

# Semi-Automatic Learning of Acoustic Models

---

**A White Paper on Proposed Research Projects**

**This document contains intellectual property developed at private expense by the author. All rights are reserved. Use of this document is restricted to colleagues of the author for the purpose of furthering the development of the inventions and ideas contained herein.**

James K. Baker

Human Language Technology Center of Excellence  
The Johns Hopkins University

Center for Innovations in Speech and Language  
Language Technology Institute  
Carnegie Mellon University

Draft date: May 18, 2009  
12:01:34 PM

## Table of Contents

<b>1.0 INTRODUCTION</b> .....	<b>3</b>
<b>2.0 SAMPLE PROJECT PROPOSALS</b> .....	<b>5</b>
SAMPLE TASK 2.1: IMPROVING PRICE-PERFORMANCE OF A FAST RECOGNIZER .....	5
<i>Architecture 2.1.1: Fast match short-term look-ahead</i> .....	6
<i>Architecture 2.1.2: Long-term look ahead</i> .....	7
<i>Architecture 2.1.3: Bottom-up error correction</i> .....	10
SAMPLE TASK 2.2: BOOTSTRAPPING ACOUSTIC MODELS FOR A NEW LANGUAGE .....	12
SAMPLE TASK 2.3: REFINING A PRONUNCIATION LEXICON.....	12
SAMPLE TASK 2.4: DEVELOPING A MODULAR, MULTI-KNOWLEDGE SOURCE SYSTEM .....	13
<i>Project 2.4.1 Data-dependent Fusion of Multiple Classifiers</i> .....	13
<i>Project 2.4.2 Modeling Meta-Knowledge and Socratic Controllers</i> .....	14
<i>Project 2.4.3 Training for Diversity</i> .....	17
<b>3.0 STATISTICAL VALIDATION</b> .....	<b>20</b>
3.1 LIKELIHOOD RATIO BASED STATISTICAL VALIDATION.....	22
<b>4.0 LEARNING-BY-IMITATION</b> .....	<b>24</b>
<b>5.0 THE NEED FOR PROOF-OF-CONCEPT EXPERIMENTS</b> .....	<b>28</b>
<b>6.0 HYBRID-UNIT, SCALABLE-VOCABULARY RECOGNITION</b> .....	<b>30</b>
SAMPLE EXPERIMENT 6.1: BUILDING A PHONETIC RECOGNIZER FOR A NEW LANGUAGE FROM SCRATCH. .....	33
<i>Objectives of experiment 6.1</i> .....	35
SAMPLE EXPERIMENT 6.2: A FAST HYBRID RECOGNIZER .....	35
<i>Objectives of experiment 6.2</i> .....	38
<b>7.0 MULTI-LEVEL STATISTICAL VALIDATION</b> .....	<b>39</b>
<b>8.0 SYLLABLE-BASED RECOGNITION</b> .....	<b>41</b>
SAMPLE EXPERIMENT 8.1: AUTOMATICALLY LEARNING A SYLLABLE-BASED LEXICON .....	41
<i>Step 1: Learning a syllable grammar</i> .....	42
<i>Step 2: Determining which Acoustic Syllables Correspond to which Characters</i> .....	49
<i>Objectives of experiment 8.1</i> .....	50
<b>9.0 DEVELOPING A PRONUNCIATION LEXICON FROM SCRATCH</b> .....	<b>51</b>
GRAPHHEME-TO-SOUND RULES: INSERTION AND DELETION .....	51
SAMPLE EXPERIMENT 9.1: LEARNING MULTIPLE-LETTER PHONEMES .....	53
ANNEALING AND STATISTICAL VALIDATION OF STRUCTURAL LEARNING.....	54
<b>APPENDIX A: TRAINING ACOUSTIC MODELS</b> .....	<b>56</b>
<b>NEW TECHNIQUES AND CONCEPTS INTRODUCED IN THIS PAPER</b> .....	<b>59</b>
<b>REFERENCES</b> .....	<b>61</b>

## 1.0 Introduction

This paper is an intuitive introduction to some newly proposed techniques for semi-automatic learning. Some of these proposed techniques are specifically aimed at greater automation of the process of developing acoustic models for speech recognition in a new language. Such automation is needed in particular to support development of speech recognition in a large number of languages, such as the All the Languages of the World (ATLOTW) Grand Challenge, whose goal is to develop speech recognition, machine-aided translation and other speech and language technologies in all of the world's languages. The emphasis in this white paper is on techniques relying primarily on learning from unlabeled data, requiring only small amounts of labeled data. Developing techniques for such a data-rich, label-poor situations has been a primary focus for the Human Language Technology Center of Excellence (HLTCOE) at Johns Hopkins University.

On the other hand, many of the techniques in this paper assume that a large amount of unlabeled data is available. For languages in which even unlabeled data is not plentiful, additional techniques will be needed. Developing technology for such languages is an important goal for future research, but is beyond the scope of this white paper.

The techniques and experiments proposed in this paper are in support of all of the long-term research at the HLTCOE at Johns Hopkins University, and of the ATLOTW program. The ATLOTW Grand Challenge has been proposed as a long-term, multi-institution, international collaboration being organized by the Center for Innovations in Speech and Language (CISL) of the Language Technology Institute (LTI) at Carnegie Mellon University. The examples in this white paper will primarily be from speech recognition, but most of the techniques are designed to apply to pattern recognition in any field.

The phrase “Semi-automatic learning” is intended to be a more general term than “semi-supervised learning” that includes semi-supervised learning as well as new techniques that are similar to semi-supervised learning but which don't fit into the standard definition. The proposed techniques also go beyond the original concept of *delayed-decision training* and even use techniques outside the standard definition of *semi-supervised learning*, so new names are proposed for these techniques. In general, the proposed techniques are characterized by the situation in which there is a large amount of data, but only a small proportion of the data has been manually labeled. However, the proposed techniques are further characterized by the utilization of extra knowledge that is outside the conventional semi-supervised learning framework.

The proposed techniques include the following:

- 1) *Cooperatively-supervised self-training*. Applies to maximum-likelihood training of parametric generative models. It is similar to conventional self-training except additional knowledge is used in the automatic labeling of the unlabeled data. The use of a language model in the self-training of acoustic models is an instance of

- this methodology. Self-training will not be a primary focus for this document, but versions of cooperative self-training other than using a language model may be included in some of the experiments. Self-training of hidden Markov models is an instance of the same basic algorithm as used for supervised training of hidden Markov models, the EM algorithm, which is presented in Appendix A.
- 2) *Statistical validation*. This technique is a method of validation rather than directly being a learning algorithm. Moreover, it can assist in structural learning as well as in the training of parametric models. It can be done without any manually labeled data, but can replace supervised development testing. Several variations of statistical validation will be discussed.
  - 3) *Learning-by-imitation*. Applies to the diagnostic paradigm as well as the generative paradigm of semi-supervised learning and to discriminative training as well as to maximum likelihood estimation. Like statistical validation, it involves the comparison of two systems. In this case, one system is designed to be lower performance than the other. Learning-by-imitation can run supervised learning algorithms on data that has not been manually labeled.

In section 2, a set of sample projects will be sketched. These projects make extensive use of statistical validation and learning-by-imitation. If you are not yet familiar with these techniques, you may want to read sections 3 and 4 first, and then return to section 2. The projects in section 2 are intended to be examples of potential projects that will utilize the proposed new techniques in semi-automatic learning. These sketches are given to show that there are many different ways in which the new techniques might be used. Obviously this total collection of projects is too much to do in a short time frame. However, achieving the goals of the ATLOTW Grand Challenge is expected to take decades of work by many research groups. There will be time to explore all of these techniques and many more.

This document also includes a separate essay each on statistical validation and learning-by-imitation. Then, several example experiments will be described and related to these and other techniques of semi-automatic learning.

## 2.0 Sample Project Proposals

The objective for the projects discussed in this section will be to provide proof-of-concept for the proposed new methodologies in semi-automatic learning. The research projects will be designed to optimize the efficiency of determining whether the new techniques work as expected. When a small-scale experiment is adequate to determine whether or not a technique works, it will be preferred over a large-scale experiment. This section makes frequent reference to the proposed new methodologies, in particular to statistical validation and learning-by-imitation. If unfamiliar with these methodologies, the reader may choose to read sections 3.0 and 4.0, which explain these methodologies, before returning to this section.

Since the focus is on semi-automatic learning methodologies, an example pattern recognition or classification task must be chosen. This example task should not be chosen merely for its own sake as a potentially useful task, but rather the task should be chosen based on the ability of experiments with the task to measure how well the proposed learning methodologies work.

In this section, only a sketch will be given for each project, with some indication of which semi-automatic learning methodologies might be relevant to each project. More in-depth discussions of a few sample experiments will be given in later sections to illustrate in more detail how the semi-automatic learning techniques may be applied in particular situations. Also in this section, some tasks will be presented with several suggestions of procedures that might be applied to the task.

On the other hand, in the research in semi-automatic learning discussed in this paper, a proof-of-concept of a general semi-automatic learning method may be more important than the particular task to which it is applied. This alternate perspective is illustrated by having multiple sample tasks to which the methods presented in this paper may be applied. Both points of view are valid. This paper makes no attempt to rank the sample tasks and sample experiments as to which ones should be done first. For both HLT/COE and CISL, the research focus is on fundamental core technology developed over a frame of time in support of long-term programs such as the ATLOTW Grand Challenge.

### Sample Task 2.1: Improving Price-Performance of a Fast Recognizer

In the context of this task, a “fast” recognizer is one that has not be optimized for recognition accuracy, but rather that has been designed for some price-performance objective in which a significant trade-off has been made of accuracy for reduced computation. Given a baseline example of a fast recognizer, three different architectures are discussed for implementing a family of fast recognizers. Associated methods are discussed for finding, within each architecture, the recognizer that optimizes a specified price-performance criterion. One research objective is to compare the performance of these methods when trained using a large amount of manually labeled data versus their

performance when using data that has not been manually labeled. Several methods will be explored for using the unlabeled data.

### Architecture 2.1.1: Fast match short-term look-ahead

In architecture 2.1.1, the fast recognizer includes a “fast match” computation. If the baseline fast recognizer does not include a fast match, one is added to it. A “fast match” is a computation that selects a subset of the total vocabulary so that the rest of the computation that is done after the fast match is done only on the selected subset. For a large vocabulary, the fast match may select a subset that is only 1/100 or 1/1000 of the total vocabulary. Typically, the fast match computation is done only once per phoneme (or perhaps only once per syllable). The acoustic match part of the fast match is typically done independent of the context. Hence the fast match typically only is done once for a particular time in the speech stream even though there may be multiple active hypotheses as to the word sequence leading up to that point in the sentence.

Research systems optimized for the highest possible performance often do not use a fast match. However, fast match has been used in commercial speech recognition systems for over twenty years. For example, here is the abstract for a key patent filed in 1985 that has been referenced by at least 143 other patents [Ba85, US Patent No. 4783803]:

#### **Abstract (of Patent 4783803)**

A system is disclosed for recognizing a pattern in a collection of data given a context of one or more other patterns previously identified. Preferably the system is a speech recognition system, the patterns are words and the collection of data is a sequence of acoustic frames. During the processing of each of a plurality of frames, for each word in an active vocabulary, the system updates a likelihood score representing a probability of a match between the word and the frame, combines a language model score based on one or more previously recognized words with that likelihood score, and prunes the word from the active vocabulary if the combined score is below a threshold. A rapid match is made between the frames and each word of an initial vocabulary to determine which words should originally be placed in the active vocabulary. Preferably the system enables an operator to confirm the system's best guess as to the spoken word merely by speaking another word, to indicate that...

This patent is only one example. There are several patents from the same era showing different methods for doing a fast match. Normally the scores computed by the fast match, if any, are used only internally in the fast match. That is, once the vocabulary subset has been selected new scores are computed using standard scoring procedures and fast match scores are replaced and then ignored. Similarly, other than whether or not a particular word is included in the selected subset, any rank order computed among the words within the fast match is also only used internally. Therefore, the performance objective of the fast match is entirely determined by that subset selection and does not depend further on the scores. This point means that standard training objective functions,

such as maximum likelihood estimation or even standard discriminative training criteria such as MMI, are not the appropriate objective for an optimized fast match.

In architecture 2.1.1, as in most implementations of fast match, there is no mechanism, except lucky chance, by which the fast match can correct an error in which the full match computation gives a better score to an incorrect answer than to the correct answer. Therefore, in any objective function for optimizing the fast match, the correct answer could just as well be replaced by the answer that receives the best score in the standard match computation.

Thus, optimizing the fast match is an ideal situation for learning-by-imitation (see section 4.0), in which the *reference* system is the standard match and the fast match is the *limited* system.

The performance part of the objective function is based on whether or not the best scoring word is in the selected subset. The best scoring word is determined by running standard recognition by the reference system and does not depend on any manual labeling or verification. The price part of the objective function is the combination of the time it takes to perform the fast match computation itself and the time it takes to perform the standard match computation on the restricted subset of the vocabulary. Both of these figures may be measured or estimated independently of knowledge of the correct answer. Hence the price-performance objective function for the fast match may be computed by automatic procedures without requiring any manual labeling.

In development of a speech recognition system, especially in experiments to improve price-performance, normally some of the (manually labeled) training data is set aside for development testing. Using the standard match as the set-aside knowledge source, unsupervised statistical validation (see section 3) may be used in place of conventional development testing. It may also be used for automated structural learning of new fast match structures.

Therefore, this experiment to optimize a fast match computation may be used as proof-of-concept experiments for both learning-by-imitation and statistical validation. The technique requires an existing recognition system as a reference. Presumably the reference system used some manually labeled training data somewhere in its development. Other than that, the fast match training and optimization in architecture 2.1.1 requires no manually labeled data.

### **Architecture 2.1.2: Long-term look ahead**

The fast match computation of architecture 2.1.1 performs a kind of short-term look-ahead to determine which word candidates are most likely given the upcoming acoustic observations. In a language model with moderate to high relative redundancy, long-term look-ahead can provide a substantial amount of pruning of hypotheses that are inconsistent with likely future word sequences.

High relative redundancy occurs naturally in applications such as command-and-control or spoken data entry or data retrieval. It also occurs with stereotypical phrases or stereotypical dialogs, such as the initial dialogue when two people meet and greet each other. Stereotypical dialog during greetings is particularly common among Arabic speakers and many Asian languages. It is also common in all languages in formal diplomatic situations. High relative redundancy also occurs in other situations, such as when the speaker is known to be speaking a translation or a paraphrase of a known passage. High relative redundancy also occurs when the speaker is telling an oft-repeated story or giving a standard sales pitch.

### *Method 2.1.2.1: Dual search*

One way to take advantage of long-term look ahead is to perform a dual search. This method is particularly appropriate when operating in streaming mode in which there is a requirement for real-time throughput with a limited delay allowed for the recognition process.

In the dual search process two decoders are used, either of which may be either a stack decoder, a multi-stack decoder or a frame-synchronous beam search decoder. The first decoder uses only a specified fraction of the available computation but has a tight pruning threshold and is otherwise tuned to keep up with the real-time audio. The first decoder may also use a weaker language model than usual or even explicitly allow the system to jump from one language model state to another with only a generic penalty. This mechanism may allow the first decoder to get back on track in later words after making a pruning error on a particular word.

The second decoder uses the same acoustic models and left-to-right language modeling as a baseline single decoder system that is allowed more computation time. However, in its hypothesis pruning decision (and its stack decoder extension decisions) it uses a look-ahead score in addition to the scores used by the baseline single decoder system. This look-ahead score is an estimate for each active hypothesis of the score that the particular active hypothesis will make on the remaining observations.

Note that this long-term look-ahead has completely different properties than the short-term look-ahead done in the fast match. In the fast match, the acoustic part of the look-ahead score is the same for all active hypotheses. The short-term look-ahead is used to select among possible extensions for each hypothesis, not to compare the hypotheses. In contrast, the long-term look-ahead is used only to compare the active hypotheses, directly pruning some of them or giving them different pruning thresholds based on their respective estimated future scores.

