Integrating Experimental Models of Syntax, Phonology, and Accent/Dialect in a Speech Recognizer

Daniel Jurafsky, Chuck Wooters[*], Gary Tajchman,
Jonathan Segal, Andreas Stolcke, and Nelson Morgan

International Computer Science Institute
and
University of California at Berkeley
<jurafsky|wooters|tajchman|jsegal|stolcke|morgan>@icsi.berkeley.edu

As the field of speech understanding matures, and particularly as the quality of front-end and phonetic components improves, researchers have begun to explore ways to add new kinds of language knowledge to the recognition process. Such work includes augmenting recognizers with models of contextual dependencies (Cohen 1989; Phillips *et al.* 1991), more advanced models of syntax, (Seneff *et al.* 1992; Kai & Nakagawa 1992), and gender information (Murveit *et al.* 1991). This new direction is being developed at ICSI in the context of the Berkeley Restaurant Project (BeRP), a medium-vocabulary (1300 word), speaker-independent, spontaneous continuous-speech understanding system. The primary function of BeRP is to serve as a testbed for a number of our speech-related research projects, including robust feature extraction, connectionist speech recognition, automatic induction of multiple-pronunciation lexicons, foreign accent detection and modeling, and the use of advanced language models.

The BeRP system functions as a knowledge consultant whose domain is restaurants in the city of Berkeley, California. As a knowledge consultant, it draws inspiration from earlier consultants like VOYAGER (Zue *et al.* 1991). Users ask spoken language questions of BeRP, which, in a mixed initiative fashion, directs questions to the user and then queries a database of restaurants and gives advice to the user, based on such use criteria as cost, type of food, and location.

This paper describes three preliminary experiments in adding new language knowledge to the recognizer BeRP:

- an automatically-induced multiple-pronunciation lexicon, which improves word error in the recognizer from 40.6% to 32.1%.
- a stochastic context-free grammar (SCFG)[1] with a probabilistic Earley-based parser which provides word transition probabilities at each frame ('tight coupling'), improving word error rate from 33.7% (bigram) to 29.6% (SCFG).
- algorithms for detecting and modeling foreign accent, which in prototype form give 67.9% accuracy in distinguishing German from American speakers of English.

Figure 1 gives an overview of the architecture.

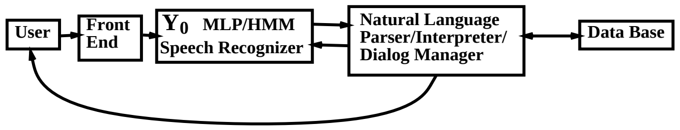

User
Front End
Y_0 MLP/HMM Speech Recognizer
Natural Language Parser/Interpreter/ Dialog Manager
Data Base

Figure 1: The BeRP Architecture

Figure 2 gives some statistics which give the reader a feeling for the BeRP system as implemented. The numbers do not yet reflect the incorporation of our tight-coupling or accent results.[2] Figure 3 illustrates a sample interaction between user and the current implementation of BeRP.

Corpus	2621 sentences spontaneous speech	
Vocabulary	1064 words	
Data Base	1 database table, 150 restaurants	
Bigram	Perplexity 10.7 with 77% coverage	
Grammar	1177 handwritten SCFG rules	
Implementation	18,000 lines of C++	
Performance	Recognition	32.1% error
	Parsing	63% training 51% test
	Understanding	34% error

Figure 2: BeRP Status in April 1994

[*]Currently at Dept. of Defense

[1]An SCFG is a context-free grammar in which each rule is augmented with the conditional probability of the nonterminal on the LHS expanding to the right-hand side.

[2]We don't yet have performance numbers on our most recent system, which has a larger corpus of 5350 sentences, a vocabulary of 1329 words, and a grammar of 1289 sentences with a 70% coverage of the training set and 54% on the test set. See Jurafsky *et al.* (1994) for these results. Note that the MLP phonetic likelihood estimator is bootstrapped with additional sentences from the TIMIT database as well.

Computer: `''Welcome to BeRP. How may I help you?''`

User: "I'm looking for a Thai restaurant"

Computer: `''When would you like to go out?''`

User: "I'd like to go for lunch on Saturday"

Computer: `''How far from ICSI are you willing to travel?''`

User: "anywhere within walking distance"

Computer: `''How much money do you want to spend?''`

User: "doesn't matter"

Computer: `''Here are some restaurants that meet your requirements:''`

```
Name                | Address            | Cost | Min. away
------------------------------------------------------------
       THAI BAR-B-Q | 1958   SHATTUCK AV |  $$  | 5
PLEARN THAI CUISINE | 2050 UNIVERSITY AV |  $$  | 5
BERKELEY THAI HOUSE | 2511  CHANNING WAY |  $$  | 20
       SIAM CUISINE | 1181 UNIVERSITY AV | $$$  | 18

You may ask for additional information on any of
the above restaurants.

Ready for a new query.
```

Figure 3: A sample BeRP interaction

1 The Basic Architecture of BeRP

The BeRP recognizer consists of five components: the *front end*, currently RASTA-PLP, the HMM pronunciation *lexicon*, the *phonetic likelihood estimator*, a MLP which computes a phonetic likelihood for each input frame, the Y_0 *Viterbi decoder*, and the *natural language backend*, including a database of restaurants.

BeRP uses RASTA-PLP (Hermansky *et al.* 1992) as its front end. RASTA-PLP is a speech analysis technique that is robust to steady-state spectral factors in speech such as those imposed by different communication channels (i.e., different microphones). We believe this is an important feature for the front end, especially when using databases that have been collected under different recording conditions than those at ICSI.

