Using Natural-Language Knowledge Sources in Speech Recognition

Robert C. Moore*
Microsoft Research
Redmond, Washington

1 Introduction

At the current state of the art, high-accuracy speech recognition with moderate to large vocabularies (hundreds to tens of thousands of words) requires a language model. That is, in addition to incorporating a model of the correspondence between acoustic signals and words, a recognition system needs to incorporate a model of what sequences of words are possible, or at least likely. Typically, the language models used in speech recognition systems pay little attention to the linguistic structure of utterances. Perhaps the most common language models used for speech recognition incorporate only n-gram statistics, looking at a sliding two-word (bigram) or three-word (trigram) window to judge the likelihood of a recognition hypothesis. Models of this type, however, miss longer-distance constraints on language, resulting in speech recognition output that appears inappropriate or even nonsensical.

For example, using a trigram model, a speech recognizer might misrecognize *the father of the bride* as *the father of the bribe*, even though to a fluent speaker of English, the former phrase is far more natural than the latter. Looking at only a three-word window, *of the bribe* seems not particularly less likely than *of the bride*, since it would occur very naturally as part of a phrase such as *the amount of the bribe*. A larger window would be required to consider *father* and *bride* simultaneously, but as the size of the window increases n-gram models suffer from a sparse-data problem. Consider the phrase *place the rubidium in the container*. The word *rubidium* is sufficiently rare in English that using its occurrence to estimate the likelihood of any following words would be difficult, yet the verb *place* by itself makes the subsequent phrase *in the container* highly probable. A simple n-gram model will not capture this effect, however, because of the rarity of the intervening noun.

Examples such as these have led researchers to try to incorporate language models that reflect more linguistic structure into recognition, especially in applications that involve interpretation of

*Published in *Computational Models of Speech Pattern Processing*, Ponting (ed.), Springer-Verlag, 1999. Part of the research described here was conducted while the author was at the Artificial Intelligence Center of SRI International, and was supported by the Defense Advanced Research Projects Agency under Contract N66001–94–C–6046 with the Naval Command, Control and Ocean Surveillance Center.

speech, rather than simply recognition. In such systems, considerable effort often goes into analyzing the linguistic structure of utterances in the process of applying natural-language processing techniques to extract their meaning. It seems natural to try to use this same information to improve the language model used in recognition.

2 Issues in Language Modeling for Speech Recognition

Before looking in detail at some of the techniques that can be used to incorporate linguistically motivated knowledge sources in recognition, there are some general issues that need to be considered. First, it is important to distinguish between a language model and the search architecture that applies the model. The model itself addresses the question of how information from a particular knowledge source (such as a natural-language grammar) can be mathematically combined with information from other knowledge sources (including acoustic models) to optimize recognition accuracy. Search architectures address the issue of how particular models can be applied computationally. Early work on applying natural-language processing techniques in speech recognition often addressed only the search problem, using rather naive models. It was only later realized that these naive models could actually degrade recognition accuracy rather than improve it, rendering the search problem for such models irrelevant.

An additional set of issues surrounds the question of whether to try to model the language in a particular application qualitatively or probabilistically. A purely qualitative language model simply provides a specification of the permitted word strings, and the search attempts to find the permitted word string that best matches the acoustics. In practical recognizers, such models are often specified in terms of a finite-state grammar, which can be directly incorporated into the acoustic search performed by a hidden-Markov-model (HMM) speech recognizer.

Probabilistic, or statistical language models (such as the n-gram models discussed above) are normally applied using the following instance of Bayes' rule:

$$p(W|A) = \frac{p(A|W) \cdot p(W)}{p(A)}$$

This equation characterizes the maximum likelihood approach to speech recognition. The goal is to find the word sequence W that is most likely given the acoustic evidence A. Bayes' rule expresses this in terms of the probability of A given W, which is provided by the acoustic model, the prior probability of W, which is provided by the language model, and the prior probability of A. Since the probability of A is constant for any particular recognition task, the term $p(A)$ can be ignored, and the problem comes down to finding the word sequence W that maximizes the product of the estimates for $p(A|W)$ and $p(W)$.

Finally, hybrid combinations of the two approaches are possible. For example, an n-gram statistical language model might be combined with a numerical penalty for hypotheses that are judged ungrammatical by some more complex qualitative language model. Or, as we shall see later, a qualitative grammar can be used to structure the hypotheses to which a statistical language model is applied.

Qualitative and statistical language models each have advantages and disadvantages. Statistical language models usually have thousands to millions of parameters that have to be estimated

empirically from thousands to millions of words of training data. To be effective, this data has to be from a source that is similar to the language used in the application. A language model trained on newspaper articles would provide a poor fit to the language used in a speech interface to a military command and control system. For novel applications, there may be no reasonable source of language model training data until the system is built and in use. Moreover, even when application-specific training data is available, there is never enough data to estimate all the parameters of the language model directly. Because of the sparse-data problem alluded to above, some parameters of the model always have to be estimated by a combination of lower-order parameters of the model. For example, in a trigram language model, there are always some trigrams that are never seen in the training data. Their probabilities have to be estimated from bigram and unigram data, to avoid assigning them a probability of 0.

Qualitative language models are usually specified in the form of a grammar and lexicon written by hand, with at least some degree of tailoring to the application. This means that human expertise about the application can be substituted for nonexistent training data. The disadvantage is that purely qualitative language models are less robust than statistical models. Statistical models are usually designed to avoid assigning any string within the vocabulary of the system a probability of 0. Hence, any string that might be uttered that is within the vocabulary has a chance of being correctly recognized if the acoustic evidence in favor of it is strong enough. Qualitative language models, on the other hand, simply classify strings as possible or impossible. Since in reality no human expert can anticipate all the meaningful utterances that could arise in a particular application, invariably some strings that are actually uttered will be "out of grammar," and thus be misrecognized, no matter how strong the acoustic evidence for them might be. Moreover, purely qualitative language models are unable to discriminate between two utterance hypotheses that are both classified as possible, even if one is far more likely than the other.

Finally, use of purely qualitative language models makes it impossible to take advantage of robust interpretation strategies that have been developed for natural-language processing. Strategies going by such names as "template matching" or "fragment combining" make it possible to successfully interpret strings, even if they cannot be completely analyzed by the grammar used by the system. If the recognizer is constrained, however, only to produce fully grammatical hypotheses, these robustness strategies will never have the opportunity to be applied, even in cases where they would be successful if applied to the out-of-grammar string actually uttered.

3 Formal Models for Natural Language

The difficulty of incorporating natural-language knowledge sources into speech recognition is largely a function of the complexity of the model. Various types of model have been used in natural-language processing systems, usually associated with a particular type of formal grammar. Of these, the ones that seem most worthy of attention are finite-state grammars, context-free grammars, and various extensions of context-free grammars. Although we present these models in nonprobabilistic form, probabilistic versions of all of them are possible, as we will discuss later.

3.1 Finite-State Grammars

Finite-state grammars can be expressed in a variety of notations. One that is often used to specify language models for speech recognition systems is a set of definitions of grammatical categories in terms of regular expressions over words or other grammatical categories. To ensure that the resulting language is finite-state, no category definition is allowed to be either directly or indirectly recursive. For example, the definition of a simple noun phrase might be written as

$$\mathsf{NP} \rightarrow \mathsf{Det\ Adj^*\ N}$$

which would mean that a noun phrase consists of a determiner (*a*, *the*, and so forth), followed by any number of adjectives, followed by a noun. This definition would classify strings such as *the girl*, *the little girl*, and *the pretty little girl* as noun phrases. The nonrecursivity constraint requires that the definitions of the categories Det, Adj, and N cannot make use of the category NP either directly or indirectly.