In the dual search, the performance does depend on whether the first decoder finds the correct answer, not just on whether it finds the best scoring answer. Furthermore, dual search only has an advantage if it is impossible to do a search with a low pruning error rate in the available amount of computation time. Therefore, the first decoder is expected to often fail to find the best scoring answer, much less the correct answer. However, in

this situation the second decoder, or a separate baseline recognizer, may still be used as the *reference* system for learning-by-imitation.

Therefore, optimization of the price-performance of the first decoder in the dual search provides another proof-of-concept for learning-by-imitation. This task is a more severe test of the concept of learning-by-imitation. In this situation it is not expected that learning-by-imitation will perform as well as supervised learning on the same amount of data. Its potential advantage, as with conventional methods of semi-supervised learning, is that it may be able to improve performance by utilizing a large amount of unlabeled (that is, automatically labeled) data.

The baseline system may also be used as a set-aside knowledge source for statistical validation. Therefore, unsupervised statistical validation may be used in this task in place of supervised development testing.

### *Method 2.1.2.2 Bottom-up long-term look ahead*

Fast-match-like bottom-up computations may also be used for long-term look ahead. The idea is that bottom-up analysis will find places in the future speech stream at which it can determine that the correct word is likely to be among the words in a small subset of the vocabulary. One-word look-ahead can be done by a standard fast match. Restricting the vocabulary two or more words in the future will improve the efficiency of the pruning if the words detected bottom-up are more likely for some of the active language model states than for others. That is, the look-ahead is most likely to be helpful when the right-side, long-term language model has significant mutual information over the residual entropy from the left-side language model and acoustic observations. That is, where  $\Delta$  in equation 2.1 is significantly greater than 0, or a similar condition is true for a multi-word phrase from the look-ahead time.

$$2.1 \Delta = \sum_w p(W_t = w | Y_1, Y_2, \dots, Y_{t(t)}, W_{1+k}) \log(p(W_t = w | Y_1, Y_2, \dots, Y_{t(t)}, W_{1+k})) \\ - \sum_w p(W_t = w | Y_1, Y_2, \dots, Y_{t(t)}) \log(p(W_t = w | Y_1, Y_2, \dots, Y_{t(t)}))$$

The bottom-up long-term look-ahead can use the same acoustic features as the fast match, but the objective function and the training will be different. For one thing, only in favorable situations does the long-term look-ahead provide enough information to be worthwhile. That is, there must be a word or phrase in the near future that can be easily detected bottom-up from the acoustics and that also has the property that there is high mutual information. When the bottom-up analysis can quickly determine that these conditions are not satisfied, then the analysis can be aborted and the look-ahead can be skipped until there is a more favorable opportunity. Unlike the fast match, the long-term look-ahead is not required to run all the time. It only needs to run when it will be useful.

Thus, for example, the bottom-up analysis for the long-term look-ahead doesn't need to work with the entire vocabulary. A set of words may be chosen that all have a large amount of acoustic entropy (usually because the words are long) and a large amount of

semantic content (so that the words will have significant mutual information with the language model across a gap of one or more words). The bottom-up detection could be limited to this set of special words.

In the bottom-up analysis for long-term look-ahead it would be useful if the correct word could be successfully detected even if it would not be recognized as part of the best scoring hypothesis, so again this method is a more demanding test of the concept of learning-by-imitation. As before, the baseline recognition system may be used as a reference system both for learning-by-imitation and for statistical validation.

Note that the subsystem for the bottom-up detection of a set of special words can be used directly as a fast spoken term detection system. Optimizing the performance of such a stand-alone system would be useful even if the long-term look-ahead is not used in the standard system.

### **Architecture 2.1.3: Bottom-up error correction**

A fast recognizer sometimes fails to even find the best scoring word sequence. First, it is necessary to detect that such a situation might have happened. Then, something must be done to fix the problem.

#### ***Project 2.1.3.1 Error detection***

Detection of errors is just confidence estimation from a different perspective. However, errors caused by pruning or other failures in the decoder search have different characteristics from the usual errors. Unless there is an anomaly in the signal or a defect in the model, the correct word will match the acoustic observations at least moderately well. If the best scoring word is found during the search, then there will be an error only if some other word sequence matches better than the correct sequence. In a high-performance system, there are very few pruning errors or other failures in the decoder search, so the best scoring sequence is almost always found.

A different situation occurs if the correct word is out of vocabulary or if the correct word was pruned, as is expected to occur more frequently with a fast recognizer. Then the best scoring sequence that is found might not match the acoustics as well as a correct word usually does. It is possible to detect such situations from the fact that the best sequence found by the decoder has only a mediocre match to the acoustic segment.

In a high-performance system the strongest indication that error is more likely than usual is that one or more other words score almost as well as the best scoring word. In the extreme case, if several words score identically, the system is saying that it is making a random choice and only has one chance in  $n$  of being right. Usually all of these alternatives score well.

In a fast recognition system, the strongest indication that the correct word doesn't even appear in the results lattice is that some recognition that is not constrained to the words in

the lattice finds an acoustic match that is much better than the one found by the base recognizer. The less constrained recognition could be a phoneme recognizer, a syllable recognizer or a word recognizer with a less constrained language model [Wh08].

Many other features have been used as features for confidence estimation [Wh08]. Except for the top two features that have just been discussed, most of the other features normally used for confidence estimation will be useful both for detection of typical errors in a high-performance system and for detection of decoder errors in a fast recognizer (and out-of-vocabulary errors in a high-performance system).

Because the principle feature is different, however, the training of the confidence measure or of the error detector is different for detection of decoder and pruning errors. In particular, the same system with a broader search may be used as a reference system. Thus the error detector, including all its features, can be trained by learning-by-imitation, without requiring any data to be manually labeled.

### *Project 2.1.3.2 Hypothesizing missing words*

Assume the situation is that the error detector has found a region of speech in which it is likely that the best scoring word was not found. The task is to hypothesize additional words to add to the lattice so that a better scoring word sequence might be found.

This is the third instance of bottom-up word detection. It has different features from either fast-match or bottom-up long-term look-ahead. This detection is done in the context of a complete results lattice that may have decoder errors because it is the result from a fast recognizer that has sacrificed some performance for speed. In this situation it is likely that some of the nearby correct words are already in the lattice. In fact, every subsequence of words that are missing from the lattice is bounded both from the left and the right either by an end-of-utterance or by a correct word that is included in the lattice. Therefore, working in from the edges of the missing subsequence, each word has an immediately adjacent word that is in the lattice.

Therefore, by matching separately in the context of all choices in the lattice that are adjacent to the region in which a putative error situation has been detected, the correct context may be assumed. In particular, it may be assumed that the word boundary time for the adjacent word has been determined. Also there is at least a forward or backward language model and correct across-word-boundary acoustic context. In addition, the lattice includes (incorrect) words in the suspected error region. Although incorrect, these words may have syllables, phonemes and other features in common with the missing correct words. Also, in a hybrid recognizer there may be the results from direct syllable and phoneme recognition.

The task is to hypothesize words that are missing from the lattice. Because of the interaction of acoustic modeling scores and language model scores with the pruning in a fast recognizer, it might be difficult to characterize *a priori* which words are most likely to be missing from the lattice. Will it be words that are so rare that they received very

poor scores from the language model? Or will it be short, unstressed words that may have very indistinct acoustics?

A standard fast match computation could be used to hypothesize missing words. However, a standard fast match is optimized for a different objective. Hypothesizing words in the situation of suspected errors is more challenging than for standard fast match and the computational impact of excess hypotheses may be greater. Therefore, a word hypothesizer should be custom trained for this situation. This word hypothesizer can use all the features of a fast match, but can also use the extra features available in this situation.

Because the task is to detect better scoring missing words whether or not they are the correct word, the baseline high-performance recognizer may be used as the reference system for learning-by-imitation, even though the objective function is different from the objective for a fast match.

### **Sample Task 2.2: Bootstrapping Acoustic Models for a New Language**

Section 6, in particular Sample Experiment 6.1, discusses a method for bootstrapping a phonetic recognizer in a new language without any transcribed data in that language, much less any manually labeled phonetic data. The method begins with a phonemic or phonetic recognizer in some other language. It uses a collection of speech data that has not been manually transcribed. The speech data is from a mixture of both languages. The method uses learning-by-imitation and statistical validation to develop an iteratively improved sequence of phonetic recognizers for the two languages.

Section 8 discusses a method for bootstrapping a syllable-character-based pronunciation lexicon for a language, such as the spoken dialects of Chinese, in which each character usually corresponds to one syllable. Sample Experiment 8.1 uses hypothesis-independent likelihood-based statistical validation, which doesn't require a reference system.

### **Sample Task 2.3: Refining a Pronunciation Lexicon**

Even the best state-of-the-art speech recognition systems have errors and omissions in their pronunciation lexicons. Omissions are inevitable because new words are continuously being invented, because there is an almost unbounded number of proper names, and because any naturally occurring speech corpus will also include foreign words. Errors are inevitable because current systems do not support modeling the full variability of dialects much less idiosyncratic pronunciations of individuals.

A desirable task then would be to detect and fix errors and omissions in a pronunciation lexicon, given a limited amount of manually labeled data and a large amount of speech data that has not been manually labeled. Experiments [Wh09] have already shown that unsupervised statistical validation can match the performance of conventional supervised methods for deciding which of two proposed pronunciations is better.

Similarly, statistical validation can be used to accept or reject individual letter-to-sound rules. Improved letter-to-sound rules can be used to generate candidate pronunciations for the comparison method mentioned in the previous paragraph.

Statistical validation may also be applied at a very fine grain, testing whether any pronunciation in the dictionary is correct for a particular instance of a particular word. With enough data, this method can be used to automatically develop dialect variation data. Given dialect variation data, rules can be developed for transforming a pronunciation lexicon across dialects. Statistical validation may be used to test individual rules within the transformation system.

## **Sample Task 2.4: Developing a Modular, Multi-knowledge Source System**

Sample Task 2.1 focused on improving the price-performance of a system that trades some performance for a reduction in the amount computation. This task seeks to improve performance at the cost of possibly a great increase in the amount of computation. One method that has used to get a modest improvement in performance is to use multiple classifiers on a shared classification task.

Almost all carefully done experiments using multiple speech recognition systems have shown better performance for the fused system than for any one individual system. However, in these experiments the interaction among the systems has usually been very limited. In particular, the systems have generally either been trained separately or have merely been cross-trained. Furthermore, because the individual system training has generally been supervised training, the cross-training has normally been limited to the test data. In other words, the systems are mostly trained independently from each other. There is no integrated training process in which the systems are trained to jointly optimize the performance of the fused system.

### **Project 2.4.1 Data-dependent Fusion of Multiple Classifiers**

One experiment within this task would be just to do such joint training using standard supervised training techniques. Note that if the hidden Markov processes run independently and the systems do not share acoustic features, then joint maximum-likelihood training would be equivalent to maximum-likelihood training done separately on the component systems. Some improvement over most current methods comes just from training a fusion function in which the weights or the parameters of a more general combining function are trained to vary in a data-dependent fashion. Mixtures of experts methods ([Ja91], [Jo94], [Wa96]) demonstrate improved performance when particular experts are assigned responsibility in particular local regions.

Furthermore, corrective training could directly do hill climbing based on the performance of the fused system. This hill-climbing could optimize the parameters in the models in the individual systems as well as the parameters in the fusion function.

However, this hill-climbing is likely to be very slow to converge to the jointly optimized parameter values. In particular, it will be difficult for hill-climbing based on local

evaluation of the performance to discover the potential value of decreasing the separate performance of individual systems in order to increase their diversity in a way that allows improved joint performance. Therefore, even for this supervised joint training it may be necessary to use the techniques discussed under projects 2.4.2 and 2.4.3.

Further performance improvement might be possible if additional recognition systems are created as variants of the existing systems. Because each system has many design decision points, exponentially many variant systems could be created. A danger in this approach, however, is that the large number of model parameters could cause over fitting of the training data and thus degrade performance. Careful smoothing could avoid the degradation in performance, but the performance would still be restricted by the limited amount of manually labeled training data.

Therefore, a second experiment in this task attempts to get further improvement in performance by adding a large amount of speech data that has not been manually labeled. Semi-supervised learning, in particular self-training, may be done using the large amount of unlabeled data. However, the joint training proposed above does corrective-training hill climbing rather than maximum-likelihood estimation. It is not clear that self-training will work in this situation because errors in the automatic labels could lead to self-fulfilling prophecies in the form of local maxima.

Rather than pure self-training, the fused system could be used as a reference system as a basis for learning-by-imitation for each individual system. Then it could be used for self-training just of the parameters in the fusion function. More novel techniques for developing and training an integrated multiple classifier system will be discussed in the following projects.

However, a greater change in paradigm is needed to address the problem that a locally evaluated objective function cannot easily represent the delayed benefits of diversity or selective assignment of responsibility to particular experts. A local objective function will mainly represent the more direct effect that in certain regions some experts are more reliable and should locally be given more weight by a data dependent fusion function. However, the benefit of controlling the training so that particular classifiers get trained to be specialized experts is delayed until these classifiers have received enough training to achieve a sufficient level of expertise. This delayed benefit means that, even if the benefit is technically represented in the joint objective function, the hill climbing is likely to be impractically slow. The concept of Socratic Controllers, combined with the semi-automatic learning methods, provides a way to avoid this difficulty.

### **Project 2.4.2 Modeling Meta-Knowledge and Socratic Controllers**

It has already been mentioned that there is a potential advantage to having the weights or other parameters in the fusion engine vary as a function of the data. Notice that estimating the optimum values of the fusion parameters as a function of the data is itself a non-linear regression problem (or a classification problem if there are discrete regions). To distinguish this regression or pattern analysis problem from the original pattern recognition problem, consider a separate entity assigned to do this second task.

This entity is a kind of meta-knowledge analyzer. It models knowledge about the component classifiers rather than knowledge about the original classes. It is called a “Socratic controller,” in honor of Socrates’ philosophy distinguishing knowledge about knowledge from the knowledge itself. In his defense speech at his trial, he made a statement that roughly translates as “the only thing that I know is that I don’t really know anything” (and that one piece of meta-knowledge is what made the Delphic Oracle declare Socrates as the wisest of the Greeks).

The Socratic controller solves a pattern recognition problem that is very different from the original problem. The difference can be illustrated by a simple example. Let the original classification problem be to distinguish two classes. Assume that we have two component classification systems that each do well in a different region of a high-dimensional space. For purposes of visualization, let this high-dimensional space be projected into two dimensions such that one of the component classifiers does well at discriminating the two classes when  $x < 0$  and that the other classifier does well at distinguishing the two classes when  $x > 0$ . Suppose further that the complementary subspaces that project onto the y-axis are such that the y-values associated with the two classes when  $x < 0$  have no relationship to the y-values associated with the two classes when  $x > 0$ .

This situation is illustrated by Figure 2.1. In this situation the Socratic controller is a classifier rather than a regression function. That is, it specifies one region in which to believe Classifier I and another region in which to believe Classifier II. In a less extreme case, it would have a local regression function that gives more weight to the better classifier. Notice that the two regions distinguished by the Socratic controller do not distinguish between classes A and B. The Socratic controller makes no attempt to directly recognize A or B. It learns the pattern of when Classifier I is more likely to be correct than Classifier II.

Figure 2.1

A Socratic controller has access to all the input data of any of its component classifiers. It also has access to the results of the analysis of each of the component classifiers. This may include scores, alternate hypotheses, and confidence measures. It may include a full results lattice as well as the single best answer. Abstractly, however, the Socratic controller is just a non-linear regression or a classifier. Therefore, the Socratic controller may itself be trained by all the available methods of supervised and semi-supervised learning.

A conventional multiple classifier system with a data-dependent fusion function, such as a mixture of experts system, can also solve the problem illustrated in Figure 2.1. The Socratic controller, by explicitly representing meta-knowledge separately from lower level data-knowledge is a more general mechanism. In fact, a Socratic controller can emulate any conventional multiple classifier system, including any mixture of experts. However, by itself that generality doesn't introduce anything new. The real power of the concept of Socratic controller comes from the use of new training and learning algorithms. In particular, a Socratic controller can use statistical validation to delay any local assignment-of-responsibility decision.