The BeRP system uses a discriminatively-trained Multi-Layer Perceptron (MLP) to estimate emission probabilities that are used in an iterative Viterbi procedure (similar in spirit to the segmental k-means algorithm); an MLP trained on phonetically hand-labeled speech (TIMIT) is used to produce the initial state alignments for the BeRP training data. Bourlard & Morgan (1991) and Renals *et al.* (1991) show that with a few assumptions, an MLP may be viewed as estimating the probability $P(q|x)$ where q is a subword model (or a state of a subword model) and x is the input acoustic speech data. We can then compute the likelihood $P(x|q)$ needed by the Viterbi algorithm by dividing by the prior $P(q)$, according to Bayes rule; we can ignore $P(x)$ since it is constant in time-synchronous Viterbi:

$$P(x \mid q) = \frac{P(q \mid x)P(x)}{P(q)} \tag{1}$$

In this hybrid HMM/MLP recognizer, it was shown that these estimates led to improved performance over standard estimation techniques when a fairly simple HMM was used. Current results on the speaker independent DARPA Resource Management database for MLP monophone estimators continue to support this contention (Cohen *et al.* 1992), and have yielded reasonable performance (4–6% word error with the standard perplexity 60 wordpair grammar).

The architecture of the MLP phonetic recognizer (like our monophone Resource Management recognizer (Cohen *et al.* 1992)) consists of a simple three-layer feed forward MLP trained with the back-propagation algorithm (see Figure 4). The input layer consists of 9 frames of input speech data. Each frame, representing 10 msec of speech, is encoded by 9 RASTA PLP coefficients, 9 delta-RASTA PLP coefficients, and an energy term and a delta-energy term. Typically, we use 500-1000 hidden units. The output layer has 61 units, one for each of the context-independent phonetic classes used in our lexicon. The MLP is bootstrapped on the TIMIT database, along with the over 5000 sentences of spontaneous speech that we have collected in the BeRP Corpus.

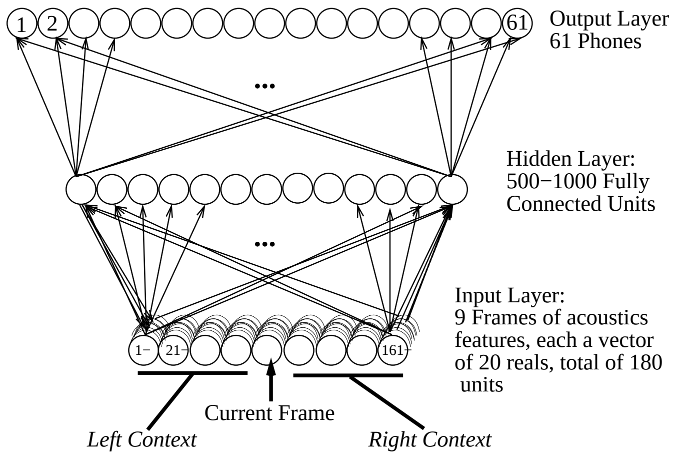


Figure 4: Phonetic Likelihood Estimator

The Y_0 recognizer uses a bigram language model (LM). For each pair of words (w_i, w_j) in the vocabulary, the language model stores $P(w_j|w_i)$. With a vocabulary of over 1300 words, there are over 1.7 million possible bigram pairs; since this number is two orders of magnitude greater than the number of tokens in our corpus, the corpus by itself is quite insufficient to estimate the probabilities, and so we use various smoothing methods, including the use of the SCFG LM (see below) as a bigram smoother.

The BeRP natural language backend accepts as input the word strings passed to it by the Y_0 recognizer, and produces both database queries and appropriate responses to the user as output.

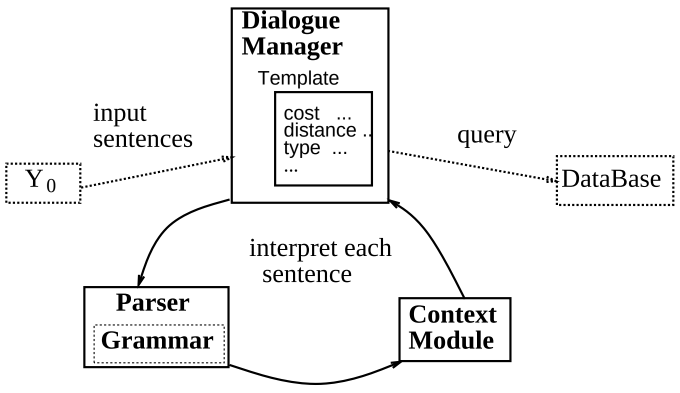


Figure 5: The BeRP I Back-End

Figure 5 shows the internal architecture of the natural language processing component of BeRP, ignoring tight coupling. The architecture is controlled by a template-filling *dialog manager*, which asks questions of the user in order to determine certain restaurant features, which are used to fill a query-template, and thus to query the database. The current template includes such information as restaurant cost, distance, food type, and business hours. For each template slot the system prompts the user with a question. Let us examine the processing of one input sentence from the BeRP corpus, in which the system asks the following question:

```
[Computer:] "HOW CAN I HELP YOU?"
[User:] "I'd like to have Indian food today"
```

Note that the user's response should give the system two kinds of information – *food-type* and *date*. First, the recognizer computes and passes to the parser the most likely word string, in this case:

```
i'd like to have indian food today
```

Next the bottom-up stochastic parser computes all possible parses and semantic interpretations of the input, along with their grammatical probabilities. It uses a stochastic context-free attribute grammar in which simple semantic rules are encoded in the attributes, written in a generalization of the PostQuel database query language used by the BeRP database backend. The 1289-rule grammar was written by hand, but rule probabilities are learned from the BeRP corpus with the EM algorithm.[3]

The 'semantic' grammar encodes more information in the context-free portion than is common in purely 'syntactic' grammars by allowing very specific non-terminal symbols, which lowers the perplexity of the grammar and allows us to avoid slow unification algorithms.[4]

The parse tree for the input above is shown in Figure 6.