Finite-state grammars have the advantageous property that they can be expressed in terms of a finite-state-transition network, which means, as noted above, that they can be incorporated directly into the search performed by an HMM speech recognizer. Unfortunately, finite-state grammars are not expressive enough to describe the structure of natural languages. Suppose we wanted to extend the definition of a noun phrase to include the fact that it can contain a series of prepositional phrases following the noun. A natural way to do this would be to modify the definition of NP and add a definition of PP (for prepositional phrase) as follows:

$$\mathsf{NP} \rightarrow \mathsf{Det\ Adj^*\ N\ PP^*}$$

$$\mathsf{PP} \rightarrow \mathsf{P\ NP}$$

This pair of definitions would classify as noun phrases strings such as *the little girl in a red dress on a bicycle*. These definitions no longer constitute a finite-state grammar, however, because the categories NP and PP are defined by mutual recursion.

It has long been been argued (e.g., by Chomsky [1957, p. 21]) that finite-state grammars are inadequate in principle to characterize the word strings of natural languages. The point here, however, is that even if it were possible to define natural-language constructions by finite-state grammars, in many cases it is extremely unnatural to do so. Indeed, in the case under discussion, the same set of strings can be classified as noun phrases by the definition

$$\mathsf{NP} \rightarrow \mathsf{Det\ Adj^*\ N\ (P\ Det\ Adj^*\ N)^*}$$

but this denies the grammar writer the very useful category of prepositional phrase, and is so unnatural that it is somewhat difficult to see that it really does define the same set of noun phrases as the definition in terms of prepositional phrases.

Finite-state language models would be more useful if it were possible to use a more general form of specification such as a context-free grammar, but transform it into an equivalent finite-state grammar if one exists. Unfortunately, this transformation problem is undecidable in the general case. (See Theorem 8.15 of Hopcroft and Ullman [1979, p. 205].)

3.2 Context-Free Grammars

Context-free grammars can be defined in several ways, including simply removing the nonrecursivity constraint in the definition we used for finite-state grammars. More usually, they are defined as a set of rules that have a single atomic grammatical category on the left-hand side, and a sequence of atomic categories and words on the right-hand side, such as

$\mathsf{S} \rightarrow \mathsf{NP\ VP}$

$\mathsf{NP} \rightarrow \mathsf{Det\ N}$

$\mathsf{NP} \rightarrow \mathsf{NP\ PP}$

$\mathsf{PP} \rightarrow \mathsf{P\ NP}$

$\mathsf{VP} \rightarrow \mathsf{V\ NP}$

As long as multiple rules are allowed for a single category, eliminating the use of regular expressions on the right-hand sides of rules does not reduce their expressive power in terms of the sets of strings that they define.

The ability to define grammatical categories by mutual recursion makes context-free grammars much more suitable for defining linguistically based language models, since they can at least model the gross surface structure of natural-language expressions. They suffer, however, in their ability to model the fine detail of constraints on natural language. The rule $\mathsf{S} \rightarrow \mathsf{NP\ VP}$, for instance, is intended to say that a sentence can consist of a noun phrase followed by a verb phrase, but this omits many important details. Generally, the noun phrase and the verb phrase agree in person and number; the verb is tensed; and an important feature of the sentence, whether it is interrogative or declarative, depends on whether the noun phrase is interrogative (i.e., whether it begins with a "question word" such as *who* or *what*).

Additional information such as this can be expressed in a context-free grammar, but only at the cost of greatly expanding the number of categories and rules. To take the issue of agreement between noun phrases and verb phrases in person and number, English has three persons (first for *I*, *we*, and so forth; second for *you*; and third for others) and two numbers (singular and plural). To express the constraint that in a sentence the subject noun phrase and the verb phrase generally agree in person and number, we would have to have six different categories each for noun phrases and verb phrases (one for each combination of person and number), and six rules of the form $\mathsf{S} \rightarrow \mathsf{NP\_first\_singular\ VP\_first\_singular}$, and so forth. If we wish to add the information that whether the sentence is interrogative or declarative depends on the form of the noun phrase, we would have to add even more distinctions among categories and more rules. This verbosity of context-free grammars in describing the details of natural languages has led researchers in natural-language processing to augment context-free grammars in various ways to allow grammars to be more concise.

3.3 Augmented Context-Free Grammars

Over the years, many kinds of augmentation for context-free grammars have been proposed to make them more useful for specifying grammars of natural languages. In the 1970s, augmented transition networks, or ATNs (Woods, 1970), were widely used. In this formalism, context-free

grammars were represented in the form of recursive transition networks, and the augmentations were defined by procedures attached to the arcs of the networks. The disadvantage of procedural augmentations to context-free grammars is that they have to be tailored to the particular processing algorithms used. Thus, the procedural annotations to a grammar appropriate for top-down parsing would be different from those appropriate for bottom-up parsing, because the information to be processed would become available in a different order.

Since the mid-1980s, the dominant approach to supplementing the expressive power of context-free grammars has shifted from ATNs or other procedural formalisms to declarative formalisms, particularly those based on various styles of "unification grammar." These formalisms have the advantage that their meaning is well-defined independent of any processing strategy or algorithm (although in practice there are interactions between the way a grammar is written and the processing algorithm that affect the efficiency, and even the termination, of the processing strategy).

The following example shows a rule written in the unification-based formalism used in the Core Language Engine (Alshawi, 1992) developed by SRI International's research group in Cambridge, England. This formalism is also used in the Gemini system (Dowding et al., 1993, 1994) developed by SRI in Menlo Park, California:

S:[tensed=yes] $\rightarrow$ NP:[person=P,num=N] VP:[tensed=yes,person=P,num=N]

What unification grammar adds to context-free grammar is the notion of feature constraints. In the notation presented here, grammatical categories are specified in terms of a major category symbol (such as S, NP, or VP), plus a set of feature constraints expressed by equations of the form feature=value. The power of the formalism comes from the ability to constrain a feature not to a specific value, but to a variable that also appears as the value of some other feature, requiring the two features to have the same value, without having to specify what value that is. Thus, in the sample rule, the noun phrase and verb phrase must agree in person and number—that is, the person and num features of the noun phrase must have the same respective values as the person and num features of the verb phrase. The term "unification grammar" comes from the fact that when the grammar is applied, the feature constraints are unified. This means that if the rule above is applied to a noun phrase of the category NP:[person=first,num=singular], the variable P will take on the value first, and the variable N will take on the value singular, which will in turn force the verb phrase to be of the category VP:[tensed=yes,person=first,num=singular].

3.4 Expressive Power of Grammar Formalisms and the Requirements of Natural Language

The example above shows how a unification grammar can specify a context-free language more concisely than a context-free grammar, but unification grammars can also define languages that are beyond the power of context-free grammars. In fact, it is easy to show that unification grammars can model any Turing machine. The construction is quite straightforward. Let there be a major category symbol corresponding to each Turing machine state, with each major category having three features: one representing the symbol currently being scanned by the head, one which encodes the tape to the left of the head, and one which encodes the tape to the right of the head. With this representation it is easy to write a grammar rule corresponding to each possible move of the Turing machine, for example,

S1:[current=a,left=X,right=(Y,Z)] $\rightarrow$ S2:[current=Y,left=(b,X),right=Z]

This rule represents the following Turing machine move: When in state S1, if the current symbol is a, replace it by b, move to the right, and go to state S2. When a final state is reached, additional rules can simulate moving to the beginning of the tape and outputting its contents.

