BOSTON UNIVERSITY
COLLEGE OF ENGINEERING

Thesis

LANGUAGE MODELING WITH SENTENCE-LEVEL MIXTURES

by
RUKMINI IYER

B.E.(Hons) Electrical Engineering,
University of Bombay, Bombay, India, 1992.

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science
1994

Acknowledgments

I would like to first acknowledge my advisor, Professor Mari Ostendorf. This thesis would not have been possible without her constant guidance and encouragement. The last year and a half with her have been a very productive period in my research. I would like to acknowledge my readers, Dr. Lev Levitin and Dr. Robin Rohlicek, whose insightful suggestions and ideas have made this a much better thesis.

I would like to thank Cheryl Kelly, Linda Hession and Phyllis Doheny who made the paperwork and other formalities a great deal more easier than it could have been.

It has been a great pleasure to work with my other colleagues in SPILAB. I would like to thank Ashvin and Owen for introducing me to a more organized coding style, Fred and Sanjay for helping me find the bugs which did and did not exist, and all my labmates for patiently hearing out my "I-am-burnt-out" cribs after a crazy day of hacking. I would also like to thank them for helping me improve on my presentations, especially Orith, Fred and Sanjay who have borne the brunt of my "practice-sessions". I am looking forward to a couple of more years with them. I wouldn't like to leave out the SPILAB machines, especially lark and oriole, who have been my long standing companions for the last year and a half.

I would like to acknowledge my friends Sanjai Singh, Mani Ramamurthy and Abha Chandra for a fun-filled two years in this country. Thanks Sanjai, for the million times I have needed your help, and all the happy times that we have shared together.

To Subbu - I owe a lot of my sanity and happiness. Thanks for all the confidence in me when my self-esteem is at an all time low! Finally, I would like to acknowledge my family without whose help, support and encouragement, I wouldn't be where I am.

This research was supported by ONR under contract number N00014-92-J-1778.

LANGUAGE MODELING WITH

SENTENCE-LEVEL MIXTURES

(Order No.)

RUKMINI IYER

Boston University, College of Engineering, 1994

Major Professor: Mari Ostendorf,
Associate Professor of Electrical, Computer
and Systems Engineering

Abstract

Language models play an important role in improving the accuracy of a continuous speech recognizer. In this thesis, we introduce a new statistical language model which captures long term topic dependencies of words within and across sentences. The model includes two main contributions. First, we develop a topic-dependent sentence-level mixture language model which takes advantage of the topic constraints in a sentence or a paragraph. Since this language model is not Markov and has a large search space, it is used only in the last stage of a multi-pass search strategy in the recognizer. Second, we introduce topic-dependent dynamic adaptation techniques in the framework of the mixture model. During the course of this thesis, we also investigate robust parameter estimation techniques, which are extremely important in light of the sparse data problems in language modeling. The model is implemented in the BU speech recognition system and provides a significant improvement in recognition accuracy. An important advantage of the framework of our model is that it is a simple extension of existing language modeling techniques that can easily be integrated with other language modeling advances.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The aim of automatic speech recognition is to accurately transcribe natural speech into text. The speech recognizer serves as a human-machine interface, for example, as in a spontaneous dictation transcriber. More futuristic applications would see the speech recognizer as a very significant player in the information highway. However, to achieve these goals, there are several problems to overcome, ranging from speech and speaker variability to domain-independence and real-time response. There are ongoing efforts to deal with these problems in almost all areas connected to speech, including prosody, acoustic and language modeling. In all these areas, statistical modeling approaches are popular, in part because many of the successful recognition systems today incorporate a statistical approach. For example, statistical language models, the model of the probability of different word sequences, has in the past few years become an integral part of the state-of-the-art speech recognizers. In this thesis, we introduce a new statistical language model which provides a significant improvement in recognition accuracy when implemented in the framework of the Boston University speech recognition system.

The overall performance of a large vocabulary continuous speech recognizer is

greatly impacted by the additional information provided by a stochastic language model. The need for a language model in a recognizer can be immediately seen with this simple example. Suppose the input to the speech recognizer is the utterance:

"Go to that room."

The possible output from the acoustic segment of the recognizer can be:

"Go to that room."
"Go too that room."
"Go two that room."

The acoustic model will not be able to differentiate between "to", "too" and "two". Hence we need some kind of a grammar which will be able to confidently pick up the first sentence as the recognized output. One might feel that a simple syntactic grammar should be able to do this job. However, though syntax has been formalized by linguists, we are still very far from a **complete** grammar for English. The current grammars would prove to be too rigid for different applications of the recognition system.

Another problem is that humans do not necessarily speak grammatically. They may use awkward or disfluent phrasing, or abbreviated structures that depend on the context of the dialog and assume a corresponding knowledge from the "listener", where the "listener" could be a computer. For example, the Air Travel Information Systems (ATIS) application basically consists of customer queries or requests about flights and fares and the corresponding computer responses. A simple customer request at some point in a dialog with the computer may be

"Round trip to Denver, please."

One would not characterize this sentence as grammatical in the sense that it is not a complete clause with a noun-phrase and a verb-phrase. In some applications, it might be appropriate to overlook the correctness of the grammar of the sentence and simply extract the information requested.

To deal with these problems and requirements, a statistical language model was suggested [1]. This statistical language model and its different variations proved to be very important components of the recognizer. These models, which consider a history of the previous $n-1$ words to estimate the probability of the nth word, are referred to as n-gram language models. Typically, due to memory and computation requirements, the value of n is restricted to 2 or 3, a bigram or trigram language model respectively. The models are therefore constrained by their inability to take advantage of long distance dependencies (greater than n) in a sentence or paragraph. There are several types of long distance constraints some of which are illustrated below.

Grammatical constraints: Grammatical constraints could be simple verb-tense agreement or singular-plural agreement in a sentence.

"Once the flood waters ran low, the people built new homes."

is a simple example of verb tense agreement. The distinction between "built" and "build" will have to be given by the acoustic model since it is beyond the capability of the current standard trigram language models.

Topic-related dependencies: To illustrate this dependency, let us examine an error made by our recognition system when recognizing an utterance using a standard ARPA trigram model:

"The first recipient Joseph Webster Junior a Phi Beta Kappa chemistry

> grad who plans to take courses this fall in **art religion** music and political science."

The output of the recognizer was:

> "The first recipient Joseph Webster Junior he friday a cap of chemistry grant who plans to take courses this fall in **aren't really chin** music and political science[1]."

In the context of "courses" and "student", "art religion" is much more relevant than "aren't really chin". The current standard trigram model is unable to take advantage of the long distance topic-related dependencies.

Speaker goal or style related constraints: These kind of dependencies are observed in cases where there are human-computer interactions rather than a simple dictation. There are differences in language between queries for information and responses or information clarification. This is clearly reflected in case of the ATIS corpus. The corpus is based on customer queries and the answers to the queries. Queries can be of the kind:

> "Show me the flights from Boston to Denver."
> "What does fare code Y mean?"

Responses or clarifications can be of the kind:

> "Wednesday between three and four PM."
> "Round trip please."

There have been other efforts in stochastic language modeling to deal with the problem of long distance constraints, and a few of these approaches will be outlined

[1]Underlined words were not recognized since they were out of vocabulary.

in Chapter 2. An important advance in this direction is dynamic adaptation of the language model to the partially observed document. For example, consider a small clip from a news article regarding taxes,

> "The real danger to the Texas economy now is that the tax increases soon will be attacked as inadequate. Politicians will then call for creation of a state personal or corporate income tax. Ominously, a panel already has been appointed to make recommendations next year on changes in the state's tax structure."

It can be seen that the word "tax" is prominent in usage throughout this clip. Dynamically adapting the language model to the partial document will automatically raise the frequency of the word "tax". This can be considered equivalent to tracking the topic in the article to take advantage of the short-term fluctuations in word frequencies. Existing dynamic language models are useful for tracking dependencies within an article but less so for capturing dependencies within a *sentence*.

This thesis introduces a sentence-level mixture model which makes use of existing techniques in statistical language modeling and at the same time extends the capability of the standard models to take advantage of long distance topic-related constraints within sentences. We also incorporate dynamic language modeling techniques in the framework of our mixture model to adapt to fluctuations in word frequencies and to capture dependencies across sentences.

The rest of the thesis is organized as follows. In Chapter 2, previous work in language modeling for speech recognition (with emphasis on statistical language models) is discussed. We follow this with a description of the speech recognition paradigm and the language model training corpus in Chapter 3. Chapter 4 describes the baseline mixture model and the experimental results obtained in the course of the initial work in this thesis. Chapter 5 provides a detailed description of the dynamic

adaptation of the mixture model to a partially observed document. Finally, Chapter 6 outlines the significance of this thesis and the important conclusions drawn from this work.

Chapter 2

Background

Statistical techniques used in speech recognition, as introduced in [2], try to model speech as an information source (the word sequence) transmitted via a noisy channel (the vocal tract and signal processor) as shown in Figure 2.1.

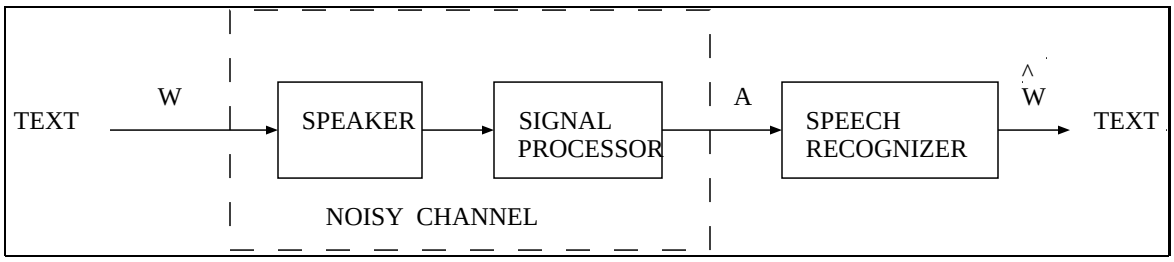

Figure 2.1: *The speech recognition problem in terms of communication theory.*

In mathematical terms, let

$$W = w_1, w_2, \ldots, w_T \tag{2.1}$$

denote a sequence of T words. Let the acoustic signal be represented as A. The goal of speech recognition is to find a word string $\hat{W}$ which maximizes the probability $P(W|A)$,

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(W|A). \tag{2.2}$$

Using Bayes formula, the recognition problem can be formulated as

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(A|W)P(W), \tag{2.3}$$

where $P(A|W)$ is given by the acoustic (or the channel) model and the a priori probability of the word string P(W) is given by the language (or source) model. The attempt to maximize probability $P(W|A)$ efficiently depends on the search algorithms used and the dependence assumptions in the model.