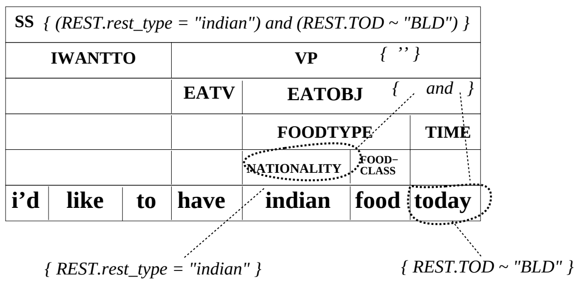


Figure 6: The parse-tree produced by the interpreter

Each node of the parse tree is automatically annotated with a a partial semantics. The semantics for the entire sentence is the attribute for the top-level symbol *ss*, and thus can be read right off of the parse tree for the complete sentence, and passed to the **context module**, which fills out all context-dependent and scope-dependent operators, such as temporal deictics ("now", "today") and negation ("not far"). The final string below specifies that the user is interested in an Indian restaurant which is open Monday ("REST.mon") for any meal (breakfast ("B"), lunch ("L") or dinner ("D")).

```
(REST.rest_type = "INDIAN") and (REST.mon ˜ "[BLD]")
```

This semantics will now fill both the food-type and time slots, and will become part of the eventual database query, which will be completed when the template is full.

An important consideration in the parser is robustness to grammatical and lexical gaps as well as recognizer errors. The BeRP parser bases its robustness on two features: the use of a bottom-up (CYK) algorithm for parsing, and the use of a greedy semantic heuristic for combining parse fragments (see Jurafsky *et al.* (1994) for details).

2 Inducing Multiple-Pronunciation Lexicons

Having described the basic architecture of the BeRP system, we turn to the first of our three uses of more sophisticated linguistic information, the automatically-induced multiple-pronunciation lexicon. A number of HMM-based systems allow words to have multiple pronunciations, in which a single complex HMM has multiple possible phone paths, indicating different possible pronunciations. These multiple-pronunciation models can be

[3]Using an algorithm suggested by (Fujisaki *et al.* 1991) which is a simpler but less efficient algorithm than the standard Inside-Outside method (Baker 1979).

[4]However we do expect to move to a feature-based grammar.

generated by phonological rules. For example, Cohen (1989) describes a semi-automatic way of generating the models in which allophonic rules are manually developed, and then automatically applied to a set of baseform or "dictionary" pronunciations, to automatically generate multiple pronunciation HMMs for any word in the dictionary. Others (Withgott & Chen 1993; Riley 1991) have proposed using decision trees, trained on phonetically transcribed data, to map the phonemes from a baseform pronunciation into contextually appropriate allophones.

The method used in the BeRP system to automatically generate a multiple-pronunciation lexicon differs from both of these techniques. Our approach is to *induce* a word model directly from a set of pronunciations for the word. Our completely automatic *bottom-up* approach is in contrast to the semi-automatic *top-down* technique introduced by Cohen, but is quite similar in spirit to the decision tree techniques.

The basis of the bottom-up technique used in the BeRP system is an algorithm proposed by Stolcke & Omohundro (1994) for automatically inducing an HMM topology from examples. The essential idea is to begin with a very specific HMM which produces exactly the set of input strings, constructed by stringing together each possible pronunciation between a start and end state. Thus the start state has as many outgoing transitions as there are pronunciations, and each pronunciation is represented by a unique path within the HMM.

Next, this initial model is simplified and generalized by repeatedly *merging* states. Merging two states q_1 and q_2 means replacing them by a new state r whose transitions and emission probabilities are weighted mixtures of those for q_1 and q_2, producing a simpler (i.e., smaller) but more general model. In generalizing a model, we are trading off model likelihood against a bias toward simpler models. Since we are looking for the model with the maximum a posteriori probability $P(M|x)$, which is proportional to the product of the model prior $P(M)$ and the likelihood of the data $P(x|M)$, we can use the prior probability distribution over HMMs to prefer simpler models. We can then repeatedly merge states until we reach a model with (locally) maximum posterior probability.

Because the multiple-pronunciation models generated by this model-merging approach are still quite complex, we apply the further step of pruning out unlikely states or paths from the model to produce a model with a more manageable number of states.

The benefit of this technique is that it is completely automatic. However, it cannot currently be used to generate HMMs of words that are not contained in the training data, as can be done with Cohen's rule-based approach, and with the decision tree methods. Future developments include using a decision tree and/or an MLP to map from baseform pronunciations to multiple-pronunciation word models for words in the BeRP lexicon that do not occur in the TIMIT database.

Figure 7 shows the initial multiple-pronunciation model for the word *waiting*, computing from the samples in the TIMIT and other corpora and dictionaries. Figure 8 shows how the multiple-pronunciation HMM for "waiting" is merged, and then pruned. The top HMM has merged together all the examples of the word, but has not pruned any unlikely paths. The bottom HMM has pruned out the unlikely paths.