The power of unification grammar to express non-context-free languages relies on the use of features that have an infinite value space. In the example above, setting the value of the left feature of the category S2 to the value (b,X), where X can be any value of the left feature of the category S1, implies that the value space for the feature left includes arbitrarily deeply nested ordered pairs. This is an infinite space, which indeed it must be, if it is to represent all possible Turing machine tapes to the left of the Turing machine head.

If we limit the features used in unification grammars to finite value spaces, however, then the formalism becomes equivalent to context-free grammars in the languages it can define. It is easy to show that an arbitrary unification grammar is equivalent to a grammar consisting of all possible fully instantiated substitution instances of the rules of the original grammar. If the features all have finite value spaces, then there will be finitely many such substitution instances, and these substitution instances can simply be viewed as the nonterminal symbols of a corresponding context-free grammar.

An important question is whether the additional expressive power that is gained by allowing unification grammars to have infinitely valued features is actually needed to describe natural languages—that is, whether all natural languages are in fact formally describable by context-free grammars. Since the earliest development of formal language theory, linguists have attempted to construct arguments to show that natural languages are not context-free, but Pullum and Gazdar (1982) have argued rather persuasively that these early arguments failed to achieve their goal.

Most of the recent attention on this topic has focused on whether there are natural languages that have unbounded cross-serial dependencies. This refers to expressions of the form

$$\ldots a_1, a_2, \ldots a_n, \ldots b_1, b_2, \ldots b_n, \ldots$$

for arbitrary n, such that the form of b_i depends on the form of a_i. Languages characterized by such expressions are known to be non-context-free. Pullum and Gazdar consider data suggesting that Dutch contains unbounded cross-serial dependencies, but argue that while the "correct" linguistic description of Dutch may indeed involve such dependencies, the data does not show that the strings of Dutch constitute a non-context-free language, essentially because the surface effects of the cross-serial dependencies in Dutch are not strong enough to rule out other ways of generating the same strings by a context-free grammar. However, subsequent data collected by Shieber (1985) regarding Swiss-German and by Culy (1985) concerning the African language Bambara have been generally accepted as demonstrating cross-serial dependencies in natural languages that cannot be modeled by context-free grammars.

Although the arguments of Shieber and Culy may show that in principle there are natural-language phenomena that cannot be modeled by context-free grammars, it is not clear that this is of much practical consequence to the choice of grammar formalism for language modeling in speech recognition. First, only a handful of languages are currently accepted as having constructions that are inherently non-context-free. Second, if these phenomena are modeled by allowing infinitely valued features in unification grammar, an approximating context-free grammar can be constructed

by partitioning the value space into finitely many equivalence classes. (If all values for a feature are put into a single equivalence class, this amounts to simply ignoring the feature in question.) This method results in a grammar that accepts every string accepted by the more detailed grammar, plus some additional strings that violate the constraints on the infinitely valued features. For purposes of language modeling for recognition, this seems preferable in terms of robustness to the alternative of an approximating context-free grammar that rejects certain strings permitted by the more detailed grammar.

4 Search Architectures for Natural-Language-Based Language Models

If a language model more complex than a finite-state grammar is to be used in speech recognition, then the question arises of how to incorporate the language model into the recognition search. Historically, three approaches to this problem have been predominant: word lattice parsing, N-best filtering or rescoring, and dynamic generation of partial grammar networks.

4.1 Word Lattice Parsing

Word lattice parsing (Chow and Roukos, 1989; Boisen et al., 1989) is probably the oldest approach to integration of complex language models into recognition. In this approach, the recognizer produces a set of word hypotheses, with an acoustic score for each potential pair of start and end points for each possible word. A natural-language parser or other complex language model is then used to find the path through the word lattice having the best combined acoustic and language model score. In the case of a qualitative language model, this is usually the fully grammatical string having the best acoustic score.

Older implementations of word lattice parsing were not particularly efficient, because they had to deal with a large degree of word boundary uncertainty. Normally, a word lattice of adequate size for accurate recognition will contain dozens of instances of the same word with slightly different start and end points. One approach to this problem (Chow and Roukos, 1989) is to associate with each word or phrase a set of triples of start points, end points, and recognition scores. Each possible parsing step is then performed only once, but a dynamic programming procedure must also be performed to compute the best score for the resulting phrase for each possible combination of start and end points for the phrase.

Recently, a much more efficient approach to processing of word lattices has been developed by Mohri (1997). In this approach, word lattices are treated as weighted finite-state transducers, which can be determinized and minimized using automata theoretic methods, resulting in lattices that have no word boundary ambiguity for any sequence spanning the lattice. Moreover, Morhi's empirical results show that this can be done very efficiently, with the resulting lattices being much smaller than the input lattices,.

4.2 N-best Filtering or Rescoring

N-best filtering or rescoring (Chow and Schwartz, 1989; Schwartz and Austin, 1990) is a very simple search architecture in which the recognizer enumerates its N-best full recognition hypotheses, which are reprocessed using other complex knowledge sources, such as natural-language-based language models. Qualitative language models can be implemented by N-best filtering, in which the recognizer simply produces an ordered list of hypotheses, and the language model chooses the first one on the list that is accepted by its grammar. Probabilistic or other numerical language models can be implemented by N-best rescoring, in which the recognition score for each of the N-best recognition hypotheses is combined with a score from the complex language model, and the hypothesis with the best overall score is selected.

The advantage of the N-best approach is its simplicity. The disadvantage is that it seems inefficient for large values of N. The computational cost of the best method known for exact enumeration of the N-best recognition hypotheses (Chow and Schwartz, 1989) increases linearly with N, but an approximate method exists (Schwartz and Austin, 1990) that increases the computational cost of recognition only by a small constant factor independent of N. The additional knowledge sources, however, then have to be applied independently to each hypothesis, so that for large values of N, this phase will take a long time.

4.3 Dynamic Generation of Partial Grammar Networks

In an HMM speech recognizer, a finite-state grammar can be represented as a set of state-word-state transitions. Any type of linguistic constraints can, in fact, be represented as such a set, but for language models beyond finite-state grammars in complexity, the set of transitions will be infinite. A network representation can be used, however, if the network is generated dynamically to include only the transitions needed for the recognition search for a particular utterance.

In the method of Moore, Pereira, and Murveit (1989; Murveit and Moore, 1990), when a word is successfully recognized beginning in a given grammar state, the recognizer sends the word and the state it started in to the natural-language parser, which returns the successor state. To the parser, such a state encodes a parser configuration. When the parser receives a state-word pair from the recognizer, it looks up the configuration corresponding to the state, advances that configuration by the word, creates a name for the new configuration, and passes back that name to the recognizer as the name of the successor state. If it is impossible, according to the grammar, for the word to occur in the initial parser configuration, then the parser sends back an error message to the recognizer, and the corresponding recognition hypothesis is pruned out. Word boundary uncertainty in the recognizer means that the same word starting in the same state can end at many different points in the signal, but the recognizer has to communicate with the parser only once for each state-word pair. Because of this, the parser does not have to consider either acoustic scores or particular start and end points for possible words, those factors being confined to the recognizer.