As the keeper of knowledge about the knowledge of its component subsystems, the Socratic controller should also control the teaching of its components. That is, as a software object, a Socratic controller should include all the methods of supervised

training and semi-automatic learning, not for itself but for the component classifiers that it controls. It should also include tools for the decision-making involved in the learning process. This decision-making process can also be formulated as a set of pattern recognition problems on metaknowledge.

In a multiple classifier system, the fused system is an obvious choice as a reference system for learning-by-imitation or statistical validation applied to the component systems. In a modular multiple knowledge source architecture, there are a large number of Socratic controllers. The overall system has many component subsystems, such as fast match, long-term look-ahead, acoustic modeling, language modeling, etc. In this architecture, each subsystem is itself a multiple classifier. The overall system can be used as a reference system not only for training the component subsystems but also their individual Socratic controllers.

In a simple multiple classifier system with only one subsystem task, a separate Socratic controller can be assigned to each small set of, say, three individual component classifiers. The total fused system provides a reference for training the component Socratic controllers.

### **Project 2.4.3 Training for Diversity**

It is well known that the improvement in performance achieved by a multiple classifier system over the performance of the component systems depends on the amount of diversity among the component systems. Many experiments have been done to develop methods for generating multiple classifiers in a way that tends to get some diversity among the component systems. Generally, however, these development methods do not directly train the component systems to be more diverse. That is, the parameters in the component systems are not adjusted to maximize some measure of diversity. Instead the systems are designed to get some diversity as a side effect of conventional model training in which each component observes different training data or different features or different transformations of the features.

The hill climbing proposed in project 2.4.1 does adjust the parameters in the component systems based on the performance of the fused system. To the extent that it is true that better fused performance will result from fused systems with more diversity, the diversity will be indirectly reflected in the objective function, that is, the joint performance. However, because the performance differences in any local regions will depend heavily on the performance of the individual systems there may be many saddle points in the objective function and the hill climbing process may be very slow to learn the performance improvement that can eventually be achieved by first teaching the component systems to be more diverse.

This project explores the concept of directly teaching the component systems to be diverse and then teaching them to optimize the joint performance when each component system learns in the context of a diverse collection of systems.

Fortunately, diversity can be directly measured without knowing the correct answer and therefore without manual labeling. There are various possible measures of diversity but in general they can be evaluated merely by knowing whether or not two or more components agree, without needing to know whether any of them are correct.

On the other hand, by itself maximizing diversity does not lead to improved joint performance. At least under some measures, diversity is maximized if two components always disagree. The fusion of two such systems has no more knowledge than either system alone.

Because it simplifies the computation of joint probabilities, it might seem that the ideal amount of diversity is for the component systems to be pair-wise statistically independent. High-performance classification systems, however, will be correlated in the sense that each almost always gets the correct answer and therefore the classifiers almost always agree. Therefore a better goal is *conditional* independence of the errors, or at least that the errors are conditionally uncorrelated. However, for the errors to be conditionally uncorrelated is not the ideal for improved fusion performance. Although it is useless to have the component systems be absolutely negatively correlated, it is ideal to have them be negatively correlated conditional on the fact that at least one of them is in error. Then the two components never make the same mistake at the same time. If there are three systems that pair-wise have perfect negative correlation conditional on there being an error, then no two of the systems will make the same mistake at the same time. In a two class problem, in every case at least two of the systems will be correct and a simple majority vote will be correct 100% of the time.

Perfect conditional negative correlation of errors is of course impossible to achieve. However, a diversity objective function should prefer negative conditional correlation of errors to zero correlation.

Although negative conditional correlation of errors is desirable, by itself it is still not a good objective function. Negative conditional correlation of errors that is achieved by having all the component error rates be higher will not necessarily lead to improved performance of the fused system.

The objective function should have some measure of the performance of the component systems and/or of the fused system in addition to the measure of diversity. For example, the objective function could include, with some weighting factor, the sum of the component error rates or the maximum of the component error rates. Alternately, some measure of diversity could be maximized subject to keeping a measure of performance constant or bounded by some inequality. However, to avoid the hill climbing problems mentioned above, the measure of performance should be delayed until the delayed benefits of diversity can have an impact. Socratic controllers and Socratic agents are ideal for implementing this kind of delayed-decision evaluation.

It might not be possible to measure the true error rates of the component systems without manual labeling. However, statistical validation can be used to test whether there has

been a statistically significant change in the error rate. Also, the fused system can be used as a reference system for learning-by-imitation applied to the component systems. The learning-by-imitation would include a quantitative measure of the change in error rate relative to the reference system. The objective function could include a measure of the change in error rate rather than the absolute error rate.

### 3.0 Statistical Validation

In the traditional approach to speech recognition research, the data is divided into three sets: a training set, a development set, and a test set. In conventional, supervised training the correct labels are known for the training set. For the test set, the labels are known by the evaluators but are not known by the developers or by the system being tested. The development data is a set of data for which the labels are known by the developers, but the data is set aside and not used for model training. The labels are not known to the training system, but are used by the developers to run internal tests or validations.

In conventional validation, the known labels for the development data are used to measure the error rate of candidate versions of the recognition system, just the way external evaluators use the test data. This validation process allows the developers to judge during the development which version of the recognition system performs best and to guide the subsequent research. It may be called “system validation.” A similar validation process may also be used internally within some automatic learning procedures. If a limited amount of development data is available, the automatic learning procedures, rather than using development data, may set aside portions of the training data to do an interleaved validation process, called cross-validation.

Either of these conventional uses of validation requires that the correct labels be known for the validation data. However, we are concerned with the situation in which there is only a small amount of data that has been manually labeled and a much larger amount of data that has not been manually labeled. In this situation, the labeled data is a scarce resource. It might be impossible to set aside enough labeled data for either form of validation without impacting performance due to reducing the amount of labeled data available for supervised training.

At first it would seem that, even in the situation of semi-supervised learning, the correct labels must be known for the validation data. Indeed the labels must be known to compute the error rate, as is done in conventional supervised validation. However, in both uses of validation, the end purpose is not to measure the performance of the system, but rather to compare the performance of two or more versions of the system. There is no logical requirement to measure the absolute performance of each version, only their comparative performance.

Thus, in statistical validation the idea is that rather than estimate the error rate for each version of the system, we concentrate just on comparing them. Specifically, we set up a Neyman-Pearson test of the hypothesis that the two versions are equal in performance, as measured by a designated statistic (called the “null hypothesis”). We will reject the null hypothesis in favor of either version only if sufficient evidence is accumulated to be statistically significant. Even in conventional supervised validation, the best practice would be to compute the statistical significance of any validation comparison. However, with only a fixed amount of set-aside data, what should be done if the difference in performance in a validation test isn’t statistically significant? Typically, system developers will simply choose the better performing version (or perhaps just the simpler

version) even though further testing, if available, might have shown the other system to be better.

However, the method of statistical validation to be proposed below can be done using automatically labeled data, that is, on data that has not been manually labeled. In the situation that we are considering, there is assumed to be a large quantity of such data. Therefore, if necessary, sequential hypothesis testing may be used in which more data can be used if the evidence is not yet statistically significant. This procedure is the source of the original name for an example of this method called “delayed-decision testing.” Because it does not require manual labeling, the extra data may be acquired automatically during on-going operational use. Then, for a widely deployed system, there is a virtually unlimited amount of data.

To be able to perform statistical validation on data that has been automatically labeled, it is necessary to assume something that is outside the normal framework of semi-supervised learning. In addition to setting aside data that has not been used in the training process, there must also be an additional source of knowledge that has not been used in the training process. That is, not only must the additional source of knowledge not be part of the models being trained, but the knowledge also must not have been used in the training process, for example as extra knowledge to cooperatively-supervise self-training.

This set-aside knowledge source is used to automatically label the data to be used for the validation test. The automatic labeling should be neutral with respect to the difference between the system versions being compared. That is, the automatic labeling should be such that, if the null hypothesis is true for the correct labels, then it is also true for the sequence of automatic labels. This assumption will be true if the automatic labeling process is independent of system versions being validated. It will also be true merely if the automatic labeling errors are neutral between the versions. For example, if the extra knowledge is not sufficient to do the automatic labeling by itself in a stand-alone system, it can be added to a neutral base system. If necessary, a neutral base system can be created by taking an equal probability mixture of the versions. That is, each version is used as a base system and is combined with the extra knowledge. Then, for each test unit (say each utterance) one of the composite systems is chosen at random with each version being equally likely to be used. An existing system that happens to be neutral between the compared versions would be even better as a base system. Surprisingly, the neutral system does not need to have performance as good as the systems being compared [Wh09].

Generally, the “extra” knowledge can be chosen to be knowledge that is by design unrelated to the difference in the versions being compared. Any knowledge that is related to the difference can be included in the base system and be neutralized as described above.

Choosing the statistic to use for the null hypothesis is a development design decision. A statistic that is recommended for simplicity and robustness can be computed as follows.

Let  $D$  be a set of set-aside data. Run recognition on  $D$  using the extra knowledge (with a base system if necessary) and also using each of the system versions being compared. Let  $\{L_1, L_2, \dots\}$  be the sequence of labels produced by recognition using the system with the extra knowledge. Let  $\{V_{i1}, V_{i2}, \dots\}$  be the sequence of labels produced by recognition using system version  $i$ . Let  $\delta_{ij} = 1$  if  $L_j \neq V_{ij}$  and 0 otherwise. Let  $\Delta_j = (\delta_{1j} - \delta_{2j})$ . Then the accumulated differential agreement statistic is given by  $A = \sum_j \Delta_j = \sum_j (\delta_{1j} - \delta_{2j})$ . The null hypothesis is that  $\Delta_j$  is equally likely to be either +1 or -1. Then the statistic  $A$  is distributed according to a binomial distribution  $2 * B(n, p) - 1$ , where  $n$  is the number of terms for which  $\Delta_j$  is non-zero and  $p = 0.5$ . That is,  $Prob(A = (2k - n)) = \binom{n}{k} 2^{-n}$ . Note that this distribution for  $A$  only requires that the null hypothesis be true. It does not depend on the (unknown) distributions of the scores computed by the different system versions.

The statistic  $A$  defined above, however, does not accumulate any evidence in the cases in which the compared versions agree with each other, because then  $\Delta_j = 0$  whether they agree with the automatic label or not. More evidence could be accumulated using a statistic based on (the sign of) the difference in the *scores* computed by the respective versions, without requiring that the score difference produce a difference in the recognition labels.

Let  $W_{ij}$  be the score computed by version  $i$  in data item  $j$  for the hypothesis  $C_j = L_j$ .  
 Let  $U_{ij}$  be the score computed by version  $i$  in data item  $j$  for the hypothesis  $C_j \neq L_j$ .  
 Let  $d_{ij} = W_{ij} - U_{ij}$ .  
 Let  $D_j = 1$  if  $d_{0j} > d_{1j}$  and  $-1$  if  $d_{0j} < d_{1j}$ .  
 Let the null hypothesis be that  $Prob(D_j = 1) = Prob(D_j = -1)$ .  
 Let  $S = \sum_j D_j$ .

The under the null hypothesis,  $S$  is distributed according to the binomial distribution.

Many other test statistics are available to fit particular situations. Choosing what test statistic to use involves considerations of statistical power and robustness. It is a design decision that depends on the particular application. Some examples will be discussed below.

### 3.1 Likelihood ratio based statistical validation.

This section discusses an alternate form of statistical validation that can overcome one of the major limitations of the standard form of statistical validation.

For the task of choosing the better of two pronunciation models, White *et al.* compared unsupervised statistical validation to the traditional supervised method [Wh08]. The traditional method for comparing two pronunciation models is to compute the likelihood for the respective models to generate the observed acoustic feature sequences for labeled instances of the word being modeled. Statistical validation outperformed the tradition method. In addition, statistical validation has the advantage of not requiring manually

labeled set-aside development data. However, the standard form of statistical validation requires the availability of an extra source of knowledge to be set aside during training.

Statistical hypothesis testing could also be applied to likelihood based validation. However, there is usually not enough manually labeled development data to get statistical significance at the levels used in statistical validation on unlabeled data. In some situations, though, a likelihood valuation can be computed without requiring manually labeled data. An instance of such a situation is given in Sample Experiment 8.1: Building a Syllable-Based Lexicon. In the general case, two stochastic generative models are to be compared. However, the models being compared are not models for a single word, but rather are models for the whole language.

Let  $\{Y_t\}_m$  be one of  $m$  sequences of data observations (unlabeled development data).  
 Let  $\{X_t\}_k$  be the (hidden Markov) state sequence for model  $k$ .  
 Each model is evaluated by its likelihood of generating the observed data.

$$(3.1) L_m(k) = \sum_{\{x_t\}} \prod_t \text{Prob}_k(X_t = x_t | X_{t-1} = x_{t-1}) \text{Prob}_k(Y_{t,m} | X_t)$$

The sum is over all sequences  $\{x_t\}$ . Notice that each model includes both an observation component  $\text{Prob}_k(Y_t | X_t)$  and transition component  $\text{Prob}_k(X_t | X_{t-1})$ . The state space for  $X$  is the whole language, not just the states in a single word. This computation of likelihood sums over all possible sequences of labels and therefore does not need to know the correct labeling.

Let the null hypothesis be that either model is equally likely to have a higher likelihood of generating a given sequence of data  $\{Y_t\}_m$  for a particular  $m$ .

Let  $D_m = 1$  if  $L_1(k) > L_0(k)$ ,  
 = -1 if  $L_1(k) < L_0(k)$ , and  
 = 0 if  $L_1(k) = L_0(k)$ .

Then, under the null hypothesis, accumulated values of  $D_m$  have a binomial distribution.  $D_m$  can be used as a test statistic just like in the standard form of statistical validation.

This form of statistical validation also does not require manually labeled data. However, its application is more limited than standard statistical validation. It is limited to maximum likelihood estimation. It cannot directly optimize an objective function such as price-performance. It requires that the models being compared be complete models for the whole language. That is, the models must generate the full observation sequence  $\{Y_t\}_m$  and not just particular words. On the other hand, this form of unsupervised statistical validation does not require an extra source of knowledge that is neutral relative to the difference between the systems being compared.

## 4.0 Learning-by-Imitation

Like statistical validation, learning-by-imitation (LbI) involves the comparison of two systems. However, LbI is a learning methodology rather than a validation method. Moreover, LbI does not seek to decide which of the two system versions performs better. Rather, it uses two versions such that one of the versions (the *reference* version) is designed to be better than the other (the *limited* version).

The principle of learning-by-imitation is very simple: the limited version learns by trying to match the recognition labels produced by the reference version. From one perspective, this is a simple and obvious technique. Similar ideas have been used before. For example, Liang et. al. [Li08] state “Many authors have used unlabeled data to transfer the predictive power of one model to another.” (See also [Cr96], [Bu06]) In particular, the special case of using the detailed match to supervise learning by a Fast Match in speech recognition appears obvious.

However, the principle of Learning-by-Imitation is not intended merely as a technique that can be used in a few special cases, but rather as a paradigm-shifting principle that can change virtually any unsupervised or semi-supervised learning situation into a situation in which supervised learning methodology can be applied. Unlabeled data can be used in essentially the same way as labeled data for the purpose of learning by the limited system.

The generality comes from the fact that the reference system and the limited system are not previously specified systems that luckily have the property that the reference system is known to be more accurate than the limited systems. Instead, the specification only describes a particular situation in which learning is desired. The designer is free to design one or both systems to make sure that the two systems have the relationship of a reference system/limited system pair.