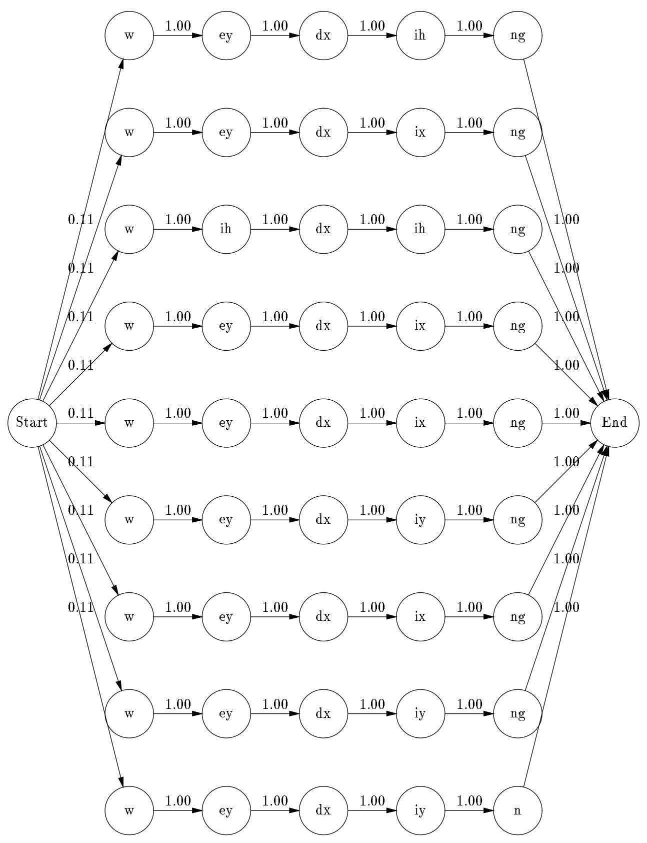


Figure 7: Unmerged, unpruned multiple-pronunciation HMM for the word "waiting"

Table 1 shows the significant improvement which our multiple pronunciation models provide over the single pronunciation models. (Note that our most recent results reported in §5 are somewhat improved from these.)

System	Word Error	Ins	Dels	Subs
Single Pron	40.6	5.2	10.3	25.1
Multi-Pron	32.1	7.3	5.9	18.9

Table 1: Performance of Pronunciation Models.

3 Accent/Dialect Detection and Modeling

Our second experiment with BeRP was the addition of knowledge of foreign accents. The users of the BeRP system, like the citizens of the United States, speak English with a broad variety of accents. These include regional United States dialects of the type included in the TIMIT database, but, more significantly, include large numbers of speakers with German-, British-, Italian-, Spanish- and Chinese-accented English, as well as many others.

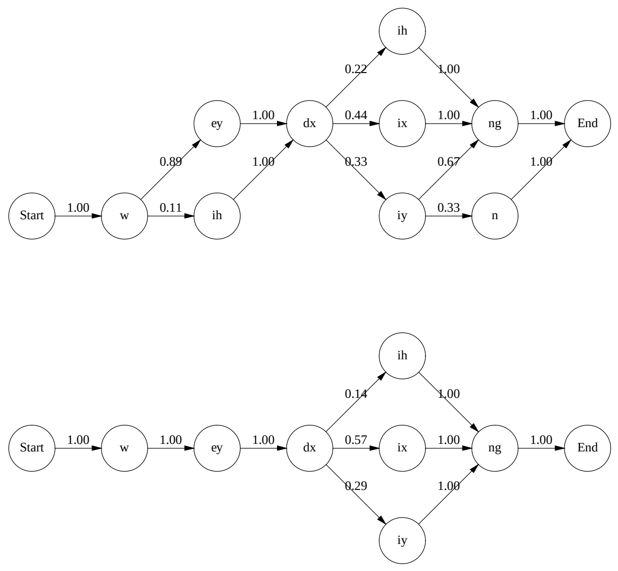


Figure 8: Merging and pruning for "waiting" multiple-pronunciation HMM

Robustness to foreign accents is a seriously underexplored area of speech recognition. For example, we have shown that a system trained only on American English speech has significantly more errors on speakers with non-native accents. Specifically, when we tested our (American-trained) BeRP system tuned for good performance with American speakers with an accent-balanced test set (277 American-accent sentences and 277 German-accent sentences) we incur a substantial 25% more errors (compare American to total test set numbers for the American optimized system) as shown in Table 2. No matter how the system is optimized, there is a penalty to pay.

	Test Set		
Optimization	American	German	Total
American	32.7	49.7	40.8
German	38.2	40.0	39.1
Total	35.2	41.9	38.4

Table 2: Percent word error for various tunings of a system trained only on American speech.

Useful speech recognition systems must have the ability to deal with these and other sources of speaker variability. Most current recognition systems deal with this problem implicitly by taking pains to use training data with a broad coverage of speaker groups. For instance, most of the large speech corpora provided by NIST were specifically designed to include speech from major dialect regions of the United States. While the large-database approach to modeling speaker variability has proved quite useful and has been widely accepted, by itself, it provides very little guidance for making significant progress in dealing with speaker-group variability. Cohen (1989) proposed an alternative approach in his study of pronunciation modeling: using *extrinsic information* about a speaker, such as dialect region and gender, to help predict the speaker's pronunciation.

Our approach to accent/dialect detection and modeling is similar in spirit to the approach Cohen and his colleagues have introduced. The specific goal is to build into our recognizer the ability to both automatically *detect* and *model* foreign accents of English, while retaining high performance with native American English speech.

Our work in foreign accent detection so far has focused on using two sources of information about accent, acoustic-phonetic information, and syntactic information. A baseline acoustic-phonetic system that uses a very simple MLP training procedure to distinguish between two accents of English (German and American), has provided evidence that, minimally, there is easily extracted information at the acoustic-phonetic level exposing the accent of a given speaker.

Our initial approach for using acoustic-phonetic information involves training a simple MLP with one output unit for each accent (the same architecture as in Figure 4 but with only two output units). The accent of the speaker who produced the input frame is used as the desired output. This essentially treats each frame, no matter what sentence or speaker it came from, as an independent instance of the accent that produced it. The goal of this type of training is to enable the MLP to locate those acoustic-phonetic events that are highly indicative of one accent or the other.