Different approaches to language modeling have evolved over the years and a few of these approaches are outlined below in Section 2.1. A quantitative evaluation of the significance of the statistical language model is described in the Section 2.2. Since our mixture model is based on statistical techniques in language modeling, the different problems encountered in statistical modeling and some of the solutions that have been developed to specifically deal with these problems in language modeling are discussed in Section 2.3. Section 2.4 outlines some of the previous work in the area of dynamic language modeling or language model adaptation, which is an important component of our model. Our mixture model aims to take advantage of long distance topic-related dependencies within and across sentences, as well as short term fluctuations in word frequencies. There have been other efforts to model long distance dependencies and this work is outlined in Section 2.5. Finally, in Section 2.6, we summarize the main issues that are relevant to this thesis.

2.1 Approaches to Language Modeling

The syntactic approach to language modeling originated before the statistical techniques did, but it was mainly used for natural language understanding. The more recent statistical language models have been combined very successfully with acoustic models to build accurate recognizers. In this section, we describe both syntactic and

statistical approaches and the attempts to combine the two approaches.

Syntactic Language Models

The language models held by humans have far more information than simple statistical models. The human model includes knowledge of syntax and knowledge of the discourse, as well as the semantics involved. Of these, syntax is the most theoretically formalized by linguists. Syntactic rules have been developed over the years and have been used in speech understanding systems.

An example of a *rule* is, a sentence S can be parsed as a noun phrase (NP) and a verb phrase (VP), namely, $S \rightarrow NP \;\; VP$ [3]. Typically, the technique is to develop a syntactic parser which may parse a sentence as follows. For example,

Dr. Chang died and Dr. Lee was appointed chairman.

may have a parse tree as illustrated in Figure 2.2. The problem with incorporating such syntactic parsers in a recognizer is that an ungrammatical sentence or unanticipated sentence will be assigned a probability of zero. However, people do speak "ungrammatical" utterances, which when combined with some knowledge of the context and semantics, make perfect sense. Moreover, since there is no **complete** formal grammar for English, there will be many reasonable sentences that are not anticipated by the grammar. Syntax is however a powerful tool and can be successfully incorporated in statistical models as explained later.

Statistical Language Models

The most commonly used statistical language modeling technique is to consider the word sequence to be a "Markov" process. A Markov process is one that satisfies

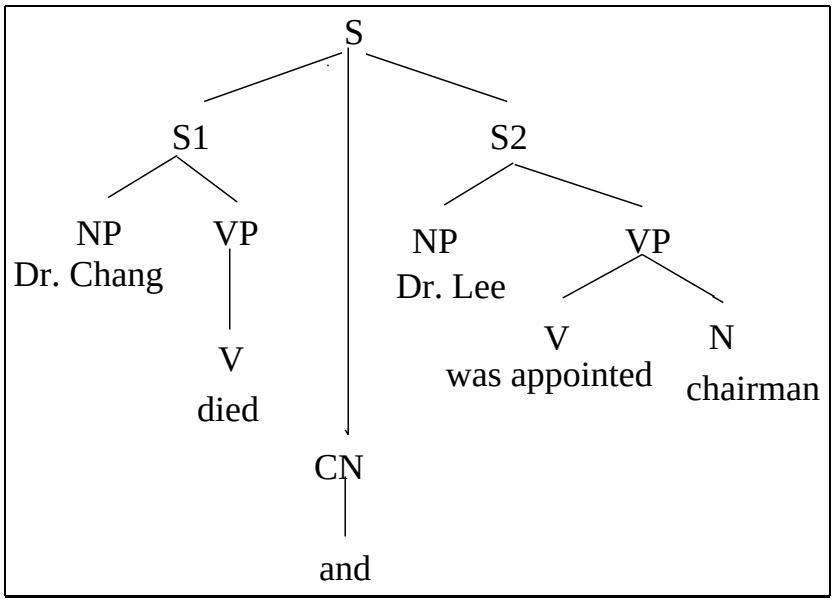

Figure 2.2: *Example of a parse tree: S, S1, S2 sentence, NP noun phrase, V verb, CN conjunction, N noun, VP verb phrase.*

the Markov property, that is for $t_n > t_{n-1} > t_{n-2} \ldots > t_1$,

$$P(x(t_n)|x(t_{n-1}), x(t_{n-2}), \ldots, x(t_1)) = P(x(t_n)|x(t_{n-1})).$$

In case of a word sequence $w_1, w_2 \ldots w_T$, which is discrete-time and discrete-valued,

$$P(w_n|w_{n-1}, w_{n-2}, \ldots, w_1) = P(w_n|w_{n-1});$$

therefore,

$$P(w_1, w_2, \ldots, w_T) = P(w_1) \prod_{i=2}^{i=T} P(w_i|w_{i-1}). \tag{2.4}$$

The above equation represents a **bigram** language model. The idea of the word sequence as a Markov process can be extended to consider a history of more than one previous word as for example, in a **trigram** language model, where

$$P(w_n|w_{n-1}, w_{n-2}, \ldots, w_1) = P(w_n|w_{n-1}, w_{n-2}).$$

Strictly speaking, this is not a Markov process and hence it is often referred to as a n-**gram** language model. In other words, in case of the n-gram models, the idea is to predict the occurrence of a word given a history of the previous $n-1$ words.

The greater the history considered, i.e. the greater the value of n, the better is the predictive power of the n-gram language model. The n-gram model, which is generally trained on a large training text, produces reasonable non-zero probabilities for all the words in the vocabulary considered for speech recognition, using smoothing techniques. The greater the amount of training data, the better the model. However, as the value of n and the training data increases, the model exceeds the virtual memory of computers in use and the probabilities become difficult to access. For example, it was observed that for a particular 50 million words of text data from a Wall Street Journal corpus and a 5000 word vocabulary, the number of unique 2-grams were 1.5 million, the number of 3-grams were 8 million and the number of 4-grams were 12 million [4]. Therefore to date, the value of n has been typically restricted to three for large vocabulary tasks.

Combined Language Models

The simplest combined model is the case where we have syntactic classes for each word. The probability of a sentence in a bigram model is then computed as

$$P(w_1, w_2, \ldots w_T) = \sum_C P(c_1)P(w_1|c_1) \prod_{i=2}^{T} P(w_i|c_i)P(c_i|c_{i-1}) \tag{2.5}$$

where c_i represents a syntactic class and C is a T-length sequence of classes. The model can be considered as a Hidden Markov Model (HMM) [2]. In the case of a Hidden Markov Model, the observation is a probabilistic function of a sequence of states where the sequence is a Markov process. It is said to be hidden because the underlying Markov process (C) is not observable and can only be observed through

another stochastic process (W). In the case of the combined model referred to in Equation 2.5, the observations are the words and the hidden states are the syntactic classes. The probability $P(w_1, w_2, \ldots, w_T)$ can be computed using an iterative procedure known as the forward algorithm [5].

In the previous subsection, we considered a simple example of parsing a sentence using a context-free grammar. Here the rules by which a word is tagged by its syntactic class is deterministic and usually the rules are derived by hand. More recently, we have seen stochastic context-free grammars where the rules have probabilities associated with them. These rules are either derived by hand or estimated using complicated algorithms [6, 7, 8, 9, 10, 11].

2.2 Goodness of a Language Model

We have different statistical language models and need to compare these different models. This brings us to the question: **How can we determine whether one language model is better than the other?**

The measure of the goodness of a language model is based on concepts from Information Theory and is referred to as **perplexity**. Perplexity is defined as,

$$P = B^{H(w)} \tag{2.6}$$

where $H(w)$ is the entropy of a word sequence w, representative of the language, and B represents the base of the logarithm (typically 2) in the definition of entropy.

Entropy: If X is a random variable which takes L values $x_1, \ldots, x_L$ with probability $P(x_1), \ldots, P(x_L)$ respectively, the entropy of the random variable is defined as (using $B = 2$)

$$H(X) = -\sum_{i=1}^{i=L} P(x_i) \log_2 P(x_i). \tag{2.7}$$

Suppose X can take two values x_1 and x_2 and each value is equally probable, the entropy or the amount of information will equal 1 bit. One bit will be sufficient to characterize this information, either a 0 or a 1. The uncertainty of predicting the occurrence of either value will be maximum. However, if one of the values is more probable than the other, entropy will be lower and the number of bits required to represent this information will be reduced. The occurrence of either value can be predicted with greater certainty. This means that random variable X_1 with L equiprobable values will have a greater entropy or correspondingly a greater uncertainty than a random variable X_2 with L values occurring with unequal probabilities, and possibly a greater entropy than a random variable with $L' > L$ values of unequal probability.

Extending Equation 2.7, if instead of considering a random variable, we consider an ergodic random process which generates a sequence of symbols $x_1, x_2, \ldots, x_T$, the entropy of the random process can be defined as

$$H(X) = -\lim_{T \longrightarrow \infty} \frac{1}{T} \sum_{i=1}^{i=L} P(x_1, x_2, \ldots, x_T) \log_2 P(x_1, x_2, \ldots, x_T). \tag{2.8}$$

If sequences $x_1, x_2, \ldots, x_T$ are typical of the process, a single long sequence should be able to convey the same information as a set of such sequences (by the property of ergodicity). Therefore,

$$H(X) = -\lim_{T \longrightarrow \infty} \frac{1}{T} \log_2 P(x_1, x_2, \ldots, x_T). \tag{2.9}$$

How does this relate to a language model?