The method of Moore, Pereira, and Murveit is designed to work with any kind of natural-language-based language model for which left-to-right parsing is possible, and was implemented for the kind of unification grammar described in Section 3.3. For context-free grammars, a simpler method of dynamic generation of partial grammar networks has been pioneered by Dupont and Snyers (1989; Dupont 1993). In this method, an finite-state network is constructed from the

grammar rules for each nonterminal symbol, but the network may incorporate transitions for nonterminals as well as words. Whenever a nonterminal is encountered in the course of the recognition search, the network for that nonterminal is copied in place of the nonterminal. This method will handle any context-free grammar as long as the grammar includes no direct or indirect left recursion (which would cause an infinite sequence of expansions of nonterminal symbols to occur). Techniques similar to this enable commercially developed recognizers from Microsoft (Huang et al., 1995) and Nuance Communications (1996) to accept non-left-recursive context-free grammars.

5 Compiling Unification Grammars into Context-Free Grammars

In Section 3.4 we pointed out that unification grammars with finitely valued features are equivalent to context-free grammars in expressive power. In this section we describe a method for efficiently compiling such unification grammars into context-free grammars, while eliminating left recursion, and give empirical results on the efficiency of the resulting grammars when used with the Nuance speech recognition system.

5.1 Instantiating Unification Grammars

Given a unification grammar with only finitely valued features, it would be trivial to transform all unification rules into context-free rules, by generating all possible full feature instantiations of every rule and making up an atomic name for each combination of category and feature values that occurs in these fully instantiated rules. This can easily increase the total number of rules to a size that would be too large to deal with, however. We therefore instantiate the rules in a more careful way that avoids unnecessarily instantiating features or generating unnecessary combinations of feature values.

A prime example of the problem we need to circumvent is a rule that occurs in a grammar developed by SRI for CommandTalk (Moore et al., 1997), a spoken-language interface to a military simulation system:

$$\text{coordinate\_nums:[]} \rightarrow \text{digit:[] digit:[] digit:[] digit:[]}$$

This rule says that a set of coordinate numbers can be a sequence of four digits. In the CommandTalk grammar, the digit category has features (e.g., singular vs. plural, 0 vs. non-0) that would generate at least 60 combinations if all instantiations were considered. So, if we naively generated all possible complete instantiations of this rule, we would get at least 60^4 rules. Even worse, we need other rules to permit as many as eight digits to form a set of coordinate numbers, which would give rise to 60^8 rules.

To avoid this unnecessary expansion in the number of rules, we define the set of atomic categories by considering, for each daughter category of each rule, all instantiations of just the subset of features on the daughter that are constrained by the rule. Thus, if a rule does not constrain a feature on a particular daughter category, an atomic category will be created for that daughter that is under-specified for the value of that feature. Since our example rule puts no constraints on any

of the features of the digit category, by generating an atomic category that is under-specified for all features, we need only a single rule in the derived grammar.

In addition to using categories that are under-specified for values of certain features, we are careful to generate only instances of rules that could actually arise in applying the given grammar and lexicon. The algorithm works as follows:

1. Generate an initial set of rules incorporating "don't care" values for unconstrained features:

    - Treat each lexical entry as a rule whose mother is the category of the lexical entry and whose daughters are a word or word sequence.
    - Transform each rule of the original grammar and lexicon by placing every expression and subexpression, including variables, that occurs in a feature value inside the wrapper val, for example, transform foo(a,X) into val(foo(val(a),val(X))). This prevents "don't care" values from unifying with expressions corresponding to variables in the original grammar, while still permitting unification of any pair of expressions corresponding to a unifiable pair of expressions in the original grammar.
    - Specify an atomic "don't care" value for all the unconstrained features on daughter categories in rules, removing the val wrapper around any expression containing the "don't care" value.
    - For each category modified to include "don't care" values for some of its features, make a corresponding "don't care" variant of every rule whose mother matches the original category.

2. Generate an intermediate set of rules by computing all instances of the initial rules that can arise by recursively applying grammar rules to the lexicon, as in a bottom-up chart parser:

    - Process each rule as an active edge, none of whose daughters has been matched.
    - When processing an active edge, if the next unmatched daughter is a word, move the daughter from the list of unmatched daughters to the list of matched daughters, and process the resulting active edge, if an identical edge has not already been processed.
    - When processing an active edge, if the next unmatched daughter is a category, for each possible unification of the category against an inactive edge, move the resulting unified category from the list of unmatched daughters to the list of matched daughters, and process the resulting active edge, if an identical edge has not already been processed.
    - When processing an active edge, if there are no more unmatched daughters, create an intermediate rule from the mother and daughters of the active edge, and process the mother as an inactive edge, if an identical edge has not already been processed.
    - When processing an inactive edge, for each possible unification of the inactive edge against the next unmatched daughter of an active edge, move the resulting unified category from the list of unmatched daughters to the list of matched daughters, and process the resulting active edge, if an identical edge has not already been processed.

3. Generate a fully instantiated set of rules by traversing the intermediate rules top-down from the "start" categories of the grammar, instantiating any remaining variables in features to a dummy value.

4. Transform the fully instantiated rules into a context-free grammar, by constructing a unique atomic name for each instantiated category.

Step 1 of this algorithm handles the "don't care" feature values, as we discussed previously. The key to the efficiency of this algorithm, however, is step 2, since the only combinations of feature values that we ever consider are those created in step 2 by unifying active and inactive edges, rather than generating all possible combinations of feature values and then discarding those that are never used. Step 3 ensures that we include in the final grammar only rule instances that can be used in a complete analysis of an utterance, by filtering the rules top-down. Step 4 converts the grammar to context-free form, by replacing the complex category structures with atomic category names.

This compilation algorithm was tested on a version of the CommandTalk grammar that had 892 phrasal rules and 1688 lexical rules. The compilation took about 5 minutes on a Sun SPARCstation 10 with a 90-MHz HyperSPARC processor. The resulting context-free grammar had 2886 phrasal rules and 5719 lexical rules; because of the care the algorithm takes not to introduce unnecessary rules or categories, the grammar size increased only by about a factor of 3. We estimated that the size of the equivalent context-free grammar without introducing "don't care" feature values would have been approximately 10^{18} rules, almost all of which would be rules for sequences of coordinate numbers. Without the rules for coordinate numbers, we estimated that the size of the context-free grammar without "don't care" feature values would have been about 10^{6} rules.

5.2 Removing Left Recursion from Context-Free Grammars

The standard way to remove left recursion from a context-free grammar is to convert the grammar to Greibach normal form, or GNF (Hopcroft and Ullman, 1979, pp. 94–99). In a GNF context-free grammar, the left-most daughter in each rule is a word, which obviously makes left recursion impossible. To eliminate left recursion from grammars generated for the Nuance recognizer, we produce grammars in a generalized version of Greibach normal form, or GGNF, taking advantage of the fact that the Nuance grammar formalism allows regular expressions on the right-hand side of grammar rules. In a GGNF grammar each category is defined by a single rule whose right-hand side is a GGNF regular expression over words and categories. In a GGNF regular expression the left-most subexpression is either a word, a GGNF regular expression, or an alternation of GGNF regular expressions. A GGNF regular expression has the property that any string of words and categories matching the regular expression must begin with a word. Thus, any GGNF rule is equivalent to a set of GNF rules. For additional compactness in the grammar, all alternations and sequences are flattened, so that no alternation has an alternation as an immediate subexpression and no sequence has a sequence as an immediate subexpression, and all alternative sequences are prefix merged, so that no alternation has as immediate subexpressions two sequences (including sequences of length 1) that have the same first element.[1]

[1]Prefix merging a sequence of length 1 with longer sequences requires using a notation for optional expressions, the merged sequence consisting of the common element followed by optional extensions.