The frame by frame MLP outputs for each accent are accumulated over an entire sentence using the formula

$$\log P(accent|S) \approx \sum_{t=0}^{N} \log O_t^A + C \qquad (2)$$

where $P(accent|S)$ is the probability of an accent given the sentence S, O_t^A is the activation of the American output unit at time t, C is a constant independent of accent for the balanced test set, and the sum is over the N frames of a sentence. By multiplying the MLP's frame level probabilities of each accent, we obtain its estimate of the most probable accent given the sentence. To date, our best speaker independent accent identification performance is 67.9% correct at the sentence level with the accent-balanced test set. We are currently investigating approaches that target specific phones (context-independent and context-dependent) that are reliably different between accents.

With respect to our primary concern, better recognition performance, we have found that knowing the speaker's accent can indeed reduce the word error rate.

We used two MLP's, one trained only on German data with a German pronunciation lexicon, and one trained only on American data with an American pronunciation lexicon.[5] We tried various methods to choose between the output from either system, including the speaker's true accent, the accent-MLP's estimate

[5]Both were initialized with a TIMIT-trained net, which had exclusively American training.

System Choice Method	Percent Word Error
True Accent	30.8
Accent ID	33.8
American Only	33.4
German Only	37.3

Table 3: Percent word error on the accent-balanced test set for several methods that choose between an American-tuned system and a German-tuned system on a sentence by sentence basis.

of the speakers accent, and simply picking either system for all sentences.

The best performance is obtained when given perfect knowledge of the speaker's accent. Picking between the two tuned systems based on the speaker's true accent significantly outperforms the other methods for combining these two systems. Our current level of accent identification performance based only on acoustic-phonetic information, however, is not yet good enough to outperform the American-tuned system. Further, when we pool both the German and American training data together, we get very good performance on the balanced test set (30.6% word error). This pooled-data error rate can not be directly compared to the other results in Table 3 because the pooled-data MLP was trained on twice as much data. But this result does demonstrate that simply including the different accents in one training data set is a reasonably effective way to deal with accent variability. Our goal is to show, unequivocally, that information about the speaker's accent can be used to modify the system and result in lower word error rates, and better semantic recognition. These results, constituting our baseline acoustic-phonetic accent system, show that for systems trained with equivalent amounts of data, accent can be used to give better system performance.

Similarly, we developed a baseline syntactic system that can be easily trained and does provide information about a speaker's accent. Of course syntactic information cannot give direct information about phonetics; our intention is to use syntactic information about the slightly different dialects used by native versus non-native speakers to help distinguish them. Once we determine that a speaker is likely to be German based on the syntactic characteristics of his or her dialect, we can use this information to help select appropriate pronunciation models.

One intuitive way to detect this type of syntactic dialect difference would be to mark certain constructions in the grammar as German, and others as particularly American. Then speakers who use more German constructions can be flagged as German, and conversely for Americans. Our formal model of this intuition is to split our corpus of 5350 sentences into an American and German set, and to train different stochastic LMs for the German and American speakers. Then for each speaker we can use these models to compute the posterior probability that the speaker is German. Table 4 shows some sample SCFG probabilities computed over our German and American corpora.

The differing probabilities show a number of interesting dialect facts. For example, Americans were twice as likely as Germans to put the word "please" at the end of a sentence. But Germans were *200 times* as likely as Americans to start a sentence with "please". This fact correlates well with the syntax of German, in which the word "bitte" (corresponding to English "please"), usually occurs sentence-initially. Note that Germans were about 4 times as likely as Americans to use the verb group "would like to", and 8 times as likely to use the contraction "wanna". Americans were more likely to say "want to" or, not shown here, to contract "I'd like to".

Rules	**American**	**German**
ss ⇒ s 'please'	0.046	0.020
ss ⇒ OPENING s	0.065	0.035
OPENING ⇒ 'please'	0.00057	0.10
OPENING ⇒ 'well'	0.0095	0.0045
WANTTO ⇒ 'want' 'to'	0.62	0.44
WANTTO ⇒ 'would' 'like' 'to'	0.10	0.38
WANTTO ⇒ 'wanna'	0.010	0.082

Table 4: A grammar trained on German and American corpora

Given these grammars and the probabilistic chart parser discussed above, it is possible to compute for each sentence the likelihood that it is generated by each grammar, i.e., P(sentence | German) or P(sentence | American). Now, using Bayes' rule, we compute the posterior probability that the sentence was spoken by a German as follows:

$$P(German|s) = \frac{P(s|German)P(German)}{P(s)} \tag{3}$$

The priors P(German) and P(American) can be estimated from relative frequency of sentences in the corpora. We have built an initial version of this accent detector, training German and American grammar priors and then computing a posterior dialect probability for each sentence in our test corpus. The results for our baseline experiments are shown in Table 5.[6] Using a confidence measure and a threshold value for rejecting ambiguous sentences we obtained results at 0 percent sentence rejection, and 50 percent sentence rejection. We found no benefit to combining the acoustic and syntactic probabilities at the 0 percent rejection criterion, but there does seem to be a benefit at the 50 percent rejection point.

Percent Sents Used	Acoustic	Syntactic	Combined
100	62	60	62
50	70	68	73

Table 5: Percent correct accent identification at sentence level.