Perplexity: The language model can be considered as a random process model which represents different words $w_1, \ldots, w_T$. The random process will generate a sequence of words and the entropy of the language model will be given as

$$H(w) = -\lim_{T \longrightarrow \infty} \frac{1}{T} \log_2 P(w_1, w_2, \ldots, w_T). \tag{2.10}$$

If every word is equiprobable and the occurrence of one word is independent of the other, the entropy will be maximum since the uncertainty of the occurrence of a word is maximum. However, if we relate the occurrence of a word to its previous word, we may be able to predict its occurrence with greater certainty and the entropy will be lowered. In human language, the greater the number of previous words considered, the lower the entropy. For example, if the last word seen in a sample text is **"event"**, the number of possible words that can follow **"event"** will be large and almost all of them will be equally probable. However if the previous history of words considered is **"in the event"**, the probability that the next word is **"that"** or **"of"** will be much higher and the number of choices taking all three words into consideration will be much smaller.

The language model can therefore be thought of as one that has as much information content as a source which chooses words equiprobably from a vocabulary of the size of perplexity P (refer to Equation 2.6). The greater the number of equiprobable choices or "branching factor", the greater is the difficulty for the recognition system. In principle: **The lower the perplexity of the language model, the better is the language model.**

One problem with perplexity is that it does not take into account the acoustic similarity between words. In other words, a large branching factor is not a problem if the words are acoustically very different. Therefore, other measures have been proposed such as **aperplexity** and **speech decoder entropy** [12], which try to relate the language model performance to the acoustic model, but we have not considered them here. We have only used perplexity and the most important measure for our purposes, namely speech recognition accuracy.

2.3 Sparse Data

Perplexity can be decreased by making better use of the prior word sequences in the model. In terms of the n-gram approach, the greater the history considered for predicting the next word, the lower is the perplexity. Typically however, the value of n is restricted to three. One of the reasons, as discussed earlier, is that the number of unique n-grams increases exponentially as n increases. Another reason is that it is difficult to get reasonable estimates for all the valid n-grams as many of them are either unobserved or occur very few times in the training text. Word classification can be viewed as one way of decreasing the number of unobserved n-grams. Syntactic categories, as evolved by linguists, are a robust manner of word classification and have been discussed earlier. Instead of syntactic categories, we can have semantic word equivalences [11]. These equivalence classes can be manually chosen and can be domain specific. Another approach is to consider statistical equivalence classes [13], where for a word w_i, the history $w_{i-1}, \ldots, w_{i-N}$ is mapped on to an equivalence class $G(w_{i-1}, \ldots, w_{i-N})$ and is found by using an iterative algorithm so as to maximize the likelihood of the training data.

However, the problem of sparse data is inherent in the bigram or trigram model and even for word class n-grams. In case the recognizer comes across an unobserved bigram or trigram, the language model cannot afford to assign the word sequence a probability of zero since it may be a perfectly valid sequence. Several approaches have been proposed for estimating the probability of unseen n-grams, as described below. All these approaches try to deal with the problem of sparse data by coming up with reasonable estimates for the unobserved n-grams using the statistics of the observed n-grams.

Good-Turing back-off scheme

The basic idea of this scheme is to reduce "unreliable" probability estimates and distribute the probability mass obtained by this reduction among the unseen n-grams [14]. The maximum likelihood (ML) estimate for an n-gram is the relative frequency estimate. If we consider the ML estimate to be unreliable, we can reduce this estimate by replacing it with the Turing estimate which basically scales the ML estimate with a discount coefficient less than 1. Since all the probability estimates together must sum to one, this reduction leaves behind a probability mass, which can then be distributed among the unseen n-grams according to their lower order, namely, "(n-1)-gram" estimates.

Mathematically illustrating this scheme for a bigram language model, let b_r represent the number of bigrams in training which occurred r times, assuming $b_r < b_{r-1}$, and B represent the total number of bigrams, namely $B = \sum_r r b_r$. The maximum likelihood estimate for the bigram probabilities would be the relative frequency estimates,

$$P_{ML}(w_1, w_2) = \frac{c(w_1, w_2)}{B}$$

where $c(w_1, w_2)$ is the count of the bigram (w_1, w_2) in training text. The Turing estimate will be given by,

$$P_T(w_1, w_2) = \frac{(c(w_1, w_2) + 1)\frac{b_{(c(w_1,w_2)+1)}}{b_{c(w_1,w_2)}}}{B}$$

The factor $(\frac{(c(w_1,w_2)+1)}{c(w_1,w_2)})(\frac{b_{(c(w_1,w_2)+1)}}{b_{c(w_1,w_2)}}) = d_c$ is called the discount coefficient since it is less than one. Relating this to the conditional bigram probabilities, we have in case the bigram is observed in the training text,

$$P(w_2|w_1) = d_c \frac{c(w_1, w_2)}{c(w_1)}$$

where d_c is the discount coefficient and $c(w_1)$ is the count of w_1 in the training text. If the bigram has not been observed, we distribute the remaining probability mass in proportion to their unigram counts,

$$P(w_0|w_1) = (1 - \sum_{c(w_1,w_j)>0} P(w_j|w_1)) \frac{c_(w_0)}{\sum_{i:c(w_1,w_i)=0} c_(w_i)} \tag{2.11}$$

If we decide to consider counts higher than k to be reliable, we reduce the ML estimates of only those n-grams with counts less than k. Namely,

$$\begin{aligned} P(w_2|w_1) &= \frac{c(w_1,w_2)}{c(w_1)} \qquad c(w_1,w_2) \geq k \\ &= d_c \frac{c(w_1,w_2)}{c(w_1)} \quad 0 < c(w_1,w_2) < k \end{aligned} \tag{2.12}$$

where

$$d_c = \frac{(\frac{c(w_1,w_2)+1}{c(w_1,w_2)})(\frac{b_{c(w_1,w_2)+1}}{b_{c(w_1,w_2)}}) - k\frac{b_k}{b_1}}{1 - k\frac{b_k}{b_1}}$$

In case of an unobserved bigram, we use Equation 2.11.

Experiments have shown that $k = 6$ is a good choice [14]. The technique is useful since the discount coefficients can be calculated earlier and used when required. The Good-Turing back-off scheme is typically used in the standard ARPA n-gram models.

Witten-Bell back-off scheme

There are a few problems with the Good-Turing back-off estimation which prompted a modified backing off technique based on the work of Witten and Bell for adaptive text compression [15]. In particular, the Good-Turing technique is overly complex and not very robust. One problem is that it makes an assumption that the number of n-grams which occur r times will be greater than the number of n-grams which occur $r + 1$ times during training, an assumption that may not hold for all training texts.

Hence a modified back-off technique was suggested [4] whereby the observed probability estimates are reduced by adding a small factor in the denominator instead of subtracting a probability mass from the observed probability estimates. For example, if r_{w_1} is the number of unique words seen in the context of w_1, then the conditional probability of bigrams observed in the context of w_1 will be given as,

$$P(w_2|w_1) = \frac{c(w_2, w_1)}{c(w_1) + r_{w_1}}. \tag{2.13}$$

The bigram probability of a previously unseen word w_0 occurring in that context is then given as,

$$P(w_0|w_1) = \frac{r_{w_1}}{c(w_1) + r_{w_1}} P(w_0) \tag{2.14}$$

where $P(w_0)$ is the unigram probability of the word w_0. The probability mass obtained by reducing the observed n-gram probabilities is thus distributed among the unseen n-grams in proportion to their lower order estimates.

Interpolation techniques:

The interpolation method is based on the same concept of relying on more general distributions if the specific distributions are "unreliable" [16]. An interpolated model is developed by computing the probability estimate as a normalized combination of the specific distribution and the general distribution.

Consider for example a bigram model. The bigram probabilities constitute the specific distributions. If the specific distributions seem unreliable (for example, unseen bigrams will have probabilities zero), we consider a more general distribution by tying together all the bigram states observed in that context. This general distribution will be the unigram estimate for the particular context. Instead of making a hard decision between the specific distribution and the general distribution (as in the case of the

other back-off estimates), the interpolated model is a combined estimate:

$$\hat{P}(w_2|w_1) = \lambda P(w_2|w_1) + (1-\lambda)P(w_2). \tag{2.15}$$

The basic use of this interpolation is to predict unseen data, hence the λ's need to be estimated on data which has not been used for training the n-gram models. We can estimate the interpolation parameters using the Expectation-Maximization (EM) algorithm [17] on a "held-out" data set (refer to Appendix A).

Summary:

Of the three interpolation techniques described above, we compared both the back-off techniques, namely the Good-Turing back-off estimation and the Witten-Bell back-off estimation in bigram and trigram models. The results are reported in Section 2.6. Due to the simplicity and performance of the Witten-Bell approach, we have used it in our mixture model. We have also considered the interpolation technique in some aspects of our model.

2.4 Dynamic Language Modeling

Almost all the language models that we have discussed so far estimate the n-gram probabilities based on a large text corpus which serves as the training base. These probabilities remain fixed or static when used in recognition. However, intuitively we know that as we see more of the document that we are recognizing, certain n-grams will become more frequent in usage than what is reflected by the static language model especially since language in general may change over time. For example, the LM training data for the years 1987 to 1989, will not reflect bigrams associated with current events such as the US Health Care Plan or the civil war in Bosnia, or "in-

news" individuals such as Hillary Clinton or Fidel Castro.

The idea of the dynamic language model is to adapt the static model to the document seen so far. There have been different approaches to dynamic language modeling, also referred to as language model adaptation, and I have outlined some of these approaches below.