Although superficially similar to cooperatively supervised self-training, LbI is a very different process that uses different assumptions and different resources. The existence of a reference version is outside the usual bounds of semi-supervised learning. Given a specified reference version, the learning may be done by algorithms designed for *supervised* learning but with no requirement for manual labeling. Unlike self-training, the methodology is not limited to parametric generative models nor to maximum likelihood estimation. In fact, for some of the proposed research, LbI will use discriminative models and corrective training. For example, for joint training of multiple classifiers (See 2.4) the objective function must include a measure of the performance of the fused system and cannot simply be maximum likelihood.

LbI can be used for structural learning as well as for model parameter estimation. However, the model space explored by any particular structural learning process should be restricted such that the reference version performance remains better than the limited version performance. For example, the structural learning could optimize a price-performance objective rather than an absolute performance criterion. The reference

system would have more resources (higher “price”) than the limited system. For learning-by-imitation, the reference system needs to be more accurate than the limited system, but it doesn’t need to match its price-performance even though the limited system is being optimized for price-performance.

Many different limited versions can learn by imitation of the same reference system. In such a case, a fusion of the limited versions may have a joint performance that exceeds the performance of the original reference system. With an additional knowledge source not used in the reference system, *statistical validation* can be used to verify that the combined system of limited versions has higher performance. For this purpose, the additional knowledge source does not need to be as accurate as the systems being compared (see section 3 and [Wh09]), Thus, the entire process can be automated with no need for manually labeled data.

In a complex pattern recognition system limited versions of the complete system occur naturally. A particularly interesting example, Fast Match, will be discussed below (See also section 2.1.1). However, an unlimited number of limited versions may be purposely constructed from any given reference version. This process can be done beginning with any pattern recognition system as the reference. Thus, the range of application of LbI is very broad.

When purposefully constructing a limited version, a particular source of knowledge can be highlighted so that the learning improves that particular knowledge element. Consider speech recognition. A very flexible technique for producing limited versions is to rewrite the pronunciation dictionary, selectively reducing the distinction between particular words.

For example, a pair of distinct phonemes can be chosen such that one member of the pair occurs in one of the words and the other member occurs in the other word. Then, in *those two words only*, each of the phonemes is replaced by a new symbol that ambiguously represents both of the pair of phonemes. Other words may also have the same or other phonemes replaced by ambiguous symbols, but the replacement decision can be made independently for each word. The focus of this process is not the phonemes being merged, but rather the remaining distinctions in the words that have been made more similar. In particular, this process can be used to create a large number of minimal pair words for any particular phonetic distinction, by eliminating other differences between word pairs except for the one of interest.

It is obvious that a limited version with artificially introduced ambiguous phoneme clusters will make more errors than the reference version. The artificial creation of minimal pairs creates an opportunity to use LbI to train custom discriminators to distinguish those minimal pairs. Such a discriminator can be used in several ways. It may be used as a feature in a Fast Match computation. For Fast Match, LbI is essentially a supervised learning technique. With a large amount of data (labeled automatically by the reference version), it is almost guaranteed to improve the performance of the Fast Match. The discriminator may also be added to the reference system either as an extra acoustic

feature or as a post-processing secondary test. It is not guaranteed to improve the performance of the reference system, but the potential performance improvement may be verified by statistical validation.

*Fast Match*: Consider a complex pattern recognition system with a large number of classes, such as the number of words in a large vocabulary speech recognition system. A fast computation approximating the standard match computation can greatly reduce the amount of computation. The fast computation, called a Fast Match, is used to quickly evaluate the entire vocabulary. Based on the Fast Match, a (relatively small) subset of the total set of classes is chosen. The expensive standard match computation is only done for the chosen subset.

The approximate scores computed by the Fast Match are only used internally within the Fast Match. All that matters to the rest of the system is the selection of the chosen subset for standard match computation. In fact, despite its name, the Fast Match doesn't need to compute match scores at all, but could use some other mechanism to directly make the selection of the chosen subset. Thus the scores computed by the Fast Match, if any, do not affect the scores computed by the rest of the system. The rank ordering of the chosen subset within the Fast Match computation also does not affect the final answer.

If the standard match computation gives a higher score to an incorrect answer than to the correct answer, then the Fast Match can provide no help to correct this mistake except by a subversion of the basic purpose of the Fast Match. The Fast Match can only correct such a mistake by rejecting the incorrect hypothesis from the chosen subset in spite of the fact that it gets a higher score in the standard match. Such a fortuitous error correction by a Fast Match can occasionally happen by lucky chance, but to attempt to train a Fast Match to do this would violate its central principle.

For discriminative or corrective learning, a Fast Match has an unusual objective function. Because its match scores and any ranking of the chosen subset are only used internally, the objective function does not care whether the designated answer is ranked number one. All that matters is whether the designated answer is within the chosen subset. From the discussion above, what happens to the correct answer also doesn't matter if the standard match computation is not going to choose the correct answer anyway. Therefore, *the objective function does not need to know the correct answer*. Instead, the objective function can be based on whether the label that gets the best match score in the standard match is included in the chosen subset. This objective function will produce the same error rate for the overall system as an objective function based on the correct answer. Therefore, the "designated" answer may be the best-scoring answer rather than the correct answer.

Thus, standard match may be used to automatically label data for supervised training of the Fast Match. This training will be equivalent to supervised training with manually labeled data. This process is also an instance of LbI, with the standard match as the reference system and the Fast Match as the limited system.

There are many other ways to choose a reference system and a limited system for LbI. Specific examples will be discussed in the context of particular experiments (Some examples were already sketched in section 2).

## 5.0 The Need for Proof-of-Concept Experiments

The main techniques in this paper may seem too good to be true. How can statistical validation [Ba06], [Wh09] replace testing on a development set without knowing the correct answer? How can learning-by-imitation use supervised learning algorithms on data that has not been manually labeled? How can either of these techniques work when the extra knowledge source has a high error rate [Ba06], [Ba07]? For that matter, how can standard semi-supervised self-training or cooperatively-supervised self-training [Ma08], [No09] work when the initial automatic labeling has a high error rate? How can unsupervised techniques [Ka05], [Ka07], [Wh09] outperform supervised techniques?

It is reasonable to raise these questions and to insist that they be answered with carefully obtained experimental evidence. Even if the techniques are theoretically sound, do they work in real-world situations? Part of the motivation for the research is to develop techniques that are more robust than existing techniques. Do the techniques presented here stand up in adverse situations?

This paper describes several proposed sample experiments. These sample experiments require little manually labeled data, but may require a large quantity of unlabeled data. There are real-world applications in which a large quantity of unlabeled data is available. Since the objective is to develop robust techniques shouldn't the research concentrate on these real-world applications from the start? In particular, for classified applications shouldn't these techniques be tried on classified data? In fact, should all the research be done only on real data collected in the field, with all of its variability?

There are several reasons to proceed with caution. Many things can go wrong in the real-world besides the direct failure of the techniques. These are new, speculative techniques. They have the potential for high impact if successful. However, they may need further refinement before they work. They may need further development even before they succeed in benign environments.

It is prudent to design a series of proof-of-concept experiments, starting with simpler, relatively benign situations. It is also prudent to first do experiments that control or eliminate some of the sources of variability. Not only does that development strategy provide a proof-of-concept "shakedown cruise," it also allows the sources of variability to be better studied.

So, "No," not all the research should be done only on classified or other "real world" data. Of course, it is important to do some of the experiments on such data. Experiments on real-world data are essential for proving the robustness of the techniques. Experiments based on real applications are important for discovering the remaining problems and setting the priorities for on-going research. However, these experiments are expensive and time-consuming. New, speculative techniques should first be proven in simpler, more controlled experiments.

So first, experiments should be done on unclassified, more controlled data, if the experiments can be designed such that adequate data is available. Experiments can be done first on English data even if the technique is aimed at new languages. To fully isolate sources of variability, initial experiments can be done on speaker-dependent, read speech even though the objective is to work on speaker-independent, conversational speech. The objective is to develop techniques that will be adaptive and robust across changes in condition, including channel, environment and speaking style. They must be robust against the change in condition from the most adverse unclassified research corpora to real-world data. The property of having robustness across changes in condition, however, can already be tested with existing research corpora by starting with even more benign conditions.

## 6.0 Hybrid-Unit, Scalable-Vocabulary Recognition

Hybrid-unit, Scalable-Vocabulary Recognition is proposed as a test bed for novel semi-automatic learning. The system will be a “hybrid-unit” system because it will recognize phonemes, words and, perhaps, syllables all within a single integrated framework. Out-of-vocabulary speech will be handled seamlessly, so the *word* vocabulary can be freely changed to whatever size is needed for a particular experiment, from zero to over a million lexical terms. The number of phonemes and the size of the syllable vocabulary can also be easily changed.

For preliminary proof-of-concept research, a moderate vocabulary of only several hundred words can be used, so most of the speech will be out-of-vocabulary if the task is viewed as a fixed vocabulary, continuous speech transcription task. However, for initial experiments the task can be spoken term detection or even discrete utterance recognition with known vocabulary, so the OOV issue will be postponed. Actually the design of a hybrid-unit system facilitates the handling of OOV issues. These issues are only postponed for the purpose of simplifying the initial experiments.

The phoneme set in the phoneme recognizer and the phoneme set used in the word pronunciation dictionary can be varied independently. The phonemes used in the Fast Match can be different from the phonemes used in the detailed match. Thus, a very rich set of experiments can be designed testing various methods of semi-supervised and semi-automatic learning.

Let  $\{\langle PRec_i, \{P_j\}_i \rangle\}$  be a set of phoneme recognizers  $PRec_i$  with associated phoneme acoustic models  $\{P_j\}_i$ . The phoneme sets in different recognition systems do not need to be the same. They do not even need to represent the same language. The acoustic models may be adapted to a particular speaker. They may be adapted to a particular acoustic environment, speaking style, speaking rate or emotion. The individual acoustic model sets do not need to be adapted to the same or even consistent conditions.

Let  $\{\langle SRec_i, \{P_j\}_i, \{S_k\}_i \rangle\}$  be a set of syllable recognizers  $SRec_i$  with associated phoneme acoustic models  $\{P_j\}_i$  and syllable models  $\{S_k\}_i$ . The phoneme sets and syllable sets do not need to be the same in different systems. They do not even need to represent the same language. The acoustic models may be adapted to a particular speaker, who may be different in different systems. The models may be adapted to a particular acoustic environment, speaking style, speaking rate or emotion. The individual acoustic models do not need to be adapted to the same or even consistent conditions. A syllable set does not necessarily represent the complete set of syllables in a language. A syllable set may include syllables representative of multiple languages. A syllable acoustic model may be a concatenation of phoneme acoustic models or may be a direct syllable-dependent acoustic model, including whole-syllable acoustic features.

Let  $\{\langle WRec_i, \{P_j\}_i, \{W_k\}_i \rangle\}$  be a set of word recognizers  $WRec_i$  with associated phoneme acoustic models  $\{P_j\}_i$  and word pronunciation lexicons  $\{W_k\}_i$ . The phoneme sets and word vocabularies do not need to be same for different values of  $i$ . They do not even

need to represent the same language. The acoustic models may be adapted to a particular speaker or to a particular language. They may be adapted to a particular acoustic environment, speaking style, speaking rate or emotion. The individual acoustic models do not need to be adapted to the same or even consistent conditions. The vocabulary of the pronunciation lexicon does not necessarily represent the complete set of words in a language. The vocabulary of a lexicon may include words representative of multiple languages. A lexicon may also contain models for phrases and multi-word spoken terms. A word acoustic model may be a network of phoneme acoustic models or may be a direct word-dependent acoustic model, including whole-word acoustic features. A word model may also include associated syllable models.

Consider now a sample of speech for which the language, speaker, channel and other environmental attributes are not known. There may be more than one speaker. Any one speaker may be code switching, speaking more than one language in a single utterance. For any conventional word recognition system there would be substantial issues of out-of-vocabulary words and out-of-language words. An appropriate stochastic process to model this situation is a combination of all the systems  $\{\langle \text{PRec}_i, \{\text{P}_j\}_i \rangle\}$ ,  $\{\langle \text{SRec}_i, \{\text{P}_j\}_i, \{\text{S}_k\}_i \rangle\}$ , and  $\{\langle \text{WRec}_i, \{\text{P}_j\}_i, \{\text{W}_k\}_i \rangle\}$ . In mixing systems, it is reasonable to prohibit switching in the middle of a word in a word recognizer and to prohibit switching in the middle of a syllable in any case. Of course, the knowledge of the location of a word or syllable boundary will only be an estimated probability distribution.

This representation allows OOV and OOL instances to be modeled easily and naturally. Even the simplest multi-system, with a single phoneme recognizer and a single word recognizer can represent OOV instances as well as a stand-alone phonetic recognizer. The representation of OOV words is so seamless that the lexicon can be shrunk to a smaller vocabulary with robust performance that still exceeds the performance of a stand-alone phoneme recognizer. It would be robust in the sense that the OOV performance degradation relative to a system with a larger word vocabulary would be gradual.

The proposed architecture provides a principled means of representing out-of-vocabulary situations both for spoken term detection and for large vocabulary transcription tasks. For transcription such an architecture can even lower the average word error rate because, for at least some of the out of vocabulary words, the correct spelling may be guessed from the recognized phoneme sequence. Furthermore, despite its apparent complexity, there are implementations of this hybrid-unit architecture that use phoneme and syllable recognizers as a fast match for the word recognition system and thus take less computation than a conventional word recognition system.

However, there are several reasons that such an architecture has not been universally adopted as a solution to the out-of-vocabulary problem. Some of the reasons relate to limitations in current implementations and will not be discussed further. Some of the reasons, however, relate to significant technical issues. One important issue is the need to train the probability models for switching between the systems.

Maximum likelihood estimation is not recommended for training the switching probability models. Even the simplest case of switching, from in-vocabulary to out-of-vocabulary, is like a composite language model probability. But it is well-known that systems perform better if there is an empirically estimated *ad hoc* adjustment of all language model scores relative to acoustic scores. This adjustment would not be needed if the acoustic models computed accurate estimates of the true probabilities. In practice, however, acoustic models systematically misestimate the probabilities because of conditional independence assumptions in the model that are violated by real speech.

Since an empirically determined adjustment would be necessary in any case, it is recommended that the switching probability be empirically estimated from the start. To reduce the complexity of empirical hill-climbing in n-dimensional space, the following procedure is recommended:

- 1) Begin with a simple architecture, say just two systems. Train the switching probabilities for this simple system based on an application-relevant objective function.
- 2) Use the existing system with trained switching probabilities as a base system. Hold the switching probabilities among the existing systems fixed. Add a small number of additional systems to the architecture, perhaps only one. At first make the probability of switching to any new system very small. The performance should be very similar to the performance of the base system. If the performance of the new system is worse than the performance of the base system, decrease the probability of switching to the new systems.
- 3) Repeat step (2) adding more new systems. Optionally, step (2) may be interleaved with re-estimation of the switching probabilities in the base architecture and/or with re-estimation of the acoustic models in one or more of the systems. It is recommended that all the model re-estimation be based on a shared objective function (therefore, not based on maximum likelihood of acoustic models in individual systems). Then, each step in the interleaved process will improve in this shared objective function and will therefore converge at least to a local optimum. Continue to repeat step (2) as long as there are more systems to be added to the architecture.

The empirical hill-climbing in steps (2) and (3) can use standard optimization methods if a sufficient quantity of labeled data is available. At the worst, the estimated objective function will have to be treated as the true objective function plus a noise term. Even that case can be handled with existing techniques, for example Robins-Monroe stochastic approximation [Ro51], since step (2) can be reduced to a one-dimensional optimization problem.

However, with a limited amount of labeled data and a relatively large amount of unlabeled data, the use of learning-by-imitation and statistical validation is proposed.

The following example shows one way they might be used in an example task involving two languages.