We conducted tests to compare the speed performance of a GGNF grammar to an equivalent finite-state grammar when used with the Nuance recognizer. In general, unification grammars cannot be compiled to finite-state grammars, even when restricted to finitely valued features. However, we had imposed additional restrictions on the CommandTalk grammar to make this possible, since earlier versions of the Nuance recognizer lacked the capability to accept context-free grammars that were not finite-state. The comparison was made with the CommandTalk grammar and a test set of about 900 within-grammar utterances, on a Silicon Graphics Indigo 2 with a 150-MHz R4400 processor. Recognition was 1.265 $\times$ real time with the finite-state grammar, and 0.856 $\times$ real time with the GGNF grammar. This represents a 32% speedup using the GGNF form of the grammar. (Word recognition accuracy was 92.5% with both grammars.)

In terms of recognition speed, the GGNF grammar was clearly superior to the finite-state grammar, but the time required to compile the GGNF grammar into a Nuance recognition package was much greater. The finite-state grammar took 42 seconds to compile, while the GGNF grammar took 83 minutes. Since this compilation happens offline, for some purposes the penalty in compilation speed might not matter, but we felt that it was great enough to be an impediment to the grammar development cycle. We found another variation in grammar form and compilation settings, however, that offered almost as fast recognition, and much faster compilation.

The variant of GGNF in question is distributed-alternation GGNF, or DAGGNF. In a DAGGNF grammar an alternation can occur in a sequence only as the last element. This is achieved by distributing (in the algebraic sense) alternations over sequences, and prefix merging the result. The purpose of DAGGNF is to make the prefix merging already imposed in GGNF apply to more cases. A DAGGNF grammar can be significantly larger than the corresponding GGNF grammar, so Nuance compilation would take even longer, except for the fact that it seems to virtually duplicate a major phase of optimization in Nuance compilation, which we can therefore turn off, using a compile-time flag provided by Nuance.

Performing a Nuance compile, with the optimization turned off, of the DAGGNF form of the version of the CommandTalk grammar reported above reduced Nuance compilation time from 83 minutes to 4:57 minutes. Nuance recognition speed under the test conditions reported above decreased from 0.856 $\times$ real time to 0.890 $\times$ real time, an increase of 4.0%, but still 30% faster than with the finite-state form of the grammar.

6 Robust Natural-Language-Based Language Models

The techniques described in the preceding section have proven effective for defining qualitative language models based on unification grammars, for the recognition of within-grammar utterances. As we have previously discussed, qualitative language models are incapable of correctly recognizing out-of-grammar utterances, since they effectively assign such utterances a probability of 0. To use natural-language-based language models in a more robust way, it is necessary to convert their results into some sort of numerical score that can be combined with the numerical scores assigned by other knowledge sources, including acoustic models, in such a way that failure to find a completely grammatical analysis of a hypothesis does not totally rule it out.

There are several simple ways in which this could be done. Probably the simplest is to give a fixed numerical penalty to any hypothesis that is not accepted by the grammar. More informa-

tion can be extracted from a natural-language grammar, however, either by computing the "edit distance" (total number of substitutions, insertions, and deletions) of a hypothesis from the closest grammatical string, or by counting the minimal number of grammatical fragments needed to cover the hypothesis. These methods have the advantage that they have very few parameters to estimate, perhaps only a single weighting factor to combine the language model score with the other knowledge sources. This means that the only data required for parameter estimation would be a small tuning set, in contrast to the large training sets required for fully statistical language models that have tens of thousands (or more) of parameters to estimate.

Experience suggests, however, that these minor variations on essentially qualitative methods do not offer much improvement over conventional n-gram language models, if sufficient training data is available to use statistical methods. In the 1993 DARPA benchmark speech recognition evaluation on the Air Travel Information Service (ATIS) task, SRI used a knowledge source based on the minimal number of grammatical fragments found by a unification-based grammar (Moore et al., 1994). Adding this knowledge source to a recognition system that also incorporated a word-class-based trigram language model reduced the word error rate only from 5.4% to 5.2%.

6.1 Combining Linguistics and Statistics in a Language Model

After the initial attempt in 1993 to use a unification-based natural-language grammar as a knowledge source for language modeling in speech recognition, the following year SRI undertook to develop a more integrated approach to combining linguistic and statistical factors in a single language model (Moore et al., 1995). Again, the approach was based on the idea that, even when the grammar fails to provide a complete analysis of an utterance, the number of grammatical (and semantically meaningful) phrases needed to span the utterance is likely to be smaller than for incorrect competing hypotheses. This is illustrated by an example from the December 1993 ARPA ATIS benchmark test set:

hypothesis: [*list flights*][*of fare code*][*a*][*q*]

reference: [*list flights*][*of fare code of q*]

These two word strings represent the SRI DECIPHER recognizer's first hypothesis for the utterance and the reference transcription of the utterance, each bracketed according to the best analysis that the SRI Gemini ATIS grammar was able to find as a sequence of semantically meaningful phrases. Because of a missing sortal combination, Gemini did not allow the preposition *of* to relate a noun phrase headed by *flights* to a noun phrase headed by *fare code*, so it was not possible to find a single complete analysis for either word string. Gemini was, however, able to find a single phrase spanning *of fare code of q*, but required three phrases to span *of fare code a q*, so it still strongly preferred the reference transcription of the utterance over the recognizer's first hypothesis.

SRI's original attempt to use this information in language modeling simply took this raw count of number of fragments, plus some other ad hoc factors, to compute a language model score. In the model that was subsequently developed, the Gemini grammar was still used to analyze a recognition hypothesis as a sequence of semantically meaningful fragments, but then n-gram statistics were used to estimate the probability of the hypothesis under that analysis. The resulting language model was a kind of multilevel n-gram model. The top level was a trigram model of the probability of a hypothesis as a sequence of types of fragments. That is, the model estimated the

probability of an utterance being a sequence of, for instance, a sentence followed by a modifier phrase followed by a nominal phrase. The fragment types used were sentence, nominal phrase, modifier phrase, filler (e.g., *please*), and "skipped". This last category consisted of the sequences of words that were left over in determining the best coverage of the utterance in terms of well-formed phrases of major semantic classes in the ATIS domain. A trigram model was used for this level to model well the important case of an utterance consisting of the *begin_utterance* token, a single sentence, and the *end_utterance* token.

The next level of the model was a word-class-based four-gram model of each fragment. Initially, separate models were estimated for each type of fragment, but this method suffered from splitting the training data into smaller pools. The method of modeling fragments that was finally adopted was a single four-gram model, but one that treats each fragment as a sequence starting with a token such as *begin_sentence* or *begin_nominal_phrase*. The result is that the probability of the first few words of each fragment is conditioned on what type of fragment it is, but once there are several words of context, the probability estimates are conditioned on that rather than on the type of fragment. The four-gram model was smoothed by linear combination with lower-order models, the weights being estimated by deleted interpolation. For the word-class models, the classes were generated semiautomatically from the Gemini lexicon. The probability estimates described above were combined into a simple joint probability estimate for a hypothesis under an analysis as a sequence of fragments.

This Gemini-based language model was used in the December 1994 ATIS speech recognition benchmark evaluation. The results for SRI's recognizer for all test utterances both with and without this knowledge source are given in Table 1. The baseline DECIPHER recognizer incorporated a word-class-based trigram language model. As the table shows, the improvement in recognition by rescoring DECIPHER output with the Gemini-based language model was about 15% on both word error and utterance error. These differences were measured to be statistically significant at the 95% confidence level on all four significance tests used in the evaluation.

	Word Error	Utterance Error
Baseline DECIPHER	2.5%	15.5%
DECIPHER+Gemini	2.1%	13.1%
Improvement	14.6%	15.1%

Table 1: December 1994 ATIS benchmark test results.