2.4.1 Dynamic Cache

The concept of the dynamic cache is to store or cache the last L words of the document seen so far. L is referred to as the window length. This idea was first implemented by [18, 19, 20, 21] where there were two bigram models. One was a static bigram model and a second was a bigram model built on the partial document cached. If the word observed next did not belong to the cache, the dynamic model was not made use of. If it did belong to the cache then a combination of the static and dynamic model was used.

A very similar approach was taken by [20] wherein, the unigram, bigram and trigram probabilities are estimated from the cache. Both the static and the dynamic trigram model are then linearly smoothed as explained in the previous subsection. The probability of the next word in the document is then computed as,

$$P(w_i|w_{i-1},w_{i-2}) = \lambda \hat{P}_{dynamic}(w_i|w_{i-1},w_{i-2}) + (1-\lambda)\hat{P}_{static}(w_i|w_{i-1},w_{i-2}) \quad (2.16)$$

where

$$\begin{aligned}
\hat{P}_{dynamic}(w_i|w_{i-1},w_{i-2}) &= \mu_1 P_{dynamic}(w_i|w_{i-1},w_{i-2}) + \mu_2 P_{dynamic}(w_i|w_{i-1}) + \mu_3 P_{dynamic}(w_i) \\
\hat{P}_{static}(w_i|w_{i-1},w_{i-2}) &= \theta_1 P_{static}(w_i|w_{i-1},w_{i-2}) + \theta_2 P_{static}(w_i|w_{i-1}) + \theta_3 P_{static}(w_i)
\end{aligned}$$

$$\begin{aligned}
\mu_1 + \mu_2 + \mu_3 &= 1 \\
\theta_1 + \theta_2 + \theta_3 &= 1
\end{aligned}$$

μ, θ and λ are the interpolating parameters. These interpolating parameters are fixed and are determined by experiments or by using the EM algorithm on text which has not been used for training or testing purposes. Alternatively, instead of using an interpolation approach for handling sparse data, we can use back-off techniques.

A similar approach was also taken by [19, 21] except that caches were maintained for each part-of-speech (POS). The words of the partially observed document were stored in their respective POS cache. The dynamic component was then used to adjust the probability of predicting the word given the class namely $P(w_i|c_i)$.

Using these techniques gave a significant reduction in perplexity and improved the recognition accuracy.

2.4.2 Dynamic Mixtures with Mixture Weights Adapted

In the case of dynamic mixtures, the concept is to adapt to the changing text styles. Humans make use of their domain knowledge to make sense of what is being spoken. Similarly, the language model can be tuned or dynamically adapted to the topic being discussed. The approach developed by Kneser and Steinbiss [22] uses bigram models which are estimated on different text corpora relating to the different topics, for example, newspaper text, scientific writing, etc. Each of the topic-specific models then contributes to an interpolated model, which uses a standard model (2.4), with the bigram probability $P(w_i|w_{i-1})$ estimated as a mixture of topic-dependent bigrams.

$$P(w_i|w_{i-1}) = \sum_k \lambda_k P_k(w_i|w_{i-1}), \tag{2.17}$$

where k refers to the specific topics and P_k refers to the bigram estimate given by that topic.

The interpolation parameters λ_k are dynamically adapted using the partially seen

document. We begin with equal values for all the λ's and then iteratively estimate the values on the previous L words cached, according to the equation

$$\lambda_k^{new} = \frac{1}{L} \sum_{i=1}^{i=L} \frac{\lambda_k^{old} P_k(w_i|w_{i-1})}{\sum_j \lambda_j^{old} P_j(w_i|w_{i-1})}. \tag{2.18}$$

In addition to topic-specific models, there is also a general model which is trained on all the training data and this is one of the models P_k used in the interpolation. It is typically observed that the λ_k for the general model is higher than the topic-specific models, since the general model has a much greater training base and therefore more robust estimates.

2.5 Modeling Long Distance Dependencies

The n-gram approach has proved to be simple to implement and very effective when used in the recognition system. Typically, the value of n has been restricted to three due to reasons discussed earlier. However with such low order dependencies, we are not able to take advantage of the long-distance dependencies in the sentence. There have been a few attempts to overcome these disadvantages and make use of long distance dependencies in a sentence or paragraph, some of which are outlined below.

2.5.1 Trigger Based Language Models: Maximum Entropy Approach

Rosenfeld and colleagues [23, 24] try to make use of long distance dependencies by defining what is referred to as *trigger pairs*. If the presence of two words are significantly correlated, for example "market" and "prices" or "weather" and "sun", they are referred to as a trigger pair with the first word being the trigger and the

second word being more likely when the trigger is observed. The trigger pairs are selected using the mutual information between two words. There can be several such trigger pairs and the information from all these pairs is combined with the static n-gram model using the Maximum Entropy Principle (MEP).

The MEP is basically a constructive criterion for setting up probability distributions from partial knowledge. It is the least biased estimate possible given partial knowledge. The partial knowledge in this case are the trigger pairs and the n-gram statistics. They are reformulated as constraints on the expectation of different functions, and the probability distribution which satisfies these constraints and has the highest entropy is selected.

The main disadvantage of the trigger-based LM is the computational complexity while training the models as well as during recognition. The advantages are that different knowledge sources can be easily incorporated in this model by redefining them as one of the constraints that is to be satisfied with the MEP. For example, the idea of a dynamic cache as explained in a previous section can also be incorporated by formulating it as a constraint and making use of self-triggers (the idea that if a word has just been observed, there is a greater likelihood that it will be seen again).

2.5.2 Using Decision Trees: Clustering Word Histories

One way of looking at the n-gram models is to consider them to map the previous history of a word w_i to the $n-1$ previous words. In other words, we consider the history of a word to be equivalent if the previous $n-1$ words are the same in each case. We can however consider greater histories (i.e. effectively consider greater values of n) if instead of this naive equivalence class approach we make use of decision trees to determine the equivalence classes. Bahl *et al.* [25] have applied binary decision trees to the language modeling problem and thus tried to take advantage of longer

histories.

At each nonterminal node in the tree, there is a question requiring a yes/no answer. The leaves of the tree are the equivalence classes. Since the objective of asking the questions is to reduce the uncertainty of predicting the next word, questions which minimize average entropy at the leaf are selected. In order to prevent the tuning of the tree to the training data, the training data is split into two sets. If the questions which minimize the entropy on one set do not minimize the entropy on the second set, it is discarded.

A distinct advantage of this technique is that we do not have to restrict the predictor to the last two words. The predictor can be parser-based information or it can be semantic information depending on the kind of questions that are asked at the nonterminal nodes. The disadvantage of this approach is that it requires massive computation to construct the tree and a significantly larger search space to handle the longer term dependency.

2.5.3 Extended n-grams

Another approach to modeling long distance dependencies basically extends the concept of a n-gram from the n words that immediately precede it to the n most appropriate words that precede it. In the work of Wright *et al.* [26], the extended bigrams $P(w_i|w_{j(i)})$ choose a most useful conditioning word $w_{j(i)}$ from a window with complex normalization techniques to obtain a valid probability distribution. The window includes all words since the beginning of the sentence. The most useful word is chosen according to a relative information measure.

Another technique is to split the vocabulary into content words and function words. Function words are closed-class words that have little or no semantic content

such as: "is", "if", "to", "are", while content words are usually nouns, adjectives and other words which do not occur as frequently as the function words. The extended bigrams in this case [27] predict a word based on its previous content word if the word is itself a content word, or on a function word in case the word is a function word.

2.6 Summary

In this chapter, we have discussed some of the previous work in the area of statistical language modeling. We have described the standard n-gram models and some of the problems encountered while training and testing these models. Since robust parameter estimation is an important issue in the light of sparse data for training the n-gram models, we have discussed in detail some of the smoothing techniques that are used in current n-gram models. We experimented with two of these techniques, namely the Witten-Bell and the Good-Turing back-off estimation. Based on the perplexity and recognition results obtained, we have implemented the Witten-Bell back-off technique for robust estimation of the n-gram probabilities in our model.

The disadvantage of the standard n-gram model lies in the fact that it is unable to take advantage of topic-related long distance dependencies within and across sentences. The problem of representing long-distance dependencies has been explored earlier in stochastic language models. Language model adaptation addresses the problem of inhomogeneity of n-gram statistics, but really only represents task level dependencies and cannot account for dependencies within a sentence. Context-free grammars could account for sentence dependencies but it is costly to build task-specific grammars. Some of the other techniques suggested are computationally very expensive while training the models. In this thesis, we introduce a sentence-level

mixture of m topic-dependent language models, which makes use of the existing statistical language modeling techniques and at the same time takes advantage of topic-related long distance constraints within and across sentences, by making simple variations in the standard model. In addition, we show how the mixture model can also incorporate the dynamic language model adaptation.

Chapter 3

Paradigm for Speech Recognition

Different approaches have been taken towards acoustic modeling for continuous speech recognition. The statistical approach plays an important role today in most successful large vocabulary continuous speech recognition systems. Statistical techniques make practical the training and testing of large networks of connected word or sub-word units. In most of these systems, words are modeled in terms of smaller subword units called phonemes. The acoustic training data is transcribed into a phonetic string and the statistics for the phonemes are compiled using this data. Once the acoustic model is trained, it can transcribe a speech waveform into a string of phonemes. A linguistic decoder then translates this string of phonemes into words [2].

The Hidden Markov Model (HMM) [2] is widely used in statistical speech recognition systems for acoustic modeling. In order to construct word models, HMM's are constructed either for the entire word or for smaller word units referred to as phones. These models are then connected in a network according to a pronunciation dictionary. The acoustic model used in the BU recognition system, referred to as the Stochastic Segment Model (SSM), is also a statistical model but somewhat different from an HMM. This chapter briefly outlines the BU recognition system, namely the

SSM and the N-best rescoring formalism. We conclude this chapter with a description of the language model training corpus and the evaluation paradigm.