Consider two languages, language A and language B. Assume that there is an existing phonemic or phonetic recognition system for language A. Assume on the other hand that nothing is known about language B, not even the phoneme set. Assume that there is a large quantity of speech data that includes speech from both language A and language B, but that the data has not been labeled. Assume that the data has not even been segmented and labeled as to which language is being spoken at a given time. Build a broad phonetic recognition system for language B by the following procedure:

### **Sample Experiment 6.1: Building a phonetic recognizer for a new language from scratch.**

- 1) Recognize all of the data using the phoneme recognizer for language A. Create initial acoustic models for a number of different phonetic recognition systems with different phone sets by clustering, splitting and merging the sets of instances that are automatically labeled with each of the phonemes in the original language A system. However, restrict each of these new systems so that initially it will perform less well on a mixture of language A and language B than does the original language A recognizer. For example, if language B is suspected of having phonemes that do not occur in language A, the new systems can all be constructed by merging phonemes in the language A system without adding any new phonemes. Before further training, these merged systems will perform less well on both language A and language B than does the original language A recognizer. However, if many different merged sets are formed their fused performance after training will be capable of exceeding the performance of the original system. Language B may include phonemes that share allophones with phonemes in language A, but that group them differently. The fused multi-systems may be able to distinguish these allophones and correctly label them phonetically.
- 2) Train each of the new systems by learning-by-imitation using the original language A system as the reference. The new systems can be trained collectively as discussed in section 2.4. Whether the new systems are trained independently or collectively, continue learning-by-imitation until the performance of the fused system exceeds the performance of the original language A system. Because the individual systems are trained by learning-by-imitation with the original A system as a reference, the individual systems are not expected to exceed the original system in performance. Evaluate comparative performance using statistical validation. If the performance of one or more individual new systems approaches that of the original language A system without the fused system exceeding the language A performance, include the language A system in the set of systems to be fused. Unless the best performing new systems are all highly correlated with the language A system, their fusion with the language A system should perform better than the language A system by itself. If they are all highly correlated, then

- any of them or the original language A system can be used as language B phoneme recognizers and steps 3 and 4 may be skipped.
- 3) If the fused system performs better on the mixed language data than does the original language A recognizer, then temporarily freeze the fused system and use it to train the language A system and variants of it to better recognize language B. Unlike the systems in step (1) these variant systems are intended to exceed the performance of the original language A system and may create models for new phones that are not in the original phoneme set for language A. They may be either new sounds or finer phonetic transcriptions of allophones of phonemes in language A. Statistical validation may be used to do structural learning such as creating additional phones. Repeated use of statistical validation will result in the false rejection of the null hypothesis for a fraction of the trials, bounded by the value set as the level of statistical significance. A proposed method for handling this problem, multi-level validation, will be described elsewhere (see section 7.0, “Multi-level Statistical Validation”). Improve the performance of the variants of the language A system using the fused system from step (2) as the reference. Continue improving the performance of these variants until the performance of their fusion exceeds the performance of the fusion created in step (2). Then let the fusion of these variants be the new reference system. This fused system will also be a phonetic recognition system for both language A and language B. Its symbols will not necessarily represent language-specific phonemes for language A.
  - 4) Continue steps (2) and (3) alternating between the two fusion systems as each one gets better than the other. In the description of step (2) use the new fused system from variants of language A whenever the original step (2) used the original language A phoneme recognizer.

Note: the fused phonetic recognizers built in steps (2) and (3) will be trained to recognize the mixture of language A and language B, with whatever mixture proportions and switching rates occur in the given data. A regularization function biased toward a lower switching rate can help cause the individual systems to specialize in one of the languages. Alternately, a bias towards a high switching rate will help steer the individual phonetic systems to language independence across the two languages.

Supplement: Suppose there is a list of words that are known to occur in language B. It doesn't matter whether or not these words also occur in language A. Then any of the individual or fused phonetic recognition systems may be used as a fast match for word detection and ranking of word candidates (See section 2.1.1 and Sample Experiment 6.2). Then a spoken term detection system for the given vocabulary may be used as the reference system in step (2). Another choice for reference system in this case would be a hybrid phone-word recognition system. For step (3) and for step (2) in later rounds, a word list from language A may also be used in the reference system. As a development design choice, which may depend on the intended application, a combined word list from both languages may be used in the reference system or a training strategy may alternate between using a language A word list and a language B word list.

As described, this experiment automatically builds a phonetic recognizer for a new language even without a lexicon for that language. If a large pronunciation lexicon is available, the reference system in the experiment may be the phonemic transcription from a large vocabulary continuous speech recognizer. It would only be necessary to learn the acoustic models. The phoneme set would be known. If the out-of-vocabulary rate is moderate, the procedure in the experiment could build a phoneme recognizer for the new language because it has a phonemic reference. A regularization function would need to be used to get the two fused systems that are used alternately as references to each specialize in its respective language.

*Self-training language diarization:* If, in at least one of the fused systems, the individual phoneme recognition systems are trained to specialize in one of the languages, then the latent state variables will be implicitly performing language identification as a by-product of dynamically selecting the best matching phoneme set for a given speech interval. The speech intervals may be probabilistically labeled with this language identification. In addition, the probability of switching to or staying in a given individual phoneme system may be trained as a probability conditional on the language identification. This process may be iterated as an instance of the EM algorithm to train the acoustic models in the phoneme sets to be even better adapted to their particular language.

### Objectives of experiment 6.1

- 1) Proof-of-concept for technique to build a phonetic recognizer in a new language without either a lexicon for the language or any manually labeled training data.
- 2) Proof-of-concept for technique of building and automatically teaching systems that are limited even relative to a *mediocre* reference system (the phonetic recognizer for language A applied to data from language B).
- 3) Basic proof-of-concept for general principle of learning-by-imitation.
- 4) Exploration of technique for creating fused systems that may outperform the reference system even though individual components are trained by learning-by-imitation.
- 5) A proof-of-concept for statistical validation in another task in addition to the pronunciation validation task.
- 6) Exploration of iterative performance improvement based on successive rounds of learning-by-imitation, system fusion and statistical validation.
- 7) Exploration of multi-level statistical validation.

### Sample Experiment 6.2: A fast hybrid recognizer.

A hybrid recognition system is proposed that is faster than a conventional large vocabulary continuous speech recognizer and that handles out of vocabulary items more gracefully. It is expected to be both faster and more accurate than a conventional phoneme-based spoken term detection system. It can be trained semi-automatically using data that has not been manually labeled.

Assume that the task is spoken term detection in a given language. Assume that there is at least one phonetic or phonemic recognition system for the language. There may be multiple phonetic systems. To reduce computation, these systems may actually recognize a smaller set of broad phonetic classes. Different systems may specialize in different classes.

Assume that there is a list of words that are likely to occur in spoken term queries. If no such list is available, a list of words that are merely known to be likely to occur in the speech data sample may be used. Given the list of words, construct a fast match system to match the acoustic observations at a given point in the speech data and rapidly select a small subset of the word list as likely to match well against the acoustics observed at that point in the speech. The more expensive standard word match is computed only for the small subset of the word list. One such fast match system uses the phonetic recognizer(s) as follows:

- 1) Run each of the phonetic recognizers to produce a sequence of phone or phonetic class labels. Each sequence of phones will be used as a stream of finite-valued acoustic feature observations. The times of the phone boundaries do not need to be synchronized among the streams. The methodology may be generalized to use results lattices from the phonetic recognizers, but that refinement is not necessary for this sample experiment.
- 2) Given the list of target words, run a hybrid phone-word system as a *reference* system. This reference hybrid system will not be limited in its amount of computation. It may use additional, more computationally expensive phonetic recognizers than the price-performance optimized recognizers introduced in step (1). It will probably not itself use a fast match because it will not do any fast match pruning of the vocabulary to a subset. However, the fast hybrid system will be a limited version of the reference hybrid system. The fast hybrid system will use the phone-based fast match described below, so the reference hybrid system will be a true reference for the fast match.
- 3) Construct and train one or more decision trees as a fast match. Each node in each decision tree will ask a question about the identity of one of the phonetic labels in one of the acoustic feature streams. The questions will “look ahead” from the putative starting time for a word. Which position in the selected phonetic stream is being queried may be determined relative to questions that have been asked at higher levels in the tree. Bottom-up, approximate syllabification will be performed. A question in the decision tree may be for a phone position relative to a syllable count and a phonotactics analysis of the syllable. Each leaf of each decision tree will have a pre-computed list of words that are likely to produce acoustic streams that match the answers to the questions that follow the branches to that leaf of the decision tree. If there is more than one decision tree, an empirically tuned fusion algorithm will combine the leaf lists from all the trees into a single list.
- 4) The phonetic recognizers, the decision trees, and the fusion algorithm will all be trained using *learning-by-imitation* with the reference hybrid system as the reference for LBI. The objective function will be a price-performance objective for the fast hybrid system. For the purposes of training the fast match, the

- detailed word match in the fast hybrid system will be held fixed. Thus the amount of computation required for the fast hybrid system is the sum of the time required to do the fast match plus the time it takes to do the detailed match, which will depend on the length of the word list subset returned by the fast match. With a moderate-sized target word list, the amount of time for the detailed match will be roughly proportional to (i.e. grow linearly in) the length of the subset list. The relative cost of the fast match and the detailed match would be determined with respect to the particular fixed word match method and models.
- 5) The word match of the fast hybrid match will also be optimized *by learning-by-imitation*. Because this optimization will be based on a price-performance objective function, a single, fixed reference system will be adequate. Once the word match has been optimized, it may again be frozen and step (4) be repeated. Iteration of steps (4) and (5) will lead to convergence to a joint (local) optimum in the price-performance objective function.

Recommended design choices:

- 1) Use context independent phonetic models. For the fast match, simpler, faster models are preferred. The recognized phones are also used for spoken term detection, but in the hybrid STD system they are only needed for query words that are not in the given word list. Hence the phone-based index does not need to be as comprehensive or as accurate as a stand-alone phonetic recognizer for STD.
- 2) Use only a small number of Gaussian components for the phonetic models. Many phones may need only one Gaussian. The same reasoning applies as in recommendation one.
- 3) Evaluate a variety of detailed word match alternatives for the fast hybrid system. Simply using discrete alphabet models for the multiple phonetic streams may be adequate. On the other hand, for a moderate-sized word list, the fast match may already cut down the detailed match word list to such a small size that much more sophisticated word models will still be cost effective.
- 4) Use techniques that trade some performance in the word match for speed. The objective in this experiment is not to match or beat the performance of a standard LVCSR-based STD system, but rather to beat both the speed and performance of a conventional, stand-alone phonetic-index-based STD system. Even word models that give up some accuracy for speed will have substantial knowledge that is outside the scope of a stand-alone phonetic-index-based STD system.
- 5) Use full context-dependent phoneme models for the reference hybrid system. Use full large vocabulary continuous speech with the best language model available. Train these models using cooperatively-supervised self-training. The vocabulary does not need to be restricted to the limited vocabulary of the fast hybrid system. However, if the vocabulary is limited because only a limited pronunciation lexicon is available in the target language, the vocabulary may be limited to whatever is available. As a hybrid system, the reference system will seamlessly handle out-of-vocabulary items.
- 6) Especially if the pronunciation lexicon is limited, consider including syllable recognition in the reference hybrid system. If a full syllable pronunciation lexicon is not available, build one automatically [see sample experiment 6.1].

Notice that the recommended design choices tend to increase the performance difference between the word recognition component and the phonetic component as well as the difference in performance between the reference hybrid system and the fast hybrid system. Thus, the ability of the reference hybrid system to serve as a reference for learning-by-imitation will be enhanced.

This sample experiment uses a fast match as a vital element to achieve the desired price-performance. It utilizes the ability of learning-by-imitation to optimize an objective function based on price-performance rather than just maximum-likelihood. Thus, the price-performance can be optimized automatically without extensive manual tuning.

### **Objectives of experiment 6.2**

- 1) Proof-of-concept for learning-by-imitation in which the limited system is a fast match.
- 2) Proof-of-concept for learning-by-imitation with an objective function based on a price-performance combination.
- 3) Exploration of fast match techniques based on decision trees.
- 4) Exploration of multiple phonetic-class-based data streams.
- 5) Proof-of-concept for learning-by-imitation for a complete system (fast hybrid system) based on a less constrained reference system.
- 6) Construction of a fast hybrid STD system.

## 7.0 Multi-Level Statistical Validation

Development of an improved speech recognition system typically utilizes a set of labeled data that has been set aside for development testing. As different versions of the system are developed, the performance of each new version is tested on the development set. The question of whether an observed improvement is statistically significant is almost an afterthought. In fact, even if the improvement is not statistically significant, there is usually no better alternative than to take the best performing system as the new baseline. In particular, there is usually no more labeled data available to resolve whether the better performing system will continue to perform better on additional data.

Semi-supervised statistical validation has advantages. Testing statistical significance is at the core of the technique. Furthermore, it does not require labeled data, so more of the limited amount of labeled data is available for supervised learning and other purposes. Because in many situations a large amount of unlabeled data is available, if the initial development test sample isn't large enough for the result to be statistically significant, the decision can be delayed and the hypothesis can be tested on more data. Thus, even though it only requires unlabeled data, in operational use it can perform as well as traditional supervised development testing. If there is enough data, it can perform better.

However, overuse of statistical hypothesis testing, whether supervised on labeled data or not, creates another danger. In any statistical hypothesis testing, it is to be expected that up to 5% of the trials will reject the null hypothesis with statistical significance at the 0.05 level even though the null hypothesis is true. Therefore, if there are 100 tests up to five of them are expected to falsely show a statistically significant statistic. One approach would be to keep the number of development trials as low as possible and to always test for a higher level of statistical significance.

However, avoiding doing frequent development tests runs the danger of not knowing whether a particular change made an improvement in the system or not. Without this knowledge, incorrect decisions can be made in the subsequent research plan and the problems could cascade. If more than one change is made in the system before the new system is tested, there are often confusing and conflicting performance results. It can be very difficult to sort this situation out if the changes interact with each other.

Fortunately, with extra unlabeled data, better alternatives are available. One alternative would simply be to always test for a very high degree of statistical significance. Another alternative, which can be applied more selectively, is to perform multi-level statistical validation. Multi-level statistical validation uses an independent set of set-aside data to run an additional statistical validation test. If the null hypothesis is true, the probability that the null hypothesis will be rejected at the 0.05 significance level in two independent tests is only  $0.05 * 0.05 = 0.0025$ , or one chance in 400. If a larger number of statistical hypothesis tests are being done, three-level testing can be used.

In many situations there is a nearly unlimited amount of unlabeled data. Therefore, a large number of independent development test sets can be set aside. The ability to do a

large number of statistical validations creates the opportunity to automate the kind of version testing that was described in the first paragraph of this section, which is normally done manually. That is, if a mechanism is developed to automatically create new versions of a system then, with sufficient (unlabeled) data, statistical validation can automate the process of hill climbing to find an optimum version of the system. Research that would normally require a sequence of many experiments with manual intervention in between, could be automated into a single experiment.

## 8.0 Syllable-Based Recognition

Syllable-based speech acoustic modeling [Ga01] has many potential advantages, especially within a hybrid system, over using only phoneme-based or word-based approaches. Although alphabetic writing systems mark word boundaries more clearly than syllable boundaries, there are widely used writing systems for which the same is not true. Chinese, the most widely used language in the world, has an almost one-to-one correspondence between characters and syllables, but there is substantial ambiguity as to what substrings in written text represent distinct words versus word fragments or multi-word phrases.

The syllable is a natural unit for acoustic modeling because the actual acoustic realization of a given sound is strongly dependent on its relative position in the syllable. Sounds in the onset of the syllable are often pronounced more clearly and distinctly than the same phonemes when they occur in the coda of the syllable.

There is substantially more redundancy in a syllable than in an individual phoneme. Only a small fraction of the possible phoneme sequences actually occur in the syllables in any given language. This redundancy aids syllable recognition. More significantly for the present discussion, it provides an extra source of knowledge to enable techniques of semi-automatic learning, such as cooperatively-supervised learning, statistical validation, and learning-by-imitation.

### Sample Experiment 8.1: Automatically Learning a Syllable-Based Lexicon

This project builds a pronunciation lexicon for a language with a syllable-based orthography. That is, the task is to build a lexicon representing the possible pronunciations for each written character, where each character usually corresponds to one syllable. The written language could be semanto-phonetic like Chinese or Japanese Kanji. It could be a syllabary like Japanese Hiragana, Katakana or Cherokee. It could be a syllabic alphabet like Devanagari, and other Indic languages, though such a language could also be approached as a normal alphabet language.