4 Tight Coupling

A number of researchers have proposed ways to use natural-language-backend information in the speech recognition process. Moore *et al.* (1989) used a unification-based CFG to generate word transitions for a Viterbi recognizer. Goodine

[6]These results were obtained with an early version of the acoustic-phonetic accent identification method.

et al. (1991) describe a system which uses the CFG-based TINA parser to predict next words for the SUMMIT speech recognizer, Kita & Ward (1991) used a CFG to filter to bigram follow-sets for the Sphinx recognizer. In all these cases, the CFG was used to generate or filter the word-transition list, but not to assign probabilities. Goddeau (1992) extended these results by using a probabilistic LR parser to actually produce word-transition probabilities. In this section, we discuss our tight coupling model, which augments a probabilistic version of the Earley algorithm (Stolcke 1993) to compute word transition probabilities from a stochastic context-free grammar (SCFG), significantly improving recognition word error from 33.7% (bigram) to 29.6% (SCFG).

The fundamental reason to use a more advanced LM is to provide a better match to the input data, and thus lower the language perplexity as seen by the recognizer, and improve recognition performance. Currently, however, structurally simple grammars like *n*-grams have outperformed more complex grammars because more complex grammars are often not probabilistic, not trained on large corpora, do not integrate semantic information on-line, and are not robust. In order to overcome these problems, the BeRP parsers are based on SCFGs trained on a corpus of thousands of hand-transcribed sentences, directly incorporate semantic features into the context-free rules, and include fragment combination and bigram back-off for robustness.

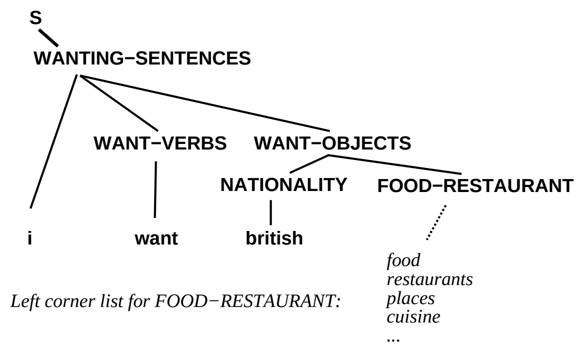
We have experimented with a number of ways to use the information provided by the SCFG:

1. Use the SCFG to smooth the bigram grammar, by taking our original corpus, adding a pseudo-corpus generated from the SCFG, and building the bigram with Monte-Carlo sampling on this joint corpus.
2. Use the SCFG to smooth the bigram grammar by generating the *characteristic* bigram for the SCFG in closed form.
3. Use the SCFG directly to provide word transition probabilities on each frame.
4. Use a mixture of the SCFG and bigram probabilities directly to provide word transition probabilities on each frame.

In the first two methods, we use the SCFG to smooth the bigram grammar, and then use this improved bigram grammar in the recognizer. The first method extends an idea of Zue *et al.* (1991), who used an advanced language model to generate random sentences from which to train a word-pair model. We extended this idea to generation of bigrams by Monte-Carlo sampling, by using the SCFG-based parser to generate a pseudo-corpus of 200,000 sentences, adding in our regular BeRP corpus, and then using our standard bigram-building tools on the combined corpus.

In the second method, we have shown (Stolcke & Segal 1994) that it is possible to generate a bigram from a stochastic context-free grammar *directly*, by computing its *characteristic n-gram* in closed form. The method computes the expected bigram counts for strings generated by each of the nonterminals in the grammar by solving a system of linear equations derived from the grammar rule probabilities. We have implemented this algorithm recently, and will use it instead of the Monte Carlo method to generate our future bigrams.

In the third method, we use the SCFG directly as the LM for the recognizer. We focus on this method for the rest of this section. We begin by abstracting away from probabilities. Consider the problem of using a CFG to produce a follow-set, given a prefix string. For example, if the recognizer passes the string *I want British* to the parser, it will produce the follow words "food", "restaurants", "places", "cuisine", etc. The parser parses the prefix string, and then looks at every non-terminal symbol that the Earley parser is predicting next. For each such non-terminal, we look up its *left-corner* list – the list of terminal symbols which the non-terminal can generate on the left fringe of some parse tree. This list can be computed in advance.



The recognizer needs more than just follow sets, however, it needs actual conditional probabilities. In this case, it needs the various probabilities P('food' | 'I want British'), P('restaurants' | 'I want British') etc.; i.e., for each word w_i in the follow set, we need to compute

$$P(w_i|w_1w_2\ldots w_{i-1}) \tag{4}$$

To compute these probabilities, we first augment the left corner list to produce the probability that a given non-terminal expands to a terminal. For a given pair of symbols $< X, w >$, where X is a non-terminal and w is a terminal, the left-corner probability is the probability that X generates some string which begins with w. Jelinek & Lafferty (1991) give an algorithm for computing this left-corner probability for every pair of non-terminals and terminals in the grammar with a single matrix-inversion.

If all sentences were unambiguous, this would be sufficient to produce the correct transition probabilities. However, sentences are ambiguous. Because of this, there will be multiple parses for each prefix, and hence we will need to combine the left-corner probabilities for non-terminals from different parses. We can do this by weighting the follow-set for each parse, or derivation, by the probability of the derivation.

$$P(w_i|w_1\ldots w_{i-1}) = \sum_{d \in derivations} P(d)P(w_i|w_1\ldots w_{i-1}, d) \tag{5}$$

Thus the parser must be able to compute *prefix probabilities* for derivations of input strings. For a given parse, the prefix probability is just the product of the probabilities of all the rules used in the parse. In order to compute this probability efficiently, we augment our probabilistic chart parser by annotating each edge of the chart with quantities: a prefix probability and an inside probability. Each edge-creation action computes the inside probability and prefix probability for the new edge from the old edges and the grammar rule probabilities. Readers with interest in the details of this probabilistic Earley computation are referred to Stolcke (1993), which extends the simpler prefix algorithm used in BeRP to deal with left-recursive grammars and unit productions.