It is notable that this improvement in recognition accuracy, by adding a natural-language-based knowledge source, was achieved relative to a state-of-the-art baseline recognizer. (The baseline SRI recognizer was otherwise the top-performing system in this evaluation.) Repeatedly in the past, improvements have been obtained with natural-language-based knowledge sources in recognition, only to have the improvements disappear as baseline recognition accuracy has improved. The results described here represent the only significant improvement we are aware of obtained by using a natural-language-based knowledge source in conjunction with a current state-of-the-art recognizer in a blind test.

6.2 Fully Statistical Natural-Language Grammars

The model we have just described combines linguistic information and statistical information in a principled way, with a substantial resulting gain in recognition accuracy. However, only a single level of linguistic structure—the utterance as a sequence of linguistic fragments—is statistically modeled. One might hope for additional improvement in recognition accuracy by modeling more of the linguistic structure statistically. Recently, there has been much interest in statistical natural-language grammars, but so far, there appear to be no significant results in using them to improve speech recognition. Hence the work and ideas described here should be regarded as promising, but unproven, in regard to language modeling for speech recognition.

6.2.1 Probabilistic Context-Free Grammar

The most obvious candidate for fully statistical natural-language grammars are probabilistic context-free grammars. In this form of statistical language model, each context-free grammar rule has associated with it a conditional probability for the right-hand side given the left-hand side. For example, assigning the rule $\mathsf{S} \rightarrow \mathsf{NP}\ \mathsf{VP}$ a conditional probability of 0.5 means that, if there is a phrase of category S, there is a probability of 0.5 that it consists of a phrase of category NP followed by a phrase of category VP. If the top-level category of the grammar is taken to have a probability of 1, then this immediately defines a probability for each analysis tree allowed by the grammar, by multiplying all the conditional probabilities associated with the rules used in the analysis.

For such an approach to produce a good enough language model to compete with conventional n-gram models, however, some form of lexical or semantic conditioning must be included. (This observation was perhaps first made in print by Church [1989].) For example, without information about likely combinations of lexical subjects, verbs, and objects, a statistical grammar incorporating only syntactic constraints would rate *the pizza ate the boy* about as likely as *the boy ate the pizza*. The most common response to this observation in work on statistical natural-language grammars has been to include the lexical head of each phrase as part of the information used to estimate probabilities in a statistical context-free grammar. (Informally, the lexical head of a phrase is the main word of the phrase.) In *the boy ate the pizza*, the subject would be categorized not just as a noun phrase (or perhaps a singular noun phrase), but as a noun phrase whose lexical head is *boy*, and the verb phrase would be categorized as a verb phrase whose lexical head is *ate*.

Taking into account the lexical head of each category mentioned in a rule means that, in effect, conditional probabilities must be estimated for all possible combinations of lexical heads for the nonterminals that appear in a given context-free rule. So, the $\mathsf{S} \rightarrow \mathsf{NP}\ \mathsf{VP}$ rule would have separate estimates for each combination of noun as the head of the NP and verb as the head of the VP. This causes a very severe sparse-data problem, since a substantial proportion of the combinations of a particular context-free rule and particular lexical heads for each nonterminal in the rule would occur no more than once in even a very large training corpus. This problem is addressed by making various independence assumptions and using back-off smoothing methods.

Models of this general type have been extensively explored in the past few years (Charniak, 1995; Collins, 1996, 1997; Grishman and Sterling, 1993; Lafferty et al., 1992; Magerman, 1995). Although not all of this work has been explicitly framed as probabilistic context-free grammar, all of it has involved atomic grammatical categories annotated with lexical head information. There

seem to be no reported results using such models for speech recognition, however. Instead, the chosen task for evaluating these models tends to be sentence parsing, where the main figure of merit is accuracy in structurally bracketing the sentences. The current state-of-the-art performance seems to be that of Collins (1997), whose method produced completely correct bracketings for 64% of the sentences of 100 words or less using a standard partitioning of the Penn Treebank corpus (Marcus et al., 1993) into training and test sets.

6.2.2 Probablistic Unification Grammar

None of the work cited above on probabilistic context-free grammar addresses the degree of detail routinely modeled in unification grammars for natural language. Much of the work adopts the linguistic categories used in the Penn Treebank, which has 35 lexical categories (not counting 13 categories for symbols and punctuation, which would not arise in spoken language) and 14 phrasal categories. Contrast this with the report of Black et al. (1993), who found that parsing a 15,000-sentence corpus with their unification grammar of English used 23,431 distinct unification categories produced by different combinations of categories and feature values. Thus, if unification grammars are also to be annotated with lexical-head information, as seems necessary to be competitive with simple n-gram models, the sparse-data problem will be correspondingly worse than with typical probabilistic context-free grammars.

Perhaps as a result of this problem, there seem to be no published reports of models that attempt to incorporate all the constraints of a complex unification grammar into a statistical model. Briscoe and Carroll (1993) use a unification grammar in a statistical model, but estimate the probabilities in their model based only on the context-free approximation formed by ignoring all the feature constraints. The feature constraints are imposed afterwards as a filter, but they are not incorporated into the statistical model per se. The statistical model of Black et al. (1993) is more complex, but somewhat similar in its treatment of unification constraints. For conditioning purposes Black et al. replace their complex feature-based categories by 50 atomic syntactic categories and 50 atomic semantic categories so there are at most 2500 possible linguistic categories available, compared to the 23,431 categories they report as actually arising in parsing their corpus. Goodman (1997) describes a method based on independence assumptions and back-off smoothing that could, in principle, be applied to arbitrary unification grammars, but he tests it only on a simplified model with many fewer features than a realistic unification grammar of a natural language would need, and he gives no reason to believe that the method would scale up particularly well.

There is a rather different approach to the sparse-data problem for probablistic unification grammar, that, while it is as yet untested, might provide a solution. Instead of focusing on the set of possible categories, which may be enormous, we focus instead on the set of rules that actually compose the grammar. The size of the rule set is typically fairly small; there are seldom many more than 1,000 phrasal syntactic rules in a hand-written unification grammar, either for a general grammar for a natural language, or a task-specific grammar for a particular application. The question we need to answer is: on what should we condition the probability of a particular rule being correct in a given context? It seems clear that the most important single factor that determines whether a particular rule correctly applies in a given partial derivation is whether the unification constraints allow it to apply. If the unification constraints are not satisfied, then it is impossible for the rule to apply, so its probability of being the correct rule would be 0. This suggests that a

primary conditioning factor in estimating the probability of a rule should be whether the unification constraints on the rule are satisfied. At its crudest, this would mean counting or estimating the number of times in the derivation of a training corpus that a given rule was correct and the number of times the constraints on the application of the rule were met, and taking the ratio between these counts as an estimate of the conditional probability of the rule.