It is assumed that text data is available for the language and that a phonetic recognition system is available for the language (See Sample Experiment 6.1 for a way to develop such a phonetic recognizer from scratch with semi-automatic learning). It is assumed that there is a large amount of unlabeled speech data for the language, but only a little manually labeled data. If a small “starter” character pronunciation lexicon is available, there doesn’t need to be any manually labeled speech data.

The procedure should be as automatic as possible, with minimal human intervention. It should apply to as many languages as possible. Therefore, the concept of “syllable” will have a loose interpretation. The syllables as discovered by the automatic procedures may

differ from the syllables as specified by linguists. It will be sufficient that the recognition system will be able to represent and learn most of the syllable-based knowledge.

It is also assumed that existing syllable detection software is available. Preferably, the syllable detection software will operate language-independently. It will be sufficient if the syllable detection software can mark the approximate location of the syllable nucleus. It does not need to locate the nucleus precisely in time. It does not need to determine the location of the syllable boundaries. The likely location of a syllable boundary in a sequence of phonemes is dependent on language-specific statistics, which will be estimated as part of this project. It is assumed that the syllable detection is not perfect, so the syllable marking will be modeled as a stochastic process.

### **Step 1: Learning a syllable grammar.**

The first step of the recommended procedure is to learn a probabilistic grammar that can generate all the one-syllable phoneme sequences that occur in the language. The proposed procedure does not require any manually labeled speech data.

Only a small fraction of all phoneme sequences correspond to valid syllables. There are only about 130 different syllables in Japanese and only about 400 phonetically different syllables in Mandarin (Tones will be treated in a separate experiment). In all languages, phoneme sequences obey certain constraints that limit the number of possible syllables. The purpose of this step is to capture this language-specific redundancy. This step is also a proof-of-concept for the general technique for learning a grammar semi-automatically without manually labeled data.

Although the general technique is not limited to finite state grammars, it appears that a finite state grammar will be sufficient to represent the phoneme sequences in the syllables in almost any known language. For this experiment, the grammar inference will be limited to finite state grammars. A candidate grammar can be represented as a finite graph. The grammar learning algorithm can either add or delete arcs from this graph.

#### ***Technique 1a: Self-training syllable acoustic models.***

The syllables in a given language will be represented as a finite state (probabilistic) grammar. Equivalently, it can be represented as a finite directed graph or as a (hidden) Markov process. It will be convenient to use the grammar and directed graph representations because the inference algorithms to be used in later steps will include adding and removing arcs from the graph as well as the algorithms that are traditionally used for training models of a hidden Markov process. The representations and their associated terminologies will be used interchangeably. The techniques can be extended to more general grammars or to more general graphical models, but the acoustic syllable models used here will only be finite state, so the three representation methods will be mathematically equivalent.

When a hidden Markov process or a finite state language is represented as a graph the generation of the acoustic observations may be associated either with the arcs or with the nodes. Neither representation is more general. Any representation of one type can be converted automatically to a representation of the other type (possibly with a different number of states). In speech recognition, most systems represent the acoustic observations as associated with the nodes [Hu01], although at least one leading system associates them with the arcs [Je97]. In representing a grammar or language model, the generated observations (words) are traditionally represented as being associated with the arcs. Representing the observation generation as associated with the arcs is more convenient for computing and representing the composition of models. The arc-based representation will be used here to simplify the following discussion of composition.

### Composite Graph

Let  $M = \langle S, A, T(a), L(a) \rangle$  be a finite state, probabilistic, labeled directed graph representing the model for sequences acoustic syllables in the given language.  $S$  is the set of states or nodes.  $A$  is the set of directed arcs.  $T(a)$  is the conditional probability of following arc  $a: s(a) \rightarrow d(a)$  from its source node  $s(a)$  to its destination node  $d(a)$ , if the process is current in the state  $s(a)$ , that is,  $T(a) = \text{Prob}(a|s(a) \rightarrow d(a) | S = s(a))$ .  $M$  is called the “language model.” For example, any  $n$ -gram language model may be represented as such a finite state network. There is a label  $L(a)$  associated with each arc  $a$ . The labels will be associated with the syllable acoustic models defined below.

Let  $\{G_k = \langle S, A, T(a), L(a) \rangle_k\}$  be a set of finite state, probabilistic, directed graphs, each representing the possible pronunciations (i.e. phoneme sequences or phonetic sequences) associated with a particular syllable. These are the syllable acoustic models. The labels  $L(a)$  on the arcs in these graphs are phonemes or phonetic labels, depending on how the models are trained. The labels on the arcs may be phonemes even if the acoustic observations are labels generated by a phonetic recognizer rather than a phoneme recognizer. If so, the model for generating a given observation ( $\text{Prob}_m(x_i | L(a))$  in the equations below) will include the probability of generating a particular allophone given the phoneme.

The composite graph  $C(M, \{G_k\}) = \langle S, A, T(a), L(a) \rangle_C$  is formed by replacing each arc  $a$  in  $M$  by the network  $G_{L(a)}$ . This composite network is still a finite state network, equivalent to a finite state Markov process. Because the arcs in  $M$  have been replaced by networks from  $\{G_k\}$ , the labels on the arcs in the composite graphs are phonemes.

The equations for training the parameters in such a labeled, directed graph representation of a hidden Markov process are presented in Appendix A. Other than the use of directed graph notation rather than the conventional hidden Markov representation, Appendix A presents the standard Baum-Welsh algorithm that has been used for years for training acoustic models [Bau72, De77, Ba74, Ba75]. The directed graph notation is particularly convenient for representing the construction and training of composite graphs, as done in this Sample Experiment.

The acoustic observations in the case of interest are the output sequence of a phonetic or phonemic recognizer. This recognizer may make insertion and deletion errors. These insertion and deletion errors also can be modeled by finite state networks. Assume that the composite  $C(M, \{G_k\}) = \langle S, A, T(a), L(a) \rangle_C$  also has been composed with the finite state networks representing insertions and deletions. This means that the equations (A.1) through (A.7) below will automatically take account of insertions and deletions, although the graphs will be more complex.

The following discussion will only concern the composite graph, so the subscript C will be dropped from the notation. Remember, however, that this graph represents the stochastic process for generating all of the syllables of the language. It does not represent a particular syllable or a particular word.

### Creating Initial Syllable Acoustic Graphs

The later steps of the learning procedure will add and remove arcs for the syllable acoustic graphs  $\{G_k\}$ , so it is not essential to get the initial versions of these graphs to be exact. Therefore, there are several strategies for creating these initial graphs.

One strategy is to use a graph that represents all the syllables in almost all of the known languages. There are certain relationships among the phonemes in a syllable that are true for most languages. Every syllable has a *nucleus* and optionally has an *onset* and a *coda*. The onset is the part preceding the nucleus; the coda is the part following the nucleus. The nucleus must contain at least one vowel or must consist of a syllabic consonant.

If there are several vowels in a syllable, they must be consecutive. There are only a few consonants that may be syllabic. In most languages, any syllabic consonant is an allophone of one of the consonants m, n, l or r. Some languages have “fricative vowels” which are really syllabic fricatives phonetically. Mandarin Chinese, for example, has three. However, one phonemic analysis of standard Mandarin treats these syllabic fricatives as allophones of the phonemic sequence with “null” in the vowel and coda positions in the syllable final.

There is a rank ordering of the phonemes such that when two phonemes are adjacent within a single syllable with a vocalic nucleus, the lower ranked phoneme is always further from the nucleus. Under this principle, vowels are ranked higher than any consonants. Glides, such as /w/ and /j/ (usually written “y” in English), are semi-vowels or non-syllabic vowels that form diphthongs with other vowels. They must be adjacent to a vowel, and hence have higher rank than any consonant. Sonorant consonants, such as /r/, /l/, and the nasals, rank below the vowels and semi-vowels but above other consonants. Voiced consonants generally rank higher than voiceless consonants.

Based on these and other rules, one strategy would be to start with an acoustic syllable network that tried to approximate a universal syllable generator for all languages. It doesn't need to cover all possibilities, because the learning process can add additional arcs and nodes to the network later.

A second strategy is to start with a very simple syllable structure, and then increase its complexity during the learning process. There is variation within this strategy. The simple syllable structure may be designed to over generate, generating many phoneme sequences that do not actually occur in the language. Gradually, a more complex network can be learned that models some of the context conditions in which certain phonemes do not occur. Another variant strategy is to start with a simple network that under generates. Then gradually a more complex network can be build to model some of the observed phoneme sequence that cannot be generated by the simple network.

### Case Study: Mandarin

If some information is available about the target language, a third strategy is to design an initial network that already represents some of this available knowledge. Many languages have many fewer different syllables than English. For example, many languages limit the number of consonants that can occur together in the same syllable. Standard Mandarin is an interesting case study that shows some of the interaction between phonetic transcription and language-specific phonemic transcription.

In standard Mandarin a syllable consists of two parts: an *initial* and a *final*. The initial may be null or a single consonant. The final, is the rime of the syllable, the nucleus plus the coda. Phonetically there are 14 vowels, but they occur only in certain sequences. Most of the vowel pairs do not occur as the distinguishing feature in any minimal pair of words, so the vowels may be grouped into a much smaller number of phonemes.

The final of a syllable may be described as having three slots in the form (G)(V)(C). The first slot, called medial because it lies between the initial consonant and the main vowel, is optional or it may be any of the three glides corresponding to the high vowels /i/, /y/ and /u/. The main vowel may be either of two phonemes /ə/ or /a/, which have many allophonic variations. The main vowel slot may also be null, in which case the medial becomes the only vowel in the syllable. The coda may be null or either of the glides /i/ or /u/ or either of the nasals /n/ or /ŋ/.

Thus the syllables of standard Mandarin, ignoring tones, may represented by the following graph:

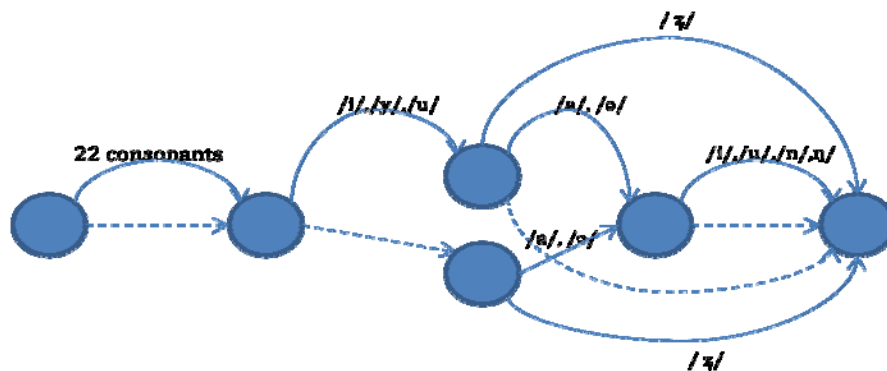


Figure 8.1

However, this is a phonemic network. Each of the vowels has several different phonetic realizations, depending on the context in which the vowel occurs. The phonetic realization also depends on the dialect. Correctly representing Chinese dialects, however, is beyond the intended scope of this experiment. The following table from Wikipedia [Wi09] shows the phonetic realizations for the phonemes in the final, as a function of context.

When the medial, nucleus, and coda combine into a final, their pronunciations may be affected. The following is the full table of finals (except an optional final /r/) of Standard Mandarin in the [IPA](#):

Nucleus	Coda	Medial			
		Ø	i	u	Y
A	Ø	a	ia	ua	
	I	ai		uai	
	U	aʊ	iaʊ		
	N	an	iɛn	uan	yɛn
	ŋ	aŋ	iaŋ	uaŋ	
ə	Ø	ɤ	iɛ	uo <sup>1</sup>	yɛ <sup>2</sup>
	I	ei		uei	
	U	ɤʊ	iɤʊ		
	N	ən	in	uən	Yn
	ŋ	ɤŋ	iŋ	uɤŋ <sup>3</sup>	yʊŋ
Ø		z□	i	u	Y

<sup>1</sup> Both [pinyin](#) and [zhuyin](#) have an additional "o", used after "b p m f", which is distinguished from "uo", used after everything else. "o" is generally put into the first column instead of the third. However, in Beijing pronunciation, these are identical.

<sup>2</sup> Another way to represent the four finals of this line is: [ɯɤ iɛ uo yɛ], which reflects Beijing pronunciation.

<sup>3</sup> /uɤŋ/ is pronounced [ʊŋ] when it follows an initial.

Clearly the phonetic table can also be represented as a finite graph. For example, it can be represented as a tree graph, as can any other finite language. Depending on the application and other considerations either a phonemic graph or a phonetic graph may be preferred. Because the phonetic form is predictable, a phonemic graph with context-dependent acoustic models can represent and learn the knowledge in the phonetic graph. The phonetic graph, however, might be able to use simpler, context-independent acoustic models.

Because there is considerable difference between the phonemic representation and the phonetic representation (especially for the phoneme /ə/), different results may occur depending on whether the initial graph is a phonetic graph or a phonemic graph. Assume for the moment that the phonetic or phonemic recognizer for the language is phonetic, since no knowledge has yet been introduced to train it to do language-specific clustering of phones. If the initial graph is phonetic, and if the acoustic models are context-independent, then the learned graphs are also likely to be phonetic in character. However, in later steps of the process when pronunciations are associated with written characters, it may be possible to learn the clustering of allophones into phonemes.

However, recall that in sample experiment 6.1, the recognizer being learned for language B might be trained to be phonemic if some manually labeled data or a moderate size phonemic lexicon is available. If a phonemic recognizer is available and the initial graph is phonemic, then it may be possible to keep a more or less phonemic representation through the training process.

A final strategy for choosing initial syllable graphs is to use a mixture of many different graph types. Let the learning process discover which graph types work best for the target language. The process may well learn to use one graph type for some syllables and another type for other syllables. A system with many graph types will necessarily be a phonetic representation because it would be impossible to restrict such a system to only make phonemic distinctions. A mixture of any finite number of finite state graphs is still a finite state graph. Despite the complexity of this graph, it is the recommended procedure for most situations. It has the major advantage that it is capable of learning a custom graph for each syllable just by learning the transition probabilities in the large mixture graph, rather than using the more complex process of adding and removing arcs from a graph.

Conventional semi-supervised learning would have needed labeled data at least to create the initial acoustic models for self-training. However, Sample Experiment 6.1 showed how to create initial phoneme models in a new language without any manually labeled data in that language.

Note that the likelihood  $P_m$  for the converged model is also a measure of the quality of the trained models. It is useful for comparing syllable graphs with different topologies, such as by adding or removing arcs and nodes.

### *Technique 1b: Comparing the performance of two syllable grammars.*

Let  $F$  and  $G$  be two composite syllable grammars for the same language. The likelihood score  $P_m$  can be used to compare how well each of these grammars matches the training sample. However, the likelihood score on the training set does not really tell us anything because a graph that is a superset of the other graph can always be trained to score at least as well as the smaller graph. The more complex, better scoring graph may be over trained, so even though it matches the training data better, that performance will not hold up on test data. Thus, the grammars  $F$  and  $G$  should be compared based on an estimate of their performance on independent test data.

Traditionally in speech recognition some of the manually labeled data is set aside as development data, just to allow such independent tests. In this experiment, using the trained models a computation of equation (A.1) on the development data can be done to compute  $\alpha_m(s, T)$ , which then can be used to compute  $P_m$ . If the more complex graph is over trained, its likelihood score on independent test data will generally be poorer than the likelihood score of the simpler model. However, the likelihood score on a given sample of data varies somewhat due to idiosyncrasies of the particular data sample. Therefore, a test of statistical significance should be performed. Because, the likelihood may be evaluated without requiring manual labeling, likelihood-based statistical validation may be performed (See section 3.1 “Likelihood-based statistical validation”).

Thus, likelihood-based statistical validation allows two grammars with different topologies to be compared. Such a comparison enables a hill climbing algorithm that explores the space of grammar topologies.