We have described how the parser is able to compute follow-set probabilities for each string that is passed to it by the recognizer. We turn now to the tight-coupling interface. For each frame, the Y_0 decoder must compute word strings to pass to the parser. A bigram-based recognizer would simply look up the bigram transition probability for each word that can end at the current frame. Since an SCFG-based recognizer will use the entire prefix to compute the transition probabilities, the recognizer must perform a *backtrace* to determine the prefix associated with the word. The optimal algorithm would search through the Viterbi array to find the N-best word strings, and pass each to the recognizer. In practice, we use a simple (but we believe poor) approximation to the N-best algorithm, in which at each 10 ms frame, Y_0 finds the 10 words most likely to end, and for each performs a single backtrace to find 10 strings. Each of these is passed to the parser, which computes a probability vector over the follow-set words. The recognizer then uses each of these probabilities as the transition probability from the word ending at frame f to each of the words which the follow-set vector gives a non-zero probability of starting at frame f. If a word is included in the follow-set of more than one backtrace, we pick the maximum probability (Viterbi) backtrace.

If the parser fails at any point in parsing a backtrace, it backs off to the bigram grammar to compute word-transition probabilities for the remainder of the sentence. As Figure 9 shows, this backoff is quite rare, only happening for a very small number of the sentences (mostly the very long sentences). Thus even sentences whose correct transcription falls outside the CFG are usually forced into the nearest CFG-grammatical string.

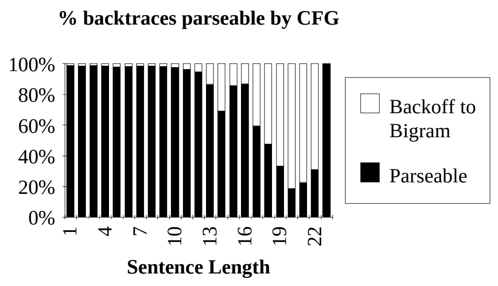


Figure 9: Percentage of backtraces covered by the SCFG

One of the most challenging design aspects of this algorithm was achieving real-time performance, since the Y_0 recognizer requires word-transition probabilities after every 10 ms frame, requiring on average 2400 calls to the parser per sentence. Despite this large number, our prototype tightly-coupled recognizer runs just 36% slower than our non-tightly-coupled recognizer using bigram probabilities. In order to make the coupling this fast, we optimized the algorithm extensively by using efficient indexing in the grammar and the chart, making use of shared substring information for the prefix computation, and adding a cache between the recognizer and the parser to avoid reparsing repeated backtraces.

The final way to use SCFG information relies on the intuition that the SCFG and the bigram offer complimentary sources of knowledge about grammar. Where the SCFG is best at modeling long-distance dependencies and hierarchical structure, the bigram is best at local and lexical dependencies. Our idea is to mix the two models on a frame-by-frame basis. We have experimented with two versions of this mixing. In one, we weight the models equally:

$$P(w_i|\mathit{prefix}) = 0.5P(w_i|\mathit{prefix,SCFG}) + 0.5P(w_i|\mathit{prefix,Bigram}) \quad (6)$$

In the second, we weight each model by how likely it is given the prefix (which we compute using Bayes' rule); this reflects the intuition that we should rely more on the model which demonstrates a better fit with previous input:

$$P(w_i|\mathit{prefix}) = P(\mathit{SCFG}|\mathit{prefix})P(w_i|\mathit{prefix,SCFG}) + P(\mathit{Bigram}|\mathit{prefix})P(w_i|\mathit{prefix,Bigram}) \quad (7)$$

Table 6 presents our word error results.

	Word Error
Bigram	33.7
SCFG-Smoothed Bigram	29.6
SCFG	29.6
SCFG/Bigram Weighted Mixture	29.5
SCFG/Bigram Equal Mixture	28.8

Table 6: BeRP Tight Coupling Performance

Note that the SCFG gave a significant 4.1% improvement in word error over the bigram. The SCFG and the SCFG-smoothed bigram performed equally, and the mixture models were slightly but not significantly better than either SCFG model. One conclusion we can reach is that compiling the SCFG into a bigram preserved most of the useful information. Additionally, the equal-mixture model seemed to do the best, although the difference with the other mixture and SCFG models was not significant – we plan to rerun these experiments on a larger test set. We suspect that the relative success of the equal-mixture over the weighted-mixture model was due to a useful side effect of equal-mixtures which penalizes the bigram model by normalizing it to 0.5 just in those backtraces where the SCFG returns a zero probability. [7]

[7] These results are on a slightly different system than that used for the

5 Results and Conclusions

The BeRP system has provided a useful and productive framework with which to implement and test a number of our research ideas, and demonstrates a successful combination of quite disparate elements in a working system. In addition, it has proved quite usable in its function as a real-time database frontend, with a word error rate of 32.1% and a semantic sentence error rate of 34.1%.[8]

The table below shows the system's overall sentence semantic error rate. This is measured by comparing with a hand-designed correct query component for each sentence. This correct query is compared to the query produced by the entire system and also to the query produced by just the backend operating without the recognizer on hand-transcribed sentences. Finally we show the results of comparing the recognizer output with the output produced by the natural language on perfect strings, to get an idea for the semantic performance of the recognizer.

	Semantic Error Rate
BeRP system	34.1
Backend alone	18.1
Recognizer alone	27.7

Table 7: BeRP semantic performance

Further details of the BeRP system are presented in Wooters (1993) and Jurafsky *et al.* (1994).

Acknowledgments

We'd like to thank Jerry Feldman for his early encouragement and support of this research, and Eric Fosler, Steve Omohundro, and an anonymous reviewer for extremely useful comments on this paper and the BeRP project.