For example, suppose, in the analysis of the sentence *which man did John see*, we have a partial derivation that correctly predicts that the sentence is a *wh*-question whose initial noun phrase must unify with the category NP:[number=X,wh=yes]. That is, the initial noun phrase must be an interrogative noun phrase. Now suppose we have the following three rules to consider:

NP:[number=X,wh=no] $\rightarrow$ Name:[number=X]

NP:[number=X,wh=Y] $\rightarrow$ Pron:[number=X,wh=Y]

NP:[number=X,wh=Y] $\rightarrow$ Det:[number=X,wh=Y] N:[number=X]

The first rule says that a non-*wh*-noun-phrase can be a proper name. The unification constraints on this rule are not satisfied, so the partial derivation we are considering is counted neither as a case in which the rule could apply nor as a case in which it is correct. The second rule says that a noun phrase can be a pronoun, and that the noun phrase is a *wh*-noun-phrase if the pronoun is a *wh*-pronoun (e.g., *who*). Since this rule is not excluded by the constraint wh=yes on the predicted noun phrase, this counts as an instance in which the rule could apply. However, since the noun phrase we actually have at this point in the data, *which man*, is not a pronoun, this is not a case in which the rule is correct. The third rule says that a noun phrase can be a determiner followed by a noun, and that the noun phrase is a *wh*-noun-phrase if the determiner is a *wh*-determiner (e.g., *which*). In this case, the rule can apply because its unification constraints are met, and it is correct, since the noun phrase in question is of the form determiner followed by noun.

Estimating the conditional probability of a unification rule as the ratio of the frequency with which it is correct to the frequency with which its unification constraints are met takes all unification constraints into account and seems to get at perhaps the most important factor in determining the likelihood of the rule being correct. It has a major problem, however. A valid probability distribution must sum to 1 for all possible hypotheses in any context. For the method we have just proposed, however, this will not always be the case, since a unification-based rule can have different competitors in different contexts, because of differing unification constraints. In the current example, if the noun-goes-to-pronoun rule and the noun-goes-to-determiner-noun rule are the only possible ways of expanding an interrogative noun phrase, then in this context the probability of those two rules must sum to 1. However, in another context when we are expanding a noun phrase that is not constrained to be interrogative, we would want to assign a non-0 probability to the noun-phrase-goes-to-proper-name rule.

This problem can be addressed by expanding the model slightly to take into account what other rules are applicable. We can partition the rules in a unification grammar into mutually exclusive sets of possible competing rules, taking into account only the major category category symbols that appear in the rules. Call a set of rules an "alternative rule set." Note that the notion of "alternative rule set" will be relative to a particular order of derivation. For top-down derivation, an alternative rule set would consist of all the rules having the same major category symbol for the left-hand side of the rule, just as in probabilistic context-free grammars. For bottom-up derivations, however,

the alternative rule sets might be defined by the major category symbol of the first or last category on the right-hand side of the rule. The treatment that follows, however, applies to any derivation process and any method of partitioning the rules into mutually exclusive subsets such that each rule falls into the same subset as all its possible competitors with respect to that derivation process.

Each alternative rule set gives us a property of a context that lets us easily define a probability distribution for which rule is correct in that context. Suppose $\{R_1, \ldots, R_k\}$ is the alternative rule set that contains all the rules that can apply in some particular context. Let $a(R_i)$ be a function that is 1 if rule R_i could apply in the context and is 0 otherwise, and let $c(R_i)$ mean that R_i is the correct rule in the context. What we need to estimate is $p(c(R_i)|a(R_1) = x_1, \ldots, a(R_k) = x_k)$ for $1 \leq i \leq k$, where x_i is 1 or 0, depending on whether rule R_i is applicable. In general, there will be too many possible combinations of values for $a(R_1), \ldots, a(R_k)$ to estimate this directly, but we can approximate it by using Bayes' rule,

$$p(c(R_i)|a(R_1) = x_1, \ldots, a(R_k) = x_k) = \frac{p(a(R_1) = x_1, \ldots, a(R_k) = x_k|c(R_i)) \cdot p(c(R_i))}{p(a(R_1) = x_1, \ldots, a(R_k) = x_k)}$$

and making the following conditional independence assumption:

$$p(a(R_1) = x_1, \ldots, a(R_k) = x_k|c(R_i)) \approx p(a(R_1) = x_1|c(R_i)) \cdot \ldots \cdot p(a(R_k) = x_k|c(R_i))$$

To make the approximation sum to 1 in all cases, we estimate the denominator of the right-hand of Bayes' rule simply to be the normalization constant found by summing the estimates for all the individual cases.

With this decomposition, the only parameters we need to estimate directly are those of the form $p(c(R_i))$ and $p(a(R_i) = 1|c(R_j))$, where R_i and R_j are members of the same alternative rule set. (Note that $p(a(R_i) = 0|c(R_j)) = 1 - p(a(R_i) = 1|c(R_j))$.) Moreover, many of the $p(a(R_i) = 1|c(R_j))$ parameters will be constrained by the grammar to be 0, because even though R_i and R_j are in the same alternative rule set, the grammar is structured in such a way that the unification constraints on both rules could never be satisfied simultaneously in any partial derivation. In a grammar with 1,000 rules, it would be rare for any single rule to have more than perhaps 25 true possible alternatives, so 25,000 is probably an upper bound on the total number of parameters we would have to directly estimate for syntactic probabilities.

This model looks quite tractable, but so far it incorporates only syntactic constraints, not lexical constraints. These are not normally expressed directly in unification-based grammar rules, but are represented using some additional formal mechanism, such as sortal restrictions on semantics representations (Alshawi, 1992, Chapter 9). This suggests an elegant factoring of syntactic and lexical constraints, based on viewing grammars, not as merely correlating strings with syntactic structures, but as mapping between semantic representations and syntactic representations. The kind of unification-based grammar formalism we have been discussing is easily extended to associate possible semantic representations, or logical forms, with each syntactic structure (Alshawi, 1992, Chapter 5; Moore, 1995, Chapter 10), so that a combined syntactic/semantic language model could be defined as

$$p(W) = \sum_L p(W|L) \cdot p(L)$$

In this formula, W is a word string and L is a logical form, so to apply this model, we would find all logical forms that could be expressed by a given word string, and sum the products of the

prior probability of each logical form (which would incorporate lexical semantic constraints) and the probability of the word string given the logical form. For a particular word string and logical form, we can compute an estimate of the probability of the word given the logical form exactly as we proposed above, by summing over all grammatical analyses that associate the word string and the logical form. Nothing in the model needs to be changed to extend it to incorporate constraints between syntax and logical form, except that the notion of a rule being applicable or correct must take into account constraints supplied by the logical form at a particular stage of the derivation.

Since logical forms are typically represented as nested functional expressions, if they are annotated with sortal information, as they are in the Core Language Engine and Gemini, it is fairly straightforward to recursively estimate their prior probability by (1) conditioning the probability estimates for the functor of a term on the semantic sort of the term, and (2) conditioning the probability estimates for the sorts of arguments of a functor on the functor and the semantic sorts of sibling arguments. It is often the case that the functors used in logical form representations are largely limited to two arguments, in which case the number of parameters needed to estimate to probability of the semantic representations would be on the order of the number needed to estimate a trigram language model.

Thus, the combined syntactic and semantic model we have sketched should require on the order of the same number of parameters to estimate as an n-gram language model, but have the advantage of incorporating both long-distance syntactic constraints and lexical constraints that are conditioned on semantically meaningful grammatical relationships rather than simple word adjacencies.

7 Summary

We have explored in detail two different kinds of language model for speech recognition that incorporate constraints from natural-language grammars. For situations in which training data for statistical models is not available, we have described methods for using complex natural-language grammars by compiling them into context-free grammars that can be efficiently incorporated into the search architecture of a commercially available real-time speech recognizer. If adequate training data for statistical models is available, we acknowledge that conventional n-gram models are difficult to improve on, but we have described one experiment in which a hybrid language model incorporating both n-gram statistics and constraints from a qualitative natural-language grammar was able to substantially improve the performance of a state-of-the-art conventional HMM-based recognizer. Finally, we have presented more speculative work on statistical context-free grammars, and described how this work might be extended to incorporate all the constraints of a complex unification-based grammar.