3.1 Stochastic Segment Model

An alternative approach to HMM's, the SSM, was proposed by [28, 29], wherein the model considers speech at the segmental level so as to better capture the spectral/temporal structure over the duration of the phone. In the SSM, a variable-length segment of speech is considered, which comprises L frames of speech. The version of the SSM used in this work uses full-covariance Gaussian density functions and assumes the frames within a segment to be conditionally independent, which does not take complete advantage of the SSM formalism. This segment of speech is mapped to a fixed-length representation, comprising m frames or samples, using a linear transformation. The model for each phone has m regions. Given the phone model, the distribution for each of the m frames is mapped onto the m regions of the model for that phone.

In order to take advantage of the effects of phone context and co-articulatory effects, context-dependent phone models are used. In other words, the phone has been modeled conditioning the distributions on its left and right phone context. More specifically, in our experiments, robust context-dependent covariance estimates are obtained by tying covariance parameters across classes of left and/or right context [30], instead of using the clustering techniques described in [31].

The segment model is trained by using an iterative algorithm wherein, the acoustic training data is segmented using the current models and the new parameters are estimated by maximizing the likelihood of the parameters given the new segmentations [28]. During phone recognition, the SSM maximizes the probability of

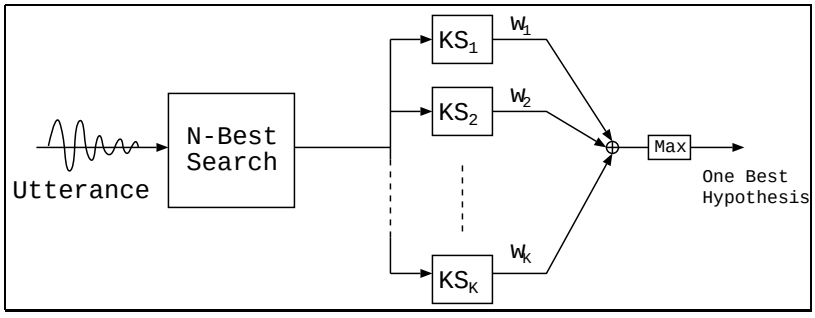

Figure 3.1: *Schematic of the N-best paradigm.*

the phone sequence and the segmentation.

3.2 N-best Rescoring: the General Strategy

There are several knowledge sources (KS) that can be useful in a recognition system. One of the approaches of combining the knowledge sources is N-best rescoring, as illustrated in Figure 3.1. The input to the recognition system is a speech signal (utterance). A search algorithm is used to come up with the best N sentence hypotheses for that utterance. Different knowledge sources can then be used for rescoring this N-best list. A weighted combination of these scores is used for reranking the hypotheses.

In our case, the BBN BYBLOS Hidden Markov Model (HMM) [32, 33] decoder is used to find the N-best. Typically, the value of N that we use for our experiments is 100. The different knowledge sources that we use to rescore the top N hypotheses include all or some subset of: the Stochastic Segment Model (SSM), the HMM acoustic model from BBN, the BBN trigram language model, the BBN Segmental Neural Net acoustic scores (SNN) [34], the phone duration scores, the number of phones, the number of silences and the language model developed during the course of this work. Not all the knowledge sources are used in all the experiments in this

thesis. Unless otherwise noted, the BBN scores are not used. A weighted linear combination of these knowledge source scores (log likelihoods and counts) is used for rescoring and the weights (w_k) are optimized to decrease the word error. The weights are optimized [35] on a development set and kept fixed over other test sets. An important observation is that it is computationally cheap to include our language model in the recognition system in the context of the N-best rescoring paradigm, but it would be expensive to use it in a beam or stack search primarily due to the fact that our model is not Markov.

3.3 Corpus and Lexicon Resources

The corpus that we have been working with over the past year is the Wall Street Journal corpus (WSJ) [36]. Besides Boston University, there are other research groups which use the same corpus. In order to compare the effectiveness of different algorithms without variability in the training and test data, the lexicon and training data is standardized across these sites [37].

3.3.1 Training and Evaluation Corpora

The training corpus used for language model development has been available to BU in two phases. The initial training data provided by the Linguistic Data Consortium (LDC) is referred to as the WSJ0 data. This training comprised 38M words of data organized in the form of paragraphs with sentence demarcations. It mainly comprised business news and stock market articles obtained from the Wall Street Journal over the period 1987-1989.

The WSJ data was initially designed to support both verbalized punctuation (VP) as well as non-verbalized punctuation (NVP) dictation. Parallel corpora were

developed for VP and NVP language modeling training and recognition evaluation. For the VP corpus, the speakers were requested to articulate the punctuations explicitly. For the NVP corpus, the speakers were requested to not pronounce the punctuations at all. During the second phase of the data collection, the VP/NVP distinctions were dropped and it was optional for the speaker to verbalize a punctuation.

Since the speakers were given the exact transcriptions to speak, the WSJ0 data does not take into account speaking-style variability. For example, "101" is referred to as "HUNDRED ONE" but we know that speakers may also say "HUNDRED AND ONE" or "ONE HUNDRED AND ONE". BBN therefore processed the NVP WSJ0 data to account for this and other similar variations [38]. Since experiments indicated that greater training data gives better language models [38], BBN did some simple processing of WSJ1 text data from 1990-1992 (38M words, not officially designated as training data), and we have used this data along with the WSJ0 data for training our language models.

The 76M words of WSJ data from the 1987-1992 are used for training the language models. Besides this data, we have two sets of test data for recognition experiments. These are referred to as the development and evaluation data sets. During the course of this thesis, we have also used the WSJ1 speech transcriptions for weight estimation purposes, which are typically not used for training the language model by other sites.

To summarize,

1. We used a total of 76M words of non-verbalized pronunciation (NVP) training data for estimating our topic and general model parameters. (38M NVP 1987-1989, 38M BBN 1990-1992).

2. The remaining 38M words of verbalized punctuation data was used for estimat-

ing the VP model, which was used for smoothing topic models.

3. In addition, we also used the WSJ1 speech transcriptions as a "held-out" data set for estimating interpolation weights.

4. We used the development test data for estimating the weights for combining scores in N-best rescoring.

5. Finally, we tested our models on the evaluation data set, for which comparable results from other sites were available.

For a fair evaluation of a speaker independent recognition system, the two test sets, namely the development and evaluation test sets, use different speakers and hence there is a difference in average recognition results.

Additional language model training data has been drawn from recent publications of five sources of North American Business, namely Dow Jones Information Services (DJIS), New York Times (NYT), Reuters North American Business Report (RNAB), Los Angeles Times (LAT) and the Washington Post (WP). The data basically considers the business portions of the above sources and amounts to around 1.6GB of text data. This data has however not been used for training models in this thesis because of its late release. However, the additional training data should prove to be beneficial for more robust topic-dependent models.

3.3.2 Lexicon

The 64,000 most frequent words in the training corpus are culled into a word list. We work with a subset, a 5000 word verbalized punctuation dictionary (5K VP). This standard dictionary refers to words that have been used during training. However speakers vary in how they say abbreviations and acronyms. For example, CORP. may

be verbalized as CORPORATION but can also be spoken as CORP. (similarly INC. vs. INCORPORATED, UNQUOTE vs. END-QUOTE) [38]. Therefore, these words are added to the standard VP dictionary before training. We have two additional symbols, one for the beginning of the sentence and one for an unknown word.

3.4 Development and Evaluation Paradigm:

In order to compare across sites that use the same data and lexicon, every year all sites report on a test set referred to as the evaluation set [42]. Typically, a development set is provided and new ideas are implemented and tested on this test set before using them on the evaluation test set. Word recognition performance is reported in terms of percent word error. This is calculated as the ratio of the sum of insertions, deletions and substitutions to the true number of words in the sentence. In addition, standard software exists for testing the statistical significance in the difference between two systems evaluating on the same data. During the course of this work, we have evaluated on the H2-P0 condition [42], on which others have reported results in the range of 4.9-9.2% word error using trigram language models [42] and the BU baseline performance is 5.8%.

We developed and tested our model in two phases. In the first phase, we used the 5K VP dictionary and the training data referred to above for developing our models. We evaluate on the 1993 development test set to measure perplexity and word recognition error. In order to compare our approach with other approaches in the field and to obtain fair results since we estimate the score-combining weights on the development set, we also report results on the 1993 evaluation set.

Chapter 4

Static Mixture Language Model

As we have already discussed, long distance dependencies can be due to the inhomogeneous nature of language; different words or sequences are more likely for particular broad topics or tasks. In this thesis, we represent long distance dependencies by using a simple model which is a sentence-level mixture of m component language models, each of which can be identified with the n-gram statistics of a specific topic or a broad class of sentences. Though this approach is similar to the approach of Kneser and Steinbiss [22], the important difference between the two approaches is that Kneser and Steinbiss use mixtures at the n-gram level while we consider mixtures at the sentence level. Thus, the probability of the word sequence is

$$P(w_1, \ldots, w_T) = \sum_{k=1}^{m} \lambda_k \prod_{i=1}^{T} P_k(w_i | w_{i-1}, \ldots, w_{i-n+1}). \tag{4.1}$$

This approach has the advantage that it can be used either as a static model, as described in this chapter, or as a dynamic model as will be discussed in Chapter 5, and can easily leverage the techniques that have been developed for adaptive language modeling.

In order to compute the probability of a sentence in the manner suggested

by Equation 4.1, we need to consider several issues. To begin with, this chapter describes our approach to automatically obtain different topic-data, since "topics" are not explicitly marked in many corpora. We then elaborate on the parameter estimation and smoothing techniques that we developed and experimented with in order to obtain robust topic n-gram probabilities and mixture weights. We conclude with a discussion of the results from the different experiments and a description of the baseline static mixture model which has been used for further development.

4.1 Automatic Clustering of Topics

The m component distributions of the language model correspond to different "topics", where topic can mean any broad class of sentences. Topics can be specified by hand if text labels are available or heuristic rules associated with known characteristics of a task domain. Topics or natural groupings of data can also be determined automatically, which is the approach taken here since the standard WSJ language model training data does not have topic labels associated with them.