This work was partially funded by ICSI and an SRI subcontract from ARPA contract MDA904-90-C-5253. Partial funding for the BeRP system development also came from ESPRIT project 6487 (The Wernicke project).

multiple-pronunciation and semantic performance tests. We have not yet computed semantic scores for the tight-coupling system.

[8]These numbers do not include our recent tight-coupling augmentations.

References

BAKER, J. K. 1979. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, ed. by D. H. Klatt & J. J. Wolf, 547–550.

BOURLARD, H., & N. MORGAN. 1991. Merging multilayer perceptrons & Hidden Markov Models: Some experiments in continuous speech recognition. In *Artificial Neural Networks: Advances and Applications*, ed. by E. Gelenbe. North Holland Press.

COHEN, M., H. FRANCO, N. MORGAN, D. RUMELHART, & V. ABRASH. 1992. Hybrid neural network/Hidden Markov Model continuous speech recognition. In *Proceedings Int'l Conference on Spoken Language Processing*, 915—918, Banff, Alberta, Canada.

COHEN, M. H., 1989. *Phonological Structures for Speech Recognition*. University of California, Berkeley dissertation.

FUJISAKI, T., F. JELINEK, J. COCKE, E. BLACK, & T. NISHINO. 1991. A probabilistic parsing method for sentence disambiguation. In *Current Issues in Parsing Technology*, ed. by Masaru Tomita, chapter 10, 139–152. Boston: Kluwer Academic.

GODDEAU, DAVID. 1992. Using probabilistic shift-reduce parsing in speech recognition systems. In *Proceedings Int'l Conference on Spoken Language Processing*, I.321—324, Banff, Alberta, Canada.

GOODINE, DAVID, STEPHANIE SENEFF, LYNETTE HIRSCHMAN, & MICHAEL PHILLIPS. 1991. Full integration of speech and language understanding in the MIT spoken language system. In *Proceedings of Eurospeech 91*, 24—26, Genova, Italy.

HERMANSKY, H., N. MORGAN, A. BAYYA, & P. KOHN. 1992. RASTA-PLP speech analysis technique. In *Proceedings Int'l Conference on Acoustics Speech and Signal Processing*, I–121–124, San Francisco, California. IEEE.

JELINEK, FREDERICK, & JOHN D. LAFFERTY. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics* 17.315–323.

JURAFSKY, DANIEL, CHUCK WOOTERS, GARY TAJCHMAN, JONATHAN SEGAL, ANDREAS STOLCKE, & NELSON MORGAN. 1994. The Berkeley restaurant project. In *Proceedings Int'l Conference on Spoken Language Processing*, Yokohama, Japan. to appear.

KAI, ATSUHIKO, & SEIICHI NAKAGAWA. 1992. A frame-synchronous continuous speech recognition algorithm using a top-down parsing of context-free grammar. In *Proceedings Int'l Conference on Spoken Language Processing*, I.257—260, Banff, Alberta, Canada.

KITA, KENJI, & WAYNE H. WARD. 1991. Incorporating LR parsing into SPHINX. In *Proceedings IEEE Int'l Conference on Acoustics, Speech, & Signal Processing*, I.269–272.

MOORE, ROBERT, FERNANDO PEREIRA, & HY MURVEIT. 1989. Integrating speech and natural-language processing. In *Proceedings DARPA Speech and Natural Language Workshop*, 243—247.

MURVEIT, HY, JOHN BUTZBERGER, & MITCH WEINTRAUB. 1991. Speech recognition in SRI's Resource Management and ATIS systems. In *Proceedings DARPA Speech and Natural Language Workshop*, 94—100.

PHILLIPS, MICHAEL, JAMES GLASS, & VICTOR ZUE. 1991. Modelling context dependency in acoustic-phonetic and lexical representations. In *Proceedings DARPA Speech and Natural Language Workshop*, 71—76.

RENALS, S., N. MORGAN, H. BOURLARD, M. COHEN, H. FRANCO, C. WOOTERS, & P. KOHN. 1991. Connectionist speech recognition: Status and prospects. Technical Report TR-91-070, International Computer Science Institute, Berkeley, CA.

RILEY, M. D. 1991. A statistical model for generating pronunciation networks. In *Proceedings IEEE Int'l Conference on Acoustics, Speech, & Signal Processing*, 737–740.

SENEFF, STEPHANIE, HELEN MENG, & VICTOR ZUE. 1992. Language modelling for recognition and understanding using layered bigrams. In *Proceedings Int'l Conference on Spoken Language Processing*, I.317—320, Banff, Alberta, Canada.

STOLCKE, ANDREAS. 1993. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. Technical Report TR-93-065, International Computer Science Institute, Berkeley, CA. To appear in *Computational Linguistics*.

——, & STEPHEN OMOHUNDRO. 1994. Best-first model merging for hidden Markov model induction. Technical Report TR-94-003, International Computer Science Institute, Berkeley, CA.

——, & JONATHAN SEGAL. 1994. Precise n-gram probabilities from stochastic context-free grammars. In *Proceedings of the 32nd ACL*, Las Cruces, NM. To appear.

WITHGOTT, M. M., & F. R. CHEN. 1993. *Computation Models of American Speech*. Center for the Study of Language and Information.

WOOTERS, CHARLES C., 1993. *Lexical Modeling in a Speaker Independent Speech Understanding System*. Berkeley, CA: University of California dissertation.

ZUE, VICTOR, JAMES GLASS, DAVID GOODINE, HONG LEUNG, MICHAEL PHILLIPS, JOSEPH POLIFRONI, & STEPHANIE SENEFF. 1991. Integration of speech recognition and natural language processing in the MIT VOYAGER system. In *Proceedings IEEE Int'l Conference on Acoustics, Speech, & Signal Processing*, I.713–716.