This hill climbing requires repeated use of a statistical significance test. Since there is always a non-zero probability that such a test will falsely reject the null hypothesis, further caution must be imposed. With repeated use, it is guaranteed that eventually a null hypothesis will be falsely rejected or be rejected in favor of the wrong alternative. The use of multi-level statistical validation will help reduce this problem. Furthermore, there are algorithms, such as simulated annealing [Ki83], that not only tolerate some amount of hill climbing steps in the “wrong” direction, but actually take advantage of them.

Notice that the EM training of step 1a must be done separately for each topology compared in step 1b. Therefore, hill climbing in step 1b may require a large amount of computation. The amount of topology changing hill-climbing will be reduced if the base models already include a selection of different topologies of syllable acoustic graphs, as was recommended in step 1a.

Note that, so far, no association has been made between acoustic syllables and written characters. The acoustic syllable models have been learned independently of their written forms.

## Step 2: Determining which Acoustic Syllables Correspond to which Characters

This step requires speech data for which the character transcription is known. That is, the transcription for the speech as a whole is known in the form a character sequence, but the time alignment of that character sequence to the acoustic feature sequence is not known. In speech recognition, the normal situation is that the transcription is known in the form of a word sequence but the alignment is not known. In the usual case, there is a word pronunciation lexicon and the alignment of the word pronunciations in sequence to the acoustic is straightforward. However, in the case at hand, although acoustic syllable models have been trained automatically, it is not yet known which acoustic syllables correspond to which written characters.

For speech data for which the script is known, the language model takes a special, very simple form. In a graphical representation, the language model is a linear sequence of nodes with one arc leading from each node to the next node. For decades, the standard practice is to use just such a word script as the “supervised” labeling in acoustic model training ([Ba72], [Ba75], [Je74]). Technically, the word script provides only partial supervision. However, when speech is modeled as a hidden Markov process, it is natural to train parameters using the Baum-Welch algorithm (See Appendix A), which inherently allows latent random variables and hence easily accommodates partial supervision. In practice, the technique works so well that the difference between full supervision and this instance of partial supervision is hardly even noticed.

In the integrated task of learning the acoustic syllable models and their correspondence with written characters, there is assumed to be a large amount of untranscribed speech data and some amount of speech data for which the written character transcription is known. The method for the integrated learning process is very simple. Use all the data together. For utterances in which the transcription is known, use a linear network language model. For utterances in which the transcription is not known, use the best available language model as in step (1). The same estimation and update equations are used as in step (1), with one small modification.

In conventional “supervised” acoustic model training not only is the word script known, but also there is a pronunciation lexicon giving the correspondence between words and phoneme sequences. In the current case, the analogous correspondence is precisely what is to be learned. Although on the surface there might appear to be a dilemma, all that is needed is a way to represent the unknown correspondence as part of the hidden stochastic process. Then it can be learned just as part of standard learning procedures.

In the graphical representation, the observed acoustics can be associated either with the nodes or with the arcs. For the current task, it is convenient to use a mixed representation. The written characters will be associated with (a subset of) the nodes; the acoustic syllable models will be associated with arcs. To enable this representation, add node labels to the graph model,  $M = \langle S, A, T(a), L(a), K(s) \rangle$ . The function  $K(s)$  can be represented either as being defined only on a subset of the nodes, or as having a special neutral value for all conventional nodes. For every node  $s$  associated with a written

character  $c$ ,  $K(s) = c$ . Furthermore, the following convention will be followed for all nodes associated with any character value  $c$ . All of these nodes will have the same number of arcs originating at the node and the labels on the arcs will be in one-to-one correspondence. This convention will ensure that the probability distribution of acoustic syllable models associated with each written character will be the same for every instance of the character.

Although trained as transition probabilities in update equation (A.6), the conditional probabilities of the arcs leaving the nodes for a given character  $c$  also represent a (probabilistic) pronunciation lexicon. This lexicon can be combined with the acoustic syllable models and a (character-sequence-based) language model, to form the knowledge sources for a complete large vocabulary continuous speech recognition system in the target language. The recommended development procedure also produces a phonetic recognizer and a syllable recognizer. These tools may be used in various configurations for additional applications such as spoken term detection, speaker identification and language identification. Once trained, the models in this language may also be used as a basis for developing models in additional languages (See Sample Experiment 6.1).

### **Objectives of experiment 8.1**

- 1) Proof-of-concept for technique to automatically learn finite-state grammars.
- 2) Proof-of-concept of likelihood-based statistical validation.
- 3) Proof-of-concept for technique to compare performance of two grammars.
- 4) Proof-of-concept for technique to automatically learn syllable acoustic models without any manually labeled speech data.
- 5) Proof-of-concept for technique to automatically learn syllable lexicon with minimal amount of orthographically transcribed speech data.

## 9.0 Developing a Pronunciation Lexicon from Scratch

Automatically building a word pronunciation lexicon for an alphabetic language presents issues and opportunities that are different from those of automatically developing a syllable-based recognizer. For each word in the vocabulary, a pronunciation lexicon gives one or more pronunciations, each represented as a sequence of *phonemes*. In most speech recognition systems it is not clearly distinguished whether the pronunciations represent sequences of phonemes or sequences of broad phonetic labels. Since the lexicon can only represent the normative pronunciations, not the actual pronunciations of individual instances of a word, the form is clearly *phonemic*. However, context-dependent “phoneme” models also clearly can learn to discriminate among allophones and hence represent at least some amount of *phonetic* knowledge. With supervised training, a distinction can be made based on whether the training data has phonemic labels or phonetic labels. With semi-supervised learning, the distinction becomes less clear. The amount of phonetic knowledge learned depends on the amount of data that has not been manually labeled and on the particular procedures that might be used to automatically label that data. Except as explicitly noted otherwise, in this section the term “phonemes” will refer to the set of symbols in the pronunciation sequences of the lexicon, regardless of the relative amount of phonetic knowledge represented.

### Grapheme-to-Sound Rules: Insertion and Deletion

In an alphabetic language, reading out loud may be represented as a transduction from a sequence of letters or graphemes to a sequence of phonemes. In some words, a particular letter may be silent, corresponding to no phonemes. In other cases, a single letter may correspond to more than one phoneme or a sequence of several letters may correspond to a single phoneme. The relationship between the sequences of graphemes and the corresponding sequences of phonemes may be represented by a *finite state transducer (FST)*. A finite state transducer may be represented as a labeled directed graph in which each arc is associated with two labels sequence. One label sequence is a sequence of zero or more graphemes and the other is a sequence of zero or more phonemes.

As a generative stochastic process, an FST may be thought of as a hidden Markov process generating two streams of symbols, graphemes and phonemes. Graphemes and phonemes are represented symmetrically. Thus, the FST is a bilateral transducer. Either the graphemic sequence or the phonemic sequence may be taken as the known input. Then the probability distribution among sequences in the other symbol set may be computed by inference from the hidden Markov model.

Because there are only a finite number of label sequences, training an FST for grapheme-to-phoneme transduction is easily reduced to a standard case of hidden Markov model training. Let  $\mathcal{L}$  be the set of phoneme sequences that occur on individual arcs of the graph representing the FST. Let  $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$  be any sequence of graphemes. Then it is possible to construct a graph  $G_{\mathcal{A}}$  representing all paths through the FST graph such that the sequence of grapheme labels on the arcs of any path through the graph corresponds to the sequence  $\mathcal{A}$ . Since all the paths in  $G_{\mathcal{A}}$  correspond to the same

grapheme sequence, the grapheme labels can be ignored, so  $G_{\mathcal{A}}$  becomes a standard labeled directed graph with labels in  $\mathcal{L}$ . The transition probabilities in  $G_{\mathcal{A}}$  and the acoustic models for the phonemes may be trained using equations (A.1) through (A.7).

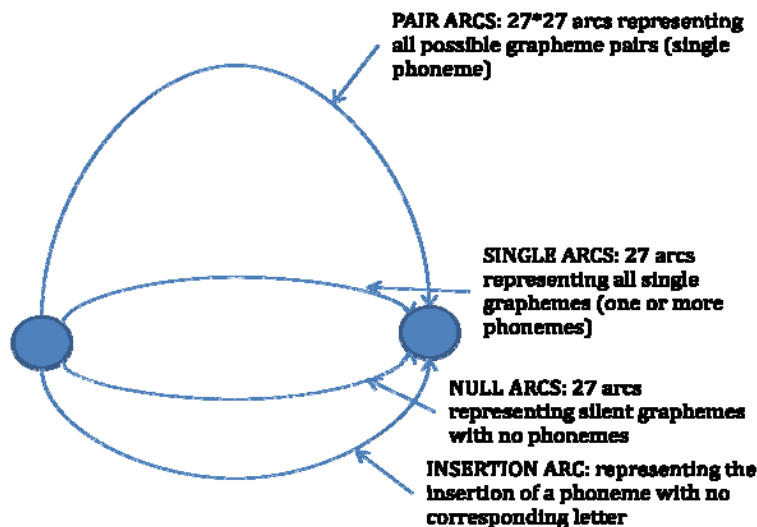


Figure 9.1

Initially in fully automatic training, any grapheme sequence may correspond to any possible phoneme sequence. As in Sample Experiment 8.1, a linear skeleton graph may be used for speech data for which the grapheme sequence is known and a language model skeleton graph may be used for speech data for which the grapheme sequence is not known. Figure 9.1 is a generic graph representing the possible grapheme and phoneme sequences corresponding to a single arc in such a skeleton graph. Figure 9.1 represents a model in which at most two graphemes correspond to a single phoneme. Examples of such grapheme pairs are “sh”, “th” and “ch” in English. A small amount of prior knowledge may be used to limit the number of PAIR arcs to just a handful of letter pairs that are expected to correspond to single phoneme. It is also possible to leave out this prior knowledge, including all possible grapheme pairs in the generic graph and relying on the automatic training to estimate probabilities close to zero for sequence that do not actually occur. The number 27 used in Figure 9.1 is for purposes of illustration only. It corresponds to the 26 letters of the English alphabet plus the space character. For a particular language, the number would be the number of graphemes that occur in that language.

Each arc in Figure 9.1 is further expanded to represent the probability distributions of phonemes. Each SINGLE letter arc that corresponds to more than one phoneme is first expanded into a sequence of arcs and nodes with one arc for each phoneme in the corresponding sequence. Again, prior information may be used to limit the possible phoneme sequences of length greater than one that might correspond to single letters. Examples in English include “n”  $\rightarrow$  /nj/, “x”  $\rightarrow$  /ks/, and “z”  $\rightarrow$  /tz/.

Then each generic phoneme arc is further expanded to represent the distribution across the set of phonemes. As in step (2) of Sample Experiment 8.1, a node is introduced with a node label corresponding to the grapheme sequence associated with the generic arc. From this labeled node, there is an arc for each phoneme. The conditional probability distribution of the phonemes is represented by the transition probabilities out of the labeled node. Whether or not the phoneme probabilities are modeled as being context-dependent is determined by whether or not the node label is shared by the corresponding nodes in different positions in the skeleton graph.

The expanded graph still represents the graphical representation of a hidden Markov process and the models may be trained according to equations (A.1) through (A.7).

If prior information is not used to restrict the number of arcs labeled with grapheme or phoneme sequences of length greater than one, then special techniques may be used to help the system learn that only a small subset of the possible sequences actually occur. These techniques include regularization with an entropic prior [Sm06, Sh07] and an annealing technique.

### **Sample Experiment 9.1: Learning Multiple-Letter Phonemes**

Fully automatic learning of a pronunciation lexicon requires a large scale experiment. A much simpler proof-of-concept experiment may focus on testing the ability of the system to learn which multiple letter sequences correspond to single phonemes. In this proposed experiment the task is to learn just grapheme-to-phoneme rules, not acoustic models for the phonemes. Thus, a language may be used for which a pronunciation lexicon already exists. Preferably the language will be like English, with complex spellings rules with many irregularities and exceptions.

For this experiment, the “manually labeled” data can be taken directly from the lexicon. Simulated unlabeled data may be generated by generating a noisy phonemic transcript for any selected set of text. The question to be studied is how well the system can automatically learn that the letter sequences such as “th”, “sh” and “ch” correspond to single phonemes. In particular, can the system learn from a generic start to assign non-negligible probability only to multi-letter sequences that actually correspond to single phonemes?

Optionally, this experiment can also provide a proof-of-concept test of statistical validation. Use statistical validation to test for over fitting of complex generic models to the training data. Compare the performance of a more complex model (with more free parameters) to the performance of a less complex model. The null hypothesis is that the two systems have the same average performance as measured by some test statistic. For statistical validation using data that has not been manually labeled, some knowledge must be set aside for use in the automatic labeling of the validation data.

In this case, the set aside knowledge is the knowledge of the pronunciation lexicon. The automatic labeling of the validation data may be done by a hybrid phoneme-word

recognition system. Such a hybrid system can utilize a moderate sized pronunciation lexicon, representing only a fraction of the vocabulary actually encountered in the data. The recognition system should be designed to be balanced in its comparison of the two systems being compared. One way to create a balanced comparison system is to have two versions of the hybrid system, one based on each of the model systems. The automatic labeling of the validation data would random choose which version to use for the automatic labeling of each utterance, with the two versions being equally likely.

Another way that an unbiased system might be created is to build an integrated system that incorporates both versions into a single, fused system. This method may be preferred if the two versions have distinct strengths and weaknesses, so that a fused system is likely to outperform either single version. However, with this method there is a danger that the fused system may favor one of the component system more than the other, so an independent test should be conducted to verify that the design is neutral relative to the systems being compared.

The statistical validation test requires that the validation data be recognized three different ways, once using each of the two systems being compared and once using the hybrid system with the set aside knowledge. Because the knowledge of the pronunciation lexicon is set aside, the recognition with each of the systems being compared uses some form of generic language model. For example, the generic language model may be a phoneme n-gram model or may be based on a syllable grammar. The automatic labeling of the validation data, on the other hand, uses the pronunciation lexicon for within vocabulary words, perhaps with the help of a word-based language model.

The test statistic is the accumulated count of the net number of times that one version of the base system agrees with the automatic labeling and the other one disagrees. Under the null hypothesis, each system is equally likely to agree with the automatic labeling. If the more complex model set successfully learns additional knowledge, then it will perform better and agree with the automatic labeling more often. On the other hand, if the more complex system has been trained to over fit the training data, then its performance on independent validation data will be not as high as the performance of the less complex system.

## Annealing and Statistical Validation of Structural Learning

The generic acoustic model illustrated in Figure 9.1 represents hundreds of possible phoneme sequences for each grapheme. In reality there will only be a fairly small number of phoneme sequences that are associated with any given grapheme. The automatic training needs to learn that almost all of the probabilities are zero and that most of the phoneme sequences never occur. However, except for beam pruning, the forward-backward computation (equations A.1 and A.2) never gives a zero count to any possibility that has a non-zero prior. On the flip side anything that has a zero probability in the current model will always get zero counts and will never train to be non-zero. Thus, setting a probability to zero or making a zero probability non-zero is a structural change. Such structural learning should not be left up to the vagaries of a side effect of

the beam pruning. Instead, making a particular probability in a model zero should be an explicit decision. When the number of non-zero counts for a particular model parameter gets to be very small, making the count zero in any particular utterance due to beam pruning should also be an explicit decision. If the decision is not to make the probability zero, the beam size may need to be adjusted to make sure that the event corresponding to the fractional count is not pruned.

The explicit decisions to set certain probabilities to zero can be made by statistical validation, which will guarantee that each decision is based on a statistically significant amount of evidence. The same property that guarantees that any possibility with a non-zero prior will actually get a non-zero count also causes the convergence of probabilities to zero to be slow. Based on the statistical significance tests, small conditional probabilities can be more aggressively be structurally set to zero, producing faster convergence to a sparse set of non-zero conditional probabilities.

To avoid getting stuck in a local optimum in the structural learning, there should also be a mechanism for periodically retesting every zero probability in the model to see if the probability should be made non-zero.