Because of its conceptual simplicity, agglomerative clustering is used to partition the training data into the desired number of clusters/topics. To reduce clustering time, the clustering is at the paragraph level, relying on the assumption that an entire paragraph comes from a single topic. Each paragraph begins as a singleton cluster. Paragraph pairs are then progressively grouped into clusters by computing the similarity between clusters and grouping the most similar two clusters. The basic clustering algorithm is as follows:

1. Let the desired number of clusters be C^* and the initial number of clusters C be the number of singleton data samples, or paragraphs.

2. Find the best matched clusters, say A_i and A_j, to minimize the similarity

criterion S_{ij}.

3. Merge A_i and A_j and decrement C.

4. If current number of clusters $C = C^*$, then stop; otherwise go to Step 2.

At the end of this stage, we have the desired number of partitions of the training data. To save computation, we run agglomerative clustering first on subsets of the data, and then continue by agglomerating resulting clusters into a final set of m clusters.

Similarity measure: A variety of similarity measures can be envisioned. We initially used a normalized measure of the number of content words in common between the two clusters. We refer to this measure as the set-intersection measure in the remainder of the thesis. Paragraphs comprise both function words (e.g. is, that, but) and content words (e.g stocks, theater, trading), but the function words do not contribute towards the identification of a paragraph as belonging to a particular topic so they are ignored in the similarity criterion. Letting A_i be the set of unique content words in cluster i, $|A_i|$ the number of elements in A_i, and N_i the number of paragraphs in cluster i, then the specific measure of similarity of two clusters i and j is

$$S_{ij} = \frac{N_{ij}|A_i \cap A_j|}{|A_i \cup A_j|}, \tag{4.2}$$

where

$$N_{ij} = \sqrt{\frac{N_i + N_j}{N_i \times N_j}} \tag{4.3}$$

is a normalization factor [39] used to avoid the tendency for small clusters to group with one large cluster rather than other small clusters.

There have been other approaches to clustering for automatic sublanguage identification. One such method is that by [40], where the similarity measure is based on the combination of inverse document frequencies of words. Specifically the similarity

measure is given by

$$S_{ij} = \sum_{w \in A_i \cap A_j} \frac{1}{|A^w|} \times \frac{1}{|A_i||A_j|} \tag{4.4}$$

where $|A_i|$ is the number of unique words in article i and $|A^w|$ is the number of articles containing the word w. We refer to this measure as the inverse document frequency measure in the remainder of the thesis. An advantage of this measure is that the function words do not have to be discarded since their high frequencies will automatically discount them. However, we discovered that for obtaining reasonably even sized clusters, we need to include the normalization factor as stated in Equation 4.3. Therefore, the similarity measure is actually implemented as,

$$S_{ij} = \sum_{w \in A_i \cap A_j} \frac{N_{ij}}{|A^w|} \times \frac{1}{|A_i||A_j|} \tag{4.5}$$

We implemented clustering using this metric and obtained a similar recognition accuracy for both metrics, as will be shown in Section 4.4.

4.2 Parameter Estimation

In parameter estimation, there are two issues that we require to deal with. First, the topic-model parameters have to be estimated from the training data that has been partitioned into different topic-data. We consider two iterative training options here, the EM algorithm and a "Viterbi-style" training technique [5]. Second, once we obtain the topic-model estimates, we need to smooth these estimates to account for sparse data and non-topic sentences. We deal with the second problem by using double mixtures, one at the sentence-level and the other at the n-gram level. In this section, we first explain the training algorithms we investigated and then discuss the additional smoothing techniques used.

4.2.1 Iterative Topic Model Re-Estimation

Each component model is a conventional n-gram model. Initial estimates for the n-gram models are based on the partitions of the training data obtained by the clustering procedure suggested above, using the Witten-Bell back-off technique. Since the clustering algorithm is very simple, the "topic" parameters may not be robust enough. Hence, we iteratively re-estimate these parameters, since some sentences belonging to one topic according to the clustering metric be more likely in another topic according to the n-gram models.

We can re-estimate these parameters using the Expectation-Maximization (EM) algorithm [17]. The EM algorithm is a technique for finding maximum likelihood parameter estimates where the observed data is in some way incomplete. In our case, the data is incomplete, since we do not know exactly which topic cluster a certain sentence belongs to. The solution is to estimate the missing information (the topic) and then find the maximum likelihood estimate. The EM algorithm that we developed for n-gram parameter estimation, with a Witten-Bell type back-off, is explained in detail in Appendix A. For a trigram language model, the trigram (w_b, w_c, w_d) probability for the jth topic-model is computed as,

$$P_j(w_d|w_b, w_c) = \frac{\sum_{i=1}^{n} \hat{z}_{ij}^{(p)} n_{bcd}^{i}}{\sum_q \sum_{i=1}^{n} n_{bcq}^{i} \hat{z}_{ij}^{(p)} + \sum_q \frac{\sum_{i=1}^{n} n_{bcq}^{i} \hat{z}_{ij}^{(p)}}{\sum_{i=1}^{n} n_{bcq}^{i}}} \tag{4.6}$$

where n represents the number of training sentences, $\hat{z}_{ij}^{(p)}$ is the likelihood of the sentence i belonging to the jth topic and n_{bcd}^{i} represents the number of occurrences of the trigram w_b, w_c, w_d in the ith sentence. Similarly, the bigram (w_b, w_c) probability for the jth topic-model is computed as,

$$P_j(w_c|w_b) = \frac{\sum_{i=1}^{n} \hat{z}_{ij}^{(p)} n_{bc}^{i}}{\sum_q \sum_{i=1}^{n} n_{bq}^{i} \hat{z}_{ij}^{(p)} + \sum_q \frac{\sum_{i=1}^{n} n_{bq}^{i} \hat{z}_{ij}^{(p)}}{\sum_{i=1}^{n} n_{bq}^{i}}} \tag{4.7}$$

where n^i_{bc} represents the count of the bigram w_b, w_c in the ith sentence. The unigram (w_b) probability is estimated as,

$$P_j(w_b) = \frac{\sum_{i=1}^{n} \hat{z}_{ij}^{(p)} n_b^i}{\sum_q \sum_{i=1}^{n} n_q^i \hat{z}_{ij}^{(p)}} \tag{4.8}$$

where n_b^i represents the count of the unigram w_b in the sentence i. The back-off probabilities are computed using an EM extension of the Witten-Bell technique, and the bigram back-off for (w_b, w_c) in the jth topic-model can be estimated as,

$$P_j^{backoff}(w_b, w_c) = \frac{\sum_q \frac{\sum_{i=1}^{n} n^i_{bcq} \hat{z}_{ij}^{(p)}}{\sum_{i=1}^{n} n^i_{bcq}}}{\sum_q \sum_{i=1}^{n} n^i_{bcq} \hat{z}_{ij}^{(p)} + \sum_q \frac{\sum_{i=1}^{n} n^i_{bcq} \hat{z}_{ij}^{(p)}}{\sum_{i=1}^{n} n^i_{bcq}}}. \tag{4.9}$$

Similarly, the unigram back-off can be computed as,

$$P_j^{backoff}(w_b) = \frac{\sum_q \frac{\sum_{i=1}^{n} n^i_{bq} \hat{z}_{ij}^{(p)}}{\sum_{i=1}^{n} n^i_{bq}}}{\sum_q \sum_{i=1}^{n} n^i_{bq} \hat{z}_{ij}^{(p)} + \sum_q \frac{\sum_{i=1}^{n} n^i_{bq} \hat{z}_{ij}^{(p)}}{\sum_{i=1}^{n} n^i_{bq}}}. \tag{4.10}$$

The EM algorithm is computationally intensive and hence we initially used a Viterbi-style [5] training technique that iteratively partitions the data by topic and re-estimates the topic-dependent parameters.

$$\begin{aligned} \hat{topic} &= \max_{topic} P(sentence|topic) \\ &= \max_{topic} \prod_{i=3}^{i=m} P(w_i|w_{i-1}, w_{i-1}, topic). \end{aligned}$$

Once the sentences have been relabeled by topics, we again estimate the parameters of the m component models using the newly labeled data and the Witten-Bell back-off technique. The stopping criterion is very simple, when the size of the m clusters becomes nearly constant, we stop the estimation procedure. The EM algorithm is a potentially more powerful way of estimating the parameters since we can assign the same sentence to different topics, allowing all the topics to take advantage of the entire training data. However, the EM algorithm is potentially more costly and may

not provide significant performance advantages. Therefore, we will evaluate the two training options experimentally.

4.2.2 Topic Model Smoothing

There are a few problems associated with the m component models that have been trained using either of the training options referred to above. These models are trained on subsets of the training data, hence there is a danger that they may be undertrained. Another problem is that in case we come across "non-topic" sentences, we may not be able to give the sentence a reasonable probability. A third problem is that in our experiments with the mixture models, we used non-verbalized punctuation data and the BBN data which had rule-based verbalized punctuation. Hence the verbalized punctuation n-grams may not have reasonable estimates.

We deal with the "non-topic" problem by including a general model in addition to the m component models at the sentence level. This general model is trained on the entire training data. An alternative approach could have been to interpolate the general model with the component models at the n-gram level but then the component models might become too general. For smoothing the n-gram estimates of the component models due to sparse data and lack of verbalized pronunciation training, we interpolate with a model trained on the VP data at the n-gram level. Specifically, the model is given by

$$P(w_1, ..., w_T) = \sum_{k=1,..,m-1,G} \lambda_k \prod_{i=1}^{T} (\theta_k P_k(w_i|w_{i-1}, w_{i-2}) + (1-\theta_k) P_{G'}(w_i|w_{i-1}, w_{i-2})) \quad (4.11)$$

where $P_{G'}(\cdot|\cdot)$ is a general model, but trained on all the data from the verbalized punctuation training set and $P_G(\cdot|\cdot)$ is the general model trained on all the remaining data (the NVP data and the additional BBN-processed WSJ1 data).