References

Alshawi, H. (ed.) (1992) *The Core Language Engine*, The MIT Press, Cambridge, Massachusetts.

Black, E., F. Jelinek, J. Lafferty, D. M. Magerman, R. Mercer, and S. Roukos (1993) "Towards History-based Grammars: Using Richer Models for Probabilistic Parsing," in *Proceedings of*

the 31st Annual Meeting of the Association for Computational Linguistics, Columbus, Ohio, pp. 31–37.

Boisen, S., Y.-L. Chow, A. Haas, R. Ingria, S. Roukos, and D. Stallard (1989) "The BBN Spoken Language System," in *Proceedings Speech and Natural Language Workshop February 1989*, Philadelphia, Pennsylvania, pp. 106–111 (Morgan Kaufmann Publishers, Inc., San Mateo, California).

Briscoe, T., and J. Carroll (1993) "Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars," *Computational Linguistics*, Vol. 19, No. 1, pp. 25–59.

Charniak, E. (1995) "Parsing with Context-Free Grammars and Word Statistics," Technical Report CS–95–28, Department of Computer Science, Brown University, Providence, Rhode Island.

Chomsky, N. (1957) *Syntactic Structures*, Mouton & Co., The Hague, Holland.

Chow, Y.-L., and S. Roukos (1989) "Speech Understanding Using a Unification Grammar," in *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, Glasgow, Scotland, pp. 727–730.

Chow, Y.-L., and R. Schwartz (1989) "The N-Best Algorithm: An Efficient Procedure for Finding Top N Sentence Hypotheses," in *Proceedings Speech and Natural Language Workshop October 1989*, Cape Cod, Massachusetts, pp. 199–202 (Morgan Kaufmann Publishers, Inc., San Mateo, California).

Church, K. W. (1989) "Syntactic Parsing May Not Help Speech Recognition Very Much," in *Spoken Language Systems Working Notes*, AAAI Spring Symposium Series, Stanford, California, pp. 6–9.

Culy, C. (1985) "The Complexity of the Vocabulary of Bambara," *Linguistics and Philosophy*, Vol. 8, No. 3, pp. 345–351.

Collins, M. J. (1996) "A New Statistical Parser Based on Bigram Lexical Dependencies," in *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, California, pp. 184–191.

Collins, M. J. (1997) "Three Generative, Lexicalized Models for Statistical Parsing," in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain, pp. 16–23.

Dowding, J., J. M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran (1993) "Gemini: A Natural Language System for Spoken-Language Understanding," in *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, pp. 54–61.

Dowding, J., R. Moore, F. Andry, and D. Moran (1994) "Interleaving Syntax and Semantics in an Efficient Bottom-Up Parser," in *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico, pp. 110–116.

Dupont, P. (1993) “Dynamic Use of Syntactical Knowledge in Continuous Speech Recognition,” in *Proceedings Third European Conference on Speech Communication and Technology*, Berlin, Germany, pp. 1959–1962.

Dupont, P., and D. Snyers (1989) “Efficient Dynamic Expansion of Context-Free Grammar in Speech Recognition,” in *Overview of Research in Speech Recognition at PRLB in 1988*, Philips Research Laboratory, Brussels, Belgium, pp. 32–68.

Goodman J. (1997) “Probabilistic Feature Grammars,” in *Proceedings of the International Workshop on Parsing Technologies*, Boston, Massachusetts.

Grishman, R., and J. Sterling (1993) “Smoothing of Automatically Generated Selectional Constraints,” in *Proceedings Human Language Technology Workshop*, Plainsboro, New Jersey, pp. 254–259 (Morgan Kaufmann Publishers, Inc., San Francisco, California).

Hopcroft, J., and J. Ullman (1979) *Introduction to Automata Theory, Languages, and Computation*, Addison Wesley Publishing Company, Reading, Massachusetts.

Huang, X., A. Acero, F. Alleva, M.-Y. Hwang, L. Jiang, and M. Mahajan (1995) “Microsoft Windows Highly Intelligent Speech Recognizer: Whisper,” in *Proceedings 1995 International Conference on Acoustics, Speech, and Signal Processing*, Detroit, Michigan, pp. 93-96.

Lafferty, J., D. Sleator, and D. Temperley (1992) “Grammatical Trigrams: A Probabilistic Model of Link Grammar,” *Probabilistic Approaches to Natural Language Working Notes*, AAAI Fall Symposium Series, Cambridge, Massachusetts, pp. 89–97.

Magerman, D. M. (1995) “Statistical Decision-Tree Models for Parsing,” in *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, Massachusetts, pp. 276–283.

Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz (1993) “Building a Large Annotated Corpus of English: The Penn Treebank,” *Computational Linguistics*, Vol. 19, No. 2, pp. 313–330.

Mohri, M. (1997) “Finite-State Transducers in Language and Speech Processing,” *Computational Linguistics*, Vol. 23, No. 2, pp. 269–311.

Moore, R. (1995) *Logic and Representation*, CSLI Publications, Center for the Study of Language and Information, Stanford University, Stanford, California.

Moore, R., M. Cohen, V. Abrash, D. Appelt, H. Bratt, J. Butzberger, L. Cherny, J. Dowding, H. Franco, J. M. Gawron, and D. Moran (1994) “SRI’s Recent Progress on the ATIS Task,” in *Proceedings of the Spoken Language Technology Workshop*, Plainsboro, New Jersey, pp. 72–75 (Morgan Kaufmann Publishers, Inc., San Francisco, California).

Moore, R., D. Appelt, J. Dowding, J. M. Gawron, and D. Moran (1995) “Combining Linguistic and Statistical Knowledge Sources in Natural-Language Processing for ATIS,” in *Proceedings of the Spoken Language Systems Technology Workshop*, Austin, Texas, pp. 261–264 (Morgan Kaufmann Publishers, Inc., San Francisco, California).

Moore R., J. Dowding, H. Bratt, J. M. Gawron, Y. Gorfu, and A. Cheyer (1997) "CommandTalk: A Spoken-Language Interface for Battlefield Simulations," in *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Association for Computational Linguistics, Washington, DC, pp. 1–7.

Moore, R., F. Pereira, and H. Murveit (1989) "Integrating Speech and Natural-Language Processing," in *Proceedings Speech and Natural Language Workshop February 1989*, Philadelphia, Pennsylvania, pp. 243–247 (Morgan Kaufmann Publishers, Inc., San Mateo, California).

Murveit, H., and R. Moore (1990) "Integrating Natural Language Constraints into HMM-based Speech Recognition," in *Proceedings 1990 International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, New Mexico, pp. 573–576.

Nuance Communications (1996) *Nuance Speech Recognition System, Version 5, Developer's Manual*, Menlo Park, California.

Pullum, G. K., and G. Gazdar (1982) "Natural Languages and Context-Free Languages," *Linguistics and Philosophy*, Vol. 4, No. 4, pp. 471–504.

Schwartz, R., and S. Austin (1990) "Efficient, High-Performance Algorithms for N-Best Search," in *Proceedings Speech and Natural Language Workshop June 1990*, Hidden Valley, Pennsylvania, pp. 6–11 (Morgan Kaufmann Publishers, Inc., San Mateo, California).

Shieber, S. M. (1985) "Evidence Against the Context-Freeness of Natural Language," *Linguistics and Philosophy*, Vol. 8, No. 3, pp. 333–343.

Woods, W. A. (1970) "Transition Network Grammars for Natural Language Analysis," *Communications of the ACM*, Vol. 13, No. 10, pp. 591–606.