The fact that the statistical significance test will give false positive results (with a known bound on the probability of such false positives) can be a useful property. The non-zero probability of a false positive is like annealing with a non-zero temperature [Ki83]. The rate of annealing can be controlled by gradually increasing the level of statistical significance required to make a structural change.

## Appendix A: Training Acoustic Models

This appendix presents the standard equations for acoustic model training. These equations are a version of the Baum-Welch algorithm, which has been used for supervised acoustic model training of a hidden Markov process for over thirty years [Ba72, Je74, Ba75]. The Baum-Welch algorithm is an instance of the EM algorithm [De77], which may be used in general to estimate the parameters of a stochastic process in which some of the random variables are “hidden.” These equations can also be used for unsupervised learning and as a tool within the semi-automatic learning methods proposed in this paper. That is, in some of the semi-automatic learning procedure there is a step within the procedure in which conventional supervised training or conventional unsupervised training is used.

This appendix presents only the standard equations, not the specific extra computations used in any one of the new semi-automatic procedures. This appendix does not use the standard notation and terminology for the Baum-Welch algorithm for training a hidden Markov model. Rather it uses a slightly more general representation that is mathematically equivalent, so all the standard theorems apply. The representation is of a labeled directed graph. The (acoustic) observations may be associated either with the nodes or with the arcs of this graph. In various situations, sometimes one representation is more convenient than the other. When the probability distributions for the observations are associated with the (labeled) arcs, several arcs with different labels may all go from the same source node to the same destination node. This representation allows a direct representation of a situation than is common in acoustic modeling networks. It is less convenient to represent the same model in a standard hidden Markov model because there is only one observation distribution associated with any one ordered pair of nodes, that is a paired source node and destination node. The generalization is mainly a matter of convenience because the same model could be represented in a conventional model with more states.

The labeled directed graphs are also convenient for representing the compound graphs and the process of building compound graphs as presented in Sample Experiment 8.1. The discussion of Sample 8.1 is the reason for presenting the notation and terminology used in this appendix, so the discussion below will make some specific references to the particular example of Sample Experiment 8.1. Because the labeled, directed graph representation is more general, these equations may also be used to implement a standard hidden Markov model Baum-Welch training algorithm.

### Training the Acoustic Models

Whatever strategy is used for choosing the initial graphs, assume that a graphical model for acoustic syllables is available.

Let  $G = \langle S, A, T(a), L(a) \rangle$  be the composite graphical model for all the syllables in the language.

Let  $\{Y_t\}$  be a sequence of acoustic observations. In Sample Experiment 8.1, “Learning a Syllable Grammar,” it is the output sequence of the phonetic or phonemic recognizer for some block of speech data. To train the models on the data  $\{Y_t\}$  it is necessary to know what state the hidden Markov process is in at each point in time and which arcs it follows.

The probability computation can be split into a forward part (A.1) and a backward part (A.2). Together, these two equations are sometimes called “the forward-backward computation.” This is the inner-loop computation for hidden Markov process training.

Let  $\alpha(j,t) = \text{Prob}_m(S_t=j, Y_1, Y_2, \dots, Y_t)$ . Then  $\alpha(j,t)$  may be computed from  $\alpha(*,t-1)$  using a dynamic programming algorithm.

$$(A.1) \alpha_m(j,t) = \sum_{a:i \rightarrow j} \alpha_m(i,t-1) T_m(a) \text{Prob}_m(Y_t|L(a)),$$

where the sum is over all  $i$  and all arcs that go from state  $i$  to state  $j$ .  $\alpha(*,0)$  is initialized to the a priori distribution of the states. In this case, that will be a uniform distribution across all the states that could represent the beginning of a syllable. In some graphs there will only be one such state. The probabilities  $T_m(a) \text{Prob}_m(Y_t|L(a))$  in the equations have the subscript  $m$  because these probabilities are not the true probabilities but only the estimates of these probabilities based on the current models.

Let  $\beta(k,t) = \text{Prob}_m(Y_{t+1}, Y_{t+2}, \dots, Y_T | S_t=k)$ . Then  $\beta(k,t)$  may be computed from  $\beta(*,t+1)$  using a dynamic programming algorithm.

$$(A.2) \beta_m(k,t) = \sum_{a:j \rightarrow k} \beta_m(j,t+1) T_m(a) \text{Prob}_m(Y_{t+1}|L(a)),$$

where the sum is over all  $j$  and all arcs that go from state  $j$  to state  $k$ .  $\beta(k,t)$  is computed backwards in time starting at the end of the  $Y$  sequence. Notice that  $\beta(k,t)$  is defined as a probability conditional on the hidden state rather than as a joint probability.  $\beta(*,T)$  is initialized as a conditional probability with no observations yet made. That is,  $\beta(*,T)$  is set to 1.0 for every state that could represent the end of a syllable.

The probability of making all the  $Y$  observations and of being in state  $i$  at time  $t$  is given by

$$(A.3) \gamma_m(i,t) = \alpha_m(i,t) \beta_m(i,t) = \text{Prob}_m(Y_1, Y_2, \dots, Y_T, S_t = i)$$

Let

$$(A.4) P_m = \sum_i \gamma_m(i,t) = \text{Prob}_m(Y_1, Y_2, \dots, Y_T).$$

Notice that the sum is the same for all  $t$ .  $P_m$  is the probability of the current model generating the observed data. The maximum-likelihood criterion is to choose the model that maximizes  $P_m$ .

The probability of following arc  $a:i \rightarrow j$  from time  $t$  to time  $t+1$  and making all the  $Y$  observations is

$$(A.5) \delta_m(a, t) = \alpha(t, t) T_m(a) \text{Prob}_m(Y_{t+1} | L(a)) \beta(j, t + 1) / P_m.$$

Let  $B_{j,k,m} = \text{Prob}_m(Y_t = k | L(a) = j)$ . The conditional independence property of a hidden Markov process assures that the expression on the right hand side of this definition is the same for all  $t$  and for all arcs  $a$  such that  $L(a) = j$ . That is, the value of  $B$  depends only on  $j, k$  and  $m$ .

The following equations, the Baum-Welch update equations, show how to update the models  $T_m(a)$  and  $B_{j,k,m}$  so as to increase the likelihood  $P_m$ .

$$(A.6) T_{m+1}(a) = \sum_t (\delta_m(a, t) / \sum_{c \text{ such that } L(c)=L(a)} \delta_m(c, t))$$

$$(A.7) B_{i,j,m+1} = \frac{\sum_{t \text{ such that } Y_t = j} \sum_{a \text{ such that } L(a) = i} \delta_m(a, t)}{\sum_t \sum_{a \text{ such that } L(a) = i} \delta_m(a, t)}$$

There is a theorem [Bau72], [De77] that shows that application of equations (A.1) through (A.7) produces new models such that  $P_{m+1} > P_m$  unless the current models are at a stationary point in the likelihood function in which case  $P_{m+1} = P_m$ . Therefore, repeated iterations of equations (A.1) through (A.7) will yield a sequence of models that converge in the sense that the likelihoods of the models converge to the value of a stationary point. From a random starting point, the sequence of models will converge to a local maximum in the likelihood function. This repeated iteration of equations (1) through (7) is an instance of the Expectation and Maximization (EM) algorithm. In the case of estimation of a hidden Markov process, the algorithm is also known as the Baum-Welch algorithm.

Notice that the application of the EM algorithm in Sample Experiment 8.1 does not need any manually labeled speech data from language  $L$ . The Baum-Welch algorithm has been used in speech recognition [Ba75] since even before the EM algorithm was named. However, the Baum-Welch or EM algorithm is usually used assuming that the transcription is known. That is, it is essentially a supervised training algorithm and requires manual labeling (that is, transcription or the reading of a known script). In Sample Experiment 8.1, only the labels generated automatically by the phonemic or phonetic recognizer are needed.

## New Techniques and Concepts Introduced in This Paper

- 1) Statistical validation: Validation using data that has not been manually labeled.
- 2) Concept: Setting aside a source of knowledge rather than setting aside labeled data.
- 3) Multi-level statistical validation: Making repeated use of statistical hypothesis testing safer.
- 4) Learning-by-imitation (advantages over self-training):
  - a. Supports objective functions other than maximum likelihood.
  - b. Supports discriminative training.
  - c. Supports automatically optimizing price-performance.
  - d. Supports the use of supervised learning algorithms on data that has not been manually labeled.
  - e. Supports structural learning.
- 5) Concept: Use of a reference system to supervise learning by a limited system.
- 6) Technique: Modeling a mixture of two languages in order to bootstrap learning in the second language from the first.
- 7) Technique: Clustering pseudo-phonemes to reduce differences between words to artificially create selected minimal-pair distinctions.
- 8) Technique: Creating multiple limited systems to enable a fused system to exceed the performance of a reference system.
- 9) Likelihood-based statistical validation: Statistical validation without requiring either manually labeled data or a set aside knowledge source.
- 10) Technique: Hybrid system of phoneme, syllable and word recognition.
- 11) Technique: Scalable vocabularies, including making vocabulary smaller as a research tool in designing experiments for a particular purpose.
- 12) Technique: Fast match as a flexible source of limited system designs to support learning-by-imitation experiments.
- 13) Concept: Less accurate, but faster phoneme recognition in a hybrid system to improve price-performance.
- 14) Technique: Mixed node-arc representation for integrated learning of acoustic models and grapheme-to-sound correspondence.
- 15) Technique: Corrective-training-based hill climbing to optimize the parameters in the components of a multiple classifier system as well as the parameters in the fusion function.
- 16) Technique: (Socratic controller) Delayed-decision testing for assignment of responsibility in joint training of multiple classifiers.
- 17) Technique: (Socratic controller) Training for diversity: delayed-decision in the evaluation of the benefit of diversity.



## References

- [Ba72] J Baker, “Machine-Aided Labeling of Connected Speech,” in *Working Papers in Speech Recognition – II*, Computer Science Department, Carnegie-Mellon University, 1972.
- [Ba74] J Baker, “The DRAGON System – an Overview,” Proc. IEEE Symposium on Speech Recognition, Pittsburgh, PA, 1974, pp. 22-26.
- [Ba75] J Baker, “Stochastic Modeling for Automatic Speech Recognition,” in *Speech Recognition*, edited by D. Raj Reddy, Academic Press, 1975.
- [Ba85] J Baker, P Bamberg, M Sidell, R Roth, “Speech Recognition Apparatus and Method,” US Patent 4783803, filed 1985.
- [Ba05] J Baker, “Distributed Pattern Recognition Training Method and System,” US Patent Application 11/180,600, filed July 2005.
- [Ba06] J Baker, "Spoken Digital Libraries: The Million Hour Speech Project," invited paper presented at the 2nd Annual International Conference on the Universal Digital Library, Alexandria, Egypt, November, 2006.
- [Ba07] J Baker, “Robust Pattern Recognition System and Method Using Socratic Agents,” US Patent Application 11/898,636, filed Sept. 2007.
- [Ba08] D Bansal, N Nair, R Singh, B Raj, “A joint decoding algorithm for multiple-example-based addition of words to a pronunciation lexicon,” [http://web.jhu.edu/HLTCOE/Publications/Singhfromcoe\\_paper1.pdf](http://web.jhu.edu/HLTCOE/Publications/Singhfromcoe_paper1.pdf).
- [Bau72] L Baum, “An inequality and associated maximization technique is statistical estimation of probabilistic functions of a Markov process,” in *Inequalities*, 3, pp. 1-8, 1972.
- [Bu06] C. Bucila, R Caruann, A. Niculescu-Mizil, “Model Compression,” International Conference on Knowledge Discovery and Data Mining (KDD), 2006.
- [Ca80] R. L. Cave and L. P. Neuwirth, “Hidden Markov Models for English,” In *Hidden Markov Models for Speech*, J. D. Ferguson (editor), IDA — CRD, pages 8–15, October 1980.
- [Cr96] M Craven, “Extracting comprehensible models from trained neural networks,” Doctoral Dissertation, University of Wisconsin at Madison, 1996.
- [De77] A. P. Dempster, N. M. Laird and D. B. Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of the Royal Statistical Society*, volume 39, number 1, pages 1–38, 1977.

[Ga01] Ganapathiraju, J. Hamaker, M. Ordowski, G. Doddington and J. Picone, "Syllable-Based Large Vocabulary Continuous Speech Recognition", *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 4, pp. 358-366, May 2001.

[Hu01] X Huang, A Acero, H Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice Hall, 2001.

[Ja91] R Jacobs, M Jordan, S Nowlan, G Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, 1991, MIT Press.

[Je74] F Jelinek, L Bahl, R Mercer, "Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech," Proc. IEEE Symposium on Speech Recognition, Pittsburgh, PA, 1974, pp. 255-260.

[Je97] F Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, 1997.

[Jo94] M Jordan, R Jacobs, "Hierarchical Mixtures of Experts and the EM algorithm," *Neural Computation*, 1994, MIT Press.

[Ka05] D Karakos, S Khudanpur, J Eisner, CE Priebe, [Unsupervised Classification via Decision Trees: An Information-Theoretic Perspective](#), Acoustics, Speech, and Signal Processing, Proceedings, 2005.

[Ka07] D Karakos, S Khudanpur, J Eisner, CE Priebe, [Iterative denoising using Jensen-Renyi divergences with an application to unsupervised document ...](#), Proc. 2007 International Conference on Acoustics, Speech and Language Processing, 2007

[Ki83] S Kirkpatrick, CD Gelatt, MP Vecchi, "Optimization by simulated annealing," *Science*, 1983.

[Li08] P Liang, H Daume, D Klein, "Structure Compilation: Trading Structure for Features," International Conference on Machine Learning, Helsinki, Finland, 2008.

[Ma08] Jeff Ma and Richard Schwartz, "Unsupervised versus supervised training of acoustic models," in *INTERSPEECH 2008*, Brisbane, Australia, 2008.

[No09] Scott Novotney, Richard Schwartz, and Jeff Ma, "Unsupervised Acoustic and language model training with small amounts of labeled data," <http://web.jhu.edu/HLTCOE/Publications/ICASSPnovotney.pdf> (submitted for publication).

[Ro51] Robbins, H. and Munro, S. "A Stochastic Approximation Method." *Ann. Math. Stat.* 22, 400-407, 1951.

[Sh07] M. Shashanka, B. Raj, and P. Smaragdis, “Sparse overcomplete latent variable decomposition of counts data,” in *Proceedings of the 21th Annual Conference on Neural Information Processing Systems (NIPS '07)*, Vancouver, BC, Canada, December 2007.

[Sm07] P. Smaragdis, B. Raj, and M. Shashanka, “Supervised and semi-supervised separation of sounds from single-channel mixtures,” in *Proceedings of the 7th International Conference on Independent Component Analysis and Blind Signal Separation (ICA '07)*, pp. 414–421, London, UK, September 2007.

[Sm09] P. Smaragdis and B. Raj, “Shift-invariant probabilistic latent component analysis,” to appear in *Journal of Machine Learning Research*.

[Sm06] P. Smaragdis, B. Raj, and M. Shashanka, “A probabilistic latent variable model for acoustic modeling,” in *Proceedings of the Advances in Models for Acoustic Processing Workshop (NIPS '06)*, Whistler, BC, Canada, December 2006.

[Wa96] S Waterhouse, D. MacKay, T Robinson, “Bayesian Methods for Mixtures of Experts,” *Advances in Neural Information Processing Systems*, 1996.

[Wh08] C. White, G. Zweig, L. Burget, P. Schwarz, and H. Hermansky, “Confidence estimation, OOV detection, and language id using phone-to-word transduction and phone-level alignments,” in Proc. ICASSP, 2008.

[Wh09] C White, A Sethy, B Ramabhadran, P Wolfe, E Cooper, M Saraclar, and J Baker, “Unsupervised Pronunciation Validation,” [http://web.jhu.edu/HLTCOE/Publications/cmw\\_unsup\\_icassp09\\_fin.pdf](http://web.jhu.edu/HLTCOE/Publications/cmw_unsup_icassp09_fin.pdf), ICASSP 2009.