There are two sets of interpolation parameters that we need to estimate, the sentence-level mixture weights λ and the n-gram level interpolation weights θ. Since these parameters are for smoothing for unseen data (either "non-topic" or unseen trigram), we need to estimate them on a separate data set. We use the WSJ1 speech transcriptions, which had not been used for training, for estimating these parameters. To simplify the implementation, the two sets of weights were estimated separately. The sentences in the "held-out" data set are labeled according to their most likely topic and thus the data is split into m clusters. A simple maximum likelihood criterion is used to estimate the θ's on the respective cluster beginning with uniform weights for each θ. For each topic (cluster)

$$\theta_k^{new} = \frac{1}{N} \sum_{i=1}^{N} \frac{\theta_k^{old} P_k(w_1, .., w_{n_i})}{\sum_j \theta_j^{old} P_j(w_1, \ldots, w_{n_i})} \tag{4.12}$$

where n_i is the number of words in sentence i, and N is the total number of sentences in cluster k. After the component models have been estimated, the λ's are estimated in the same way using an analogous algorithm over the entire data set. This is an iterative alogorithm. However, in our experiments, we iterated only once over the data set, assuming a very fast convergence.

4.3 Implementation Details

As the training data increases, the number of unique trigrams increases. Two important issues for efficiently using the language model are: efficient storage of the model and fast access of the n-gram probabilities. To deal with the problem of storage, we implemented an efficient structure for the model, such that only the relevant details are stored. We experimented with two techniques to store the probabilities to a lesser precision. The first technique is a simplified version of the sign/logarithm number system exploited by Brown [41] (and references therein). The second technique

involves storing the probabilities to a precision of three places after the decimal. We observed that both these techniques reduce storage requirements considerably and at the same time do not affect the perplexity or recognition accuracy when used on different sets of text data.

In order to decrease access time and prevent paging due to excessive memory requirements, we parallelized our software such that we work with a subset of the vocabulary at any given time. Since it is difficult to train the entire vocabulary simultaneously because of excessive memory requirements, we train a subset of the vocabulary at a time. The training algorithm can be run simultaneously on different machines. During testing, when we compute the log probability of an entire utterance, the log probabilities of the different vocabulary subsets are summed after the entire vocabulary has been considered. For example, the probability of the sentence $w_1, w_2, \ldots, w_T$ is computed by a trigram language model as, $\prod_{i=1}^{T} P(w_i|w_{i-1}, w_{i-2})$. Hence, the log probability can be given as,

$$\begin{aligned} \log P(w_1, w_2, \ldots, w_T) &= \sum_{i=1}^{T} \log P(w_i|w_{i-1}, w_{i-2}) \\ &= \sum_{k} \sum_{i:w_i \in V_k} logP(w_i|w_{i-1}, w_{i-2}) \end{aligned}$$

where V_k is the subset of the vocabulary currently under consideration. If V_k comprises w_1 and w_3, then during this pass, we compute the log probability of all instances of w_1 and w_3 in the utterance, for example, the unigram w_1 and the trigram w_1, w_2, w_3. Once the entire vocabulary has been considered, the different log probabilities are summed to obtain the log probability of the entire utterance. This implementation is important in our case, since with our computing power, it is difficult to consider all the clustered models at the same time.

4.4 Experiments

We implemented both the Good-Turing back-off estimation technique as well as the Witten-Bell back-off technique in bigram and trigram models. Since, the Witten-Bell technique is simpler to implement and consistently results in small improvements in recognition accuracy, as indicated by the results below, in all the future models that we implemented for the thesis, we used the Witten-Bell technique. An interesting observation here is that there is an increase in recognition accuracy, in spite of a slight increase in perplexity. The perplexity based on n and back-off technique used and the corresponding recognition results are given in Table 4.1.

Table 4.1: *Perplexity on 1992 ARPA LM development test set and recognition performance on 1992 ARPA 5K development test set (for Good-Turing vs. Witten-Bell).*

Model	Back-off	Perplexity	Recognition
Bigram	Good-Turing	118	7.4
	Witten-Bell	121	7.2
Trigram	Good-Turing	65	6.0
	Witten-Bell	68	5.7

We mainly experimented with single-component and five-component mixtures of trigram models. To explore the tradeoff of using different number of clusters, we also worked with eight-component trigram mixtures. There was no significant difference in the perplexity and recognition accuracy, between a five-mixture and eight-mixture model. Hence, we worked with five-component mixture for the remaining experiments to reduce computation and storage costs.

The recognition error on the ARPA 5K development and evaluation test sets us-

ing the five-component cluster models is reported in Table 4.2. In these experiments, the data was clustered with the set intersection similarity measure (Equation 4.2), and we used the Viterbi-style training technique for parameter estimation.

Table 4.2: *Recognition results: 1993 ARPA 5K development and evaluation test (Model implemented with set-intersection similarity measure and Viterbi training).*

LM	Recognition Error %	
	Dev	Eval
BBN LM	7.4	6.1
(5-mixture) BU LM	7.1	5.8
BBN+BU LM	7.0	5.7

The recognition results obtained by using the five-component cluster models with the set intersection similarity metric and the Viterbi-style parameter estimation indicated a gain in recognition accuracy. The BBN trigram language model against which the recognition results have been compared is basically the general model trained on all the training data. The acoustic model used in these experiments is the SSM.

We also investigated combining the BU mixture language model with the BBN trigram language model. Since the scores are combined in a non-linear manner, the combination is not yet another mixture term and could potentially give a small improvement in accuracy over using only the BU language model. In this experiment, as in most experiments, there was no significant gain from linearly combining the two log LM scores.

The results were extremely encouraging. As a next step, we implemented cluster-

ing using the inverse document frequency similarity metric referred in Equation 4.5. The rest of the experimental conditions and parameter estimation techniques were unchanged. The new clustering metric gave a slight improvement in our recognition accuracy on the development set and mixed results on the evaluation. Here again the acoustic model used is the SSM.

Table 4.3: *Recognition results: 1993 ARPA 5K development and evaluation test (Model implemented with inverse-document frequency similarity measure and Viterbi training).*

LM	Recognition Error %	
	Dev	Eval
BBN LM	7.4	6.1
(5-mixture) BU LM	7.0	5.8
BBN+BU LM	7.0	5.8

The differences in the results which degenerated/improved on the development and the evaluation set could not be considered significant. The two similarity metrics gave us almost the same performance with the other experimental conditions remaining unchanged. We decided to use the inverse document similarity measure for the remaining experiments since it appears to be broadly used in language processing work.

Since we experimented with a small number of clusters, it is difficult to see coherent topics in them. However, it appears that the current models are putting news related to foreign affairs (politics, as well as travel) into one cluster and news relating to finance (stocks, prices, loans) in another.

As we have discussed earlier in this chapter, using the Expectation-Maximization algorithm after the first stage of clustering, though computationally expensive, is a more robust parameter estimation technique. Hence, after clustering the data using the inverse document frequency similarity metric, we estimated the model parameters using the EM algorithm. The recognition results from this experiment are as indicated in Table 4.4. The acoustic model used is the SSM.

Table 4.4: *Recognition results: 1993 ARPA 5K development and evaluation test (Model implemented with inverse-document frequency similarity measure and EM training).*

LM	Recognition Error %	
	Dev	Eval
BBN LM	7.4	6.1
(5-mixture) BU LM	7.0	5.7
BBN+BU LM	6.9	5.7

In general, there was a consistent though not very large improvement in the recognition accuracy due to EM training. However, the optimization of the weights on the development data set indicated that the BU mixture LM with EM training was more heavily weighted relative to the BBN LM than in the case of the Viterbi-trained mixture LM. Since the additional cost of using EM over Viterbi is mainly increased storage and not computation, we felt that the cost of EM was not a significant disadvantage.

Besides the SSM, we also had other knowledge sources available in the form of the HMM and Segmental Neural Net (SNN) acoustic scores. When we combine all the knowledge sources together, the sequence of modifications in the mixture model gave

statistically insignificant differences in the recognition accuracy on the development and evaluation sets. Table 4.5 reports results for the system that combines the SSM with the HMM and the SNN scores obtained from BBN in addition to the BBN trigram model and the mixture language model. The only modifications made are to the mixture language model; all the other knowledge sources remaining the same. If we denote the set intersection similarity metric as S1 and the inverse document frequency similarity metric as S2, the recognition results on the development and evaluation test sets are as given in Table 4.5.

Table 4.5: *Recognition results: 1993 ARPA 5K development and evaluation test, S1: set intersection similarity metric, S2: inverse document frequency similarity metric, includes all knowledge sources.*

BU LM	Recognition Error %	
	Dev	Eval
S1+Viterbi training	6.3	5.3
S2+Viterbi training	6.3	5.3
S2+EM training	6.2	5.3

For reference, the best reported result on this evaluation test is 4.9% word error by Cambridge University (HTK group), using a trigram language model.

4.5 Baseline Static Mixture Model

From the experiments conducted with the static mixture model, we decided on the baseline static mixture model to comprise the following:

- Clustering using the similarity metric suggested by [40].
- The topic n-gram parameter estimation using the EM algorithm.
- The sentence-level mixture weights and the smoothing weights are estimated using the WSJ1 acoustic transcriptions.

This baseline model is used for further experimentation in the development of the dynamic language model.

Chapter 5

Dynamic Mixture Model

The parameters of the n-gram language models are estimated from a large training corpus. In the last chapter, these parameters remained static when we used them on different test sets. However, there might be short-term fluctuations in the word frequencies due to several reasons. For example, certain words might become more prominent in usage due to a sub-language topic shift in the conversation. The word frequencies might increase or decrease depending on the speaking-style of the speaker. Dynamic language modeling attempts to capture these short-term fluctuations in the language, and it easily fits into the framework of the sentence-level mixture model.

Research in the area of dynamic language modeling [20, 21, 22, 24] has basically concentrated on two approaches: a cache model for capturing the fluctuations in word frequencies, and dynamic mixture weight adaptation to capture sub-language topic shifts in the document. We adapted both these techniques to our mixture language model via simple variations to the baseline static model.

