is unattached and it is also a determiner,
Then propose the deletion of the second determiner as a possible action.
```

Figure 3: A rule used to detect "determiner pair" errors.

do not result in parsing failures, but I have not implemented this capability yet. Although the detection of the error may be simple, the complexity of blame assignment makes up for this.

Knowing that we have an error, we must determine its type. Is it an early or a late error? If it is an early error, is it one we believe came from the recognition system, or are we trying to "correct" the speaker to achieve a more complete representation? If it is a late error, is it one we can correct, or one we must accept and work around? And if we cannot determine the source of the error, what default action will we take? I argue that we must actively detect different kinds of errors in those cases where we have sufficient knowledge. We would like to avoid performing a default action whenever possible, since this is could easily be an inappropriate response to the error.

Figure 3 shows one of the rules used in the prototype system to detect a particular type of early error ("determiner pairs") that we have seen occur with some frequency as a recognition error. We know that an adjacent pair of determiners will not be parseable in our system; we believe that this error is unlikely to be generated by a human speaker. (Obviously this rule is overgeneral – I might *want* to discuss the **the** I deleted!) This rule proposes one way of dealing with the problem, which for most domains is the only one necessary. If we were pursuing a domain in which the ability to refer to determiners as words was important, we would see other alternatives for parsing this utterance, and more specific knowledge, referring to other knowledge sources, would be used to chose between the alternatives.

It is also important that the system be aware of ways in which its grammar is incomplete. In NL-Soar, because we are parsing from left to right using a tree-structured syntax, there are times when we will not have enough information to incorporate the current word into the syntax tree. The same principle of active detection of errors can apply to these late errors, to prevent us from passing the blame on to one of the earlier components. For example, Figure 4 shows a rule (somewhat similar to the previous one) that recognizes a specific inadequacy of the syntactic model we are using. When a determiner arrives, the word for which it will eventually act as a specifier cannot have arrived yet, and we will be left with an apparently disconnected utterance. Knowing that this is expected, we can postpone any error corrections until we have had the opportunity to survey more of the situation.

Once we isolate a problem as a recognition error, we need to determine how to correct it. What exactly do we mean by "correcting" an error? Specifically, the system uses feedback to the speech recognizer to explore alternative hypotheses when it detects a recognition error.

```
PROPOSE-EMPTY-OP*DETERMINER

If the most recent word is a determiner
   and the word preceding it was not a determiner,
       or there was no previous word,
Then propose waiting for the next word as a possible action.
```

Figure 4: A rule used postpone processing of dangling determiners.

Looking back at Figure 2, we can now be more specific about how this correction occurs, and what the term "deletion" in Figure 3 means. Deletion amounts to lowering the score of a particular word by a sufficient amount to drive that hypothesis below the next-best hypothesis (or perhaps below the next N hypotheses). If we recognized the perceived word **hi** as an error (by some other active rule, or by defaulting to error correction), deleting it would make the correct alternative, **I**, rise to the top.

Of course, NL-Soar does not know what alternative will present itself, or whether that alternative itself is correct; the mechanism simply allows it to discard one hypothesis to find out what the next one would be. The other mechanism implemented in the prototype is "competition," which allows the deletion of the least certain member of a set of words. This might be appropriate for a circumstance like an agreement failure, where NL-Soar knows one of the words is incorrect but is uncertain of which one. Since the scores in Sphinx are intended to represent log probabilities, the word least differentiated from the next alternative is the one most likely to be in error.[4]

A more general solution is to make it possible to extrapolate reasonable behavior in cases where the speech recognition lexicon is more complete than the NLP lexicon. I intend to modify NL-Soar so that it is able to detect when a word arrives from the speech recognition system that is not in its lexicon. Then, either through knowledge about the prior context, or by taking this in combination with external knowledge sources (such as WordNet (Miller, Beckwith, Fellbaum, Gross, & Miller, 1990)), NL-Soar can hypothesize a syntactic and semantic role for the word and continue processing. After the word has been encountered the first time, the newly hypothesized meaning can be compiled into the system so that it can be accessed quickly when it occurs again.

Salvaging information:

The next important feature is the decision to extract as much information as possible from sentences that are either syntactically or semantically ill-formed. Many traditional parsers (GLR*? Link Grammars? other refs?) treat syntactic validity as a precondition for

[4]If we look at alternatives A and B for the first word and C and D for the second, with $P(A) > P(B)$ and $P(C) > P(D)$, the fact that $log(P(A)) - log(P(B)) < log(P(C)) - log(P(D))$ is equivalent to $P(A) * P(D) < P(B) * P(C)$. In this case, the word least differentiated from its next alternative (A) should be replaced by the alternative (B) to maximize the probability of the new string. `Should this formalization go into the body rather than into a footnote?`

any other processing. Although some parsers are capable of modeling ungrammaticalities as insertions and deletions, these parsers simply give up information in this process. NL-Soar processes input for both syntax and semantics, and neither one is *in principle* a prerequisite for the other. Thus, because semantics may still be able to make a meaningful commitment about an input utterance when syntax fails, the system can often continue to evaluate the validity of the input in circumstances where another system would merely be able to score it as "less grammatical by degree N".

In the current implementation, the system builds syntactic empty-ops before it begins to search for semantic knowledge. My proposed method to bring semantics to the foreground in this situation is to modify the system so that it searches the semantic knowledge before it builds the empty-op. This way, if a semantic connection can be made, the system can detect this before it marks the utterance as ill-formed. By involving the semantic component at this early stage, later processing can be constrained in ways that would not be possible if entire segments of input were ignored *in toto*.

Efficiency:

Methodology and Schedule

I plan to pursue a multi-strategy approach in this research. Each of the problems I am able to solve will constitute a small amount of progress in improving the robustness of linguistic models for speech recognition tasks. However, because some of these strategies explore completely new territory, I may discover that some may be too difficult to undertake at this point. Therefore the goal is to make a small contribution with as many strategies as possible, rather than to explore each one to exhaustion.

Generality across speech systems is an important issue, and one that I would like to address, but I intend to postpone this work and perform it as time allows. A large degree of generality is implicit in the decision to operate on word lattices, because a great number of speech recognition systems already produce this form of output. The primary generality question then becomes that of whether different systems make similar recognition errors. To the degree that this research attempts to repair problems that arise from a particular acoustic model, it is important to explore this issue. However, there is a belief in the speech community that, although the particular errors may differ across systems, many of the same types of errors occur.

There are a number of related research issues which I do not plan to address, not least because many of them would constitute theses in and of themselves. First of all, a great deal of work has been done in psychology and linguistics (e.g. Clark, others?) to characterize the kinds of errors that humans do and don't make in verbal communication. Although this may be a helpful background in trying to get the system to predict which errors are made by the speaker and which are acoustic mismatches, and I may use existing work in this area, I do not plan to engage in further studies of this question. Likewise, techniques for integrating constraints across syntactic, semantic, and pragmatic knowledge sources are an interesting research issue themselves. I expect to make some limited contributions to this area, but the goal of the thesis is not to generate an inclusive theory of these interactions.

There are general issues of robust NLP that have been addressed elsewhere. I intend to focus on several problems that are specific to speech and to rule-based (rather than stochastic) models of language. I believe this makes the work somewhat unique, and I expect that though the problems and solutions will be of general interest, they are sufficiently focused to be both novel and within the scope of a reasonable research topic.

Finally, there are a few other aspects which would be interesting to explore if time allows. Since I am working in the translation domain, it would be nice if it were possible to demonstrate the translation capability within the same system. In fact, Deryle Lonsdale is in the process of developing a model of human simultaneous interpretation for his thesis, which is also based on NL-Soar. If his research is mature near the end of my thesis work, connecting the two systems would provide the translation capability, as well as demonstrating the generality of the NL-Soar core. Julie Van Dyke has also been working within NL-Soar, looking at the Polish language, which shares a fair amount of grammar with Serbo-Croatian. It would be interesting to use her work to bootstrap a Serbo-Croatian version of the robust parser, if sufficient time and expertise were available.

Significance

The goal of this thesis is to develop a robust symbolic model of spoken language for use with speech recognition systems. This parser will take advantage of detectable differences between errors made by human speakers, those introduced by speech recognition, and those resulting from its own lack of knowledge to decide what action to take in the face of an error. It will apply syntactic, semantic, and discourse knowledge in a partially independent manner, using this redundancy compensate for the inadequacies of the individual knowledge sources. It will interact with and apply rescoring to a generalized word lattice interface that will make it compatible with a variety of speech recognition systems, and leave open the possibility of moving the parser inside the recognizer.

One of the most important contributions of this thesis will be to make symbolic language models a viable alternative to stochastic ones, at least within certain limited circumstances. With domains for speech recognition becoming more complex all the time, there is growing concern within the speech community about the ability of static stochastic language models to capture the necessary linguistic phenomena. Although there is extensive ongoing research on how to extend stochastic language modeling, this work seeks to overcome some of the critical shortcomings of symbolic language models, and thus to open the door to a promising alternative.

References

Lehman, J. F., Lewis, R. L., & Newell, A. (1991). Natural language comprehension in Soar: Spring 1991. Tech. rep. CMU-CS-91-117, Carnegie Mellon University Computer Science.

Lewis, R. L. (1993). *An Architecturally-based Theory of Human Sentence Comprehension.* Ph.D. thesis, Carnegie Mellon University. Also available as Technical Report CMU-CS-93-226.

Miller, G., Beckwith, R., Fellbaum, C., Gross, G., & Miller, K. (1990). Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, *3*, 235–244.

Rubinoff, R., & Lehman, J. F. (1994). Real-time natural language generation in NL-Soar. In *7th International Workshop on Natural Language Generation, Kennebunkport, Maine.*