In this chapter, we outline the two different dynamic modeling techniques that we have incorporated in our sentence-level mixture model and conclude with some of the experimental results obtained for these two techniques, used alone and in

combination.

5.1 Mixture Weight Adaptation

The static mixture language model described in Chapter 4, comprises m different "topic" language models. These different models are interpolated at the sentence-level, assuming that there will not be a significant topic shift within a sentence. A human listener usually conditions her/his expectations on what might be said based on what she/he has already heard in the conversation. For example, if the dictated material or conversation is about law and order, the frequency of "JUDGE", "LAWYER","CASES", etc. might increase compared to other words. As the conversation shifts from law and order to finance, words such as "STOCKS", "SHARES" might become more frequent in usage. If we have a mixture of topic-dependent models, one of which is based on law and the other on finance, then, as the topic shifts and/or our estimates of the topic become more reliable, we would like to place more emphasis on the estimates from the specific topic models. This can be considered equivalent to identifying the topic in a document or tracking the topic in a conversation. There have been other approaches to dynamically adapting to topic-shifts in a document [22]. In our case, we adapted the sentence-level mixture weights to the changing text-styles.

Recall that the underlying equation for the baseline mixture model is,

$$P(w_1, \ldots, w_T) = \sum_{k=1}^{m} \lambda_k \prod_{i=1}^{T} P_k(w_i | w_{i-1}, \ldots, w_{i-n+1}). \tag{5.1}$$

During dynamic mixture weight adaptation, we re-estimate the sentence-level mixture weights $\{\lambda_k : k = 1 \ldots m\}$ to reflect the increased confidence in the choice of weights, as more data is observed. In order to dynamically estimate the sentence-level mixture weights, we cache the partial document observed so far. The λ's are then adapted

using Equation 4.12 and the cached partial document, namely,

$$\lambda_k^{new} = \frac{1}{N}\sum_{i=1}^{N}\frac{\lambda_k^{old}P_k(w_1,\ldots,w_{n_i})}{\sum_j \lambda_j^{old}P_j(w_1,\ldots,w_{n_i})} \tag{5.2}$$

where N is the number of sentences of the document observed so far and n_i is the number of words in the ith sentence. Ideally, we would like to update the $\{\lambda_k : k = 1\ldots m\}$ after every utterance using all of the document observed so far. Fortunately, the adaptation can be expressed as a simple recursive update as follows: Let λ_k^{init} represent the initial value of λ_k, the sentence level mixture weight for the kth mixture component estimated using Equation 4.12 on previously unobserved data.

1. After each sentence i is recognized, compute $\forall k = 1,\ldots,m$,

$$z_k^i = \frac{\lambda_k^{init}P_k(w_1,\ldots,w_{n_i})}{\sum_j \lambda_j^{init}P_j(w_1,\ldots,w_{n_i})} \tag{5.3}$$

   where n_i is the length of the sentence i and $P_k(w_1,\ldots,w_{n_i})$ is the probability of the ith sentence $w_1,\ldots,w_{n_i}$, computed during recognition using the parameters of the kth mixture component. z_k^i represents the likelihood of the sentence $w_1,\ldots,w_{n_i}$ belonging to the kth topic. If we use the current values of $\{\lambda_k : k = 1\ldots m\}$, this likelihood will be biased towards the document observed so far. In order to obtain an impartial estimate of z_k^i, we use the initial values $\{\lambda_k^{init} : k = 1\ldots m\}$.

2. Update the mixture-weights after every sentence,

$$\begin{aligned}\lambda_k^N &= \frac{1}{N}\sum_{i=1}^{N} z_k^i \\ &= \frac{N-1}{N}\lambda_k^{N-1} + \frac{z_k^N}{N},\end{aligned} \tag{5.4}$$

   where N is equal to the number of sentences observed so far.

A significant advantage of this algorithm is that we are able to use all of the document observed thus far for estimating the topic weights. Another important advantage of the above technique as compared to [22] is that the computation and storage costs are negligible, since we compute the z_k^i once at the end of every sentence and $P_k(w_1, \ldots, w_{n_i})$ is computed during recognition. Additional storage is required only to save the initial mixture weights $\{\lambda_k^{init} : k = 1, \ldots, m\}$. The mixture weight adaptation algorithm assumes known session boundaries and simply increases the weight of a topic as the evidence increases. If we wanted to track topic shifts, we would need some other measure using a sliding window.

5.2 n-gram Adaptation

As explained earlier, one of the inherent characteristics of language for dynamically adapting the model to the document observed thus far, is the topic of the conversation or document. There might also be fluctuations in word frequencies because of the speaking style of the person, or due to a specific topic within a general area. For example, although this work is based in the area of speech recognition, since it concentrates on language modeling, the word "language" is more prominent in usage as compared to "acoustic".

The change in the speaking style or the fluctuations in the word frequencies can be captured by using a dynamic cache model as suggested by [20, 21, 24]. There are two approaches possible here: either the partially observed document is cached and the cache is flushed between documents or if the document or article is reasonably long, a sliding window is used to determine the contents of the cache. After caching the document observed, an n-gram model is built on this cache. In our case, we selected the first approach of caching the entire partially observed document, because the Wall

Street Journal ARPA task does not have long articles. The cache probabilities are then interpolated with the static n-gram probabilities, with the interpolation weights obtained by minimizing the perplexity of an independent data set or maximizing the likelihood of unobserved data. In the case of our mixture model, an additional feature is that caches can be maintained for the different component models, or alternatively one "cache" can be maintained with separate component-dependent counts.

The important issues to be resolved in dynamic language modeling with a cache are the definition of the cache and the mechanism for choosing the interpolation weights. Since there are a variety of approaches for these two problems, we will begin by outlining alternatives for the single model, ($m = 1$) language model, in Section 5.2.1 and describe the extension to the mixture language model in Section 5.2.2.

5.2.1 Single n-gram adaptation

For single component mixture model, namely with $m = 1$, the equation for the adapted mixture model, incorporating a cache component is:

$$P(w_1, \ldots, w_T) = \prod_{i=1}^{T} \begin{array}{l} (\theta^s P^s(w_i|w_{i-1}, w_{i-2}) \\ \quad +\theta^c P^c(w_i|w_{i-1}, w_{i-2})) \end{array} \tag{5.5}$$

where P^s represents the static model parameters, P^c represents the cache model parameters, and θ^s and θ^c represent the weights for the static and cache models respectively. In our case, the static model parameters can be expanded as,

$$\theta^s P^s(w_i|w_{i-1}, w_{i-2}) = \theta_G P_G(w_i|w_{i-1}, w_{i-2}) + \theta_{G'} P_{G'}(w_i|w_{i-1}, w_{i-2})) \tag{5.6}$$

where $P_G(\cdot|\cdot)$ is a general model trained on all the training data and $P_{G'}(\cdot|\cdot)$ is a general model, but trained on all the data from the verbalized punctuation training set. Since $\theta^c + \theta^s(\theta_{G'} + \theta_G) = 1$ and $\theta_{G'} + \theta_G = 1$, we have two free parameters, θ^c

which is the weight of the cache language model (optionally dynamic), and θ_G which is the weight of the NVP model which is static.

The initial values of θ_G and optionally θ^c are estimated by maximizing the likelihood of an unobserved training set using Equation 4.12. However, since we cannot assign a zero value to the cache weight, we assign it an arbitrarily small value and renormalize the other n-gram weight estimates.

We initially implemented the cache as a standard trigram model. The cache probabilities were computed using the back-off technique implemented in our baseline mixture model. The θ's were then re-estimated, using a cache built on the partially observed document and using Equation 4.12 on previously unseen data. In order to alleviate some of the sparse data problems in the trigram cache, a conditional bigram/trigram cache has been suggested by Rosenfeld [24]. Consider the last observed trigram (w_1, w_2, w_3). If the first two words in this trigram, namely (w_1, w_2) are a bigram cache hit (i.e. have been previously observed in the document as a bigram), a trigram cache probability is used, i.e. $P(w_3|w_1, w_2)$. Otherwise, if w_2 is a unigram cache hit, the bigram estimate $P(w_3|w_2)$ is used. In the case of the conditional bigram/trigram cache, the cache is weighted only during a cache hit and not otherwise.

In addition to a conditional bigram/trigram cache, Rosenfeld [24] has also indicated a rare word unigram cache as a significant cache component. The idea being that, observing a word usually elevates the probability of it being observed again within a short duration in the document. Since the unigram cache is not significantly useful for high frequency words such as A, THE, etc., the cache is built for rare words only. A word can be classified as "rare", based on a threshold frequency and the frequency of the word in a large training base. In our case, the threshold frequency was empirically assigned after observing the range of word counts for the

vocabulary over a large training base. We based another classification of a "rare" word on whether it was a content or a function word, reasoning that a rare function word will not provide significant information of the document to be observed while a frequent content word might. In the case of a selective unigram cache, in order to reflect the importance of the cache estimates as we observe more of the document, the cache weights θ^c can be increased at the end of every sentence using a function of the cache size. These weights are reset to the initial values at the beginning of a new document or session.

5.2.2 Mixture n-gram adaptation

We consider two approaches for extending n-gram adaptation to our $m > 1$ mixture model, in both cases maintaining separate counts for each topic, for all the component models, including the general model. A sentence is categorized as belonging to a particular component-cache based on the likelihood of the sentence as estimated by the component models. The likelihood of the sentence i belonging to a particular topic is given by Equation 5.3. In the first approach, the counts are determined by assigning a sentence to the most likely topic. In the second approach, fractional counts are assigned to each topic according to their relative likelihoods.

Thus, in the first approach, the topic that the sentence i belongs to is obtained as,

$$\begin{aligned} \hat{topic} &= \max_k P(sentence^i | topic_k) \\ &= \max_k z_k^i \end{aligned} \tag{5.7}$$

where k refers to the topic. The equation for the adapted mixture model incorporating

component-caches is:

$$P(w_1,\ldots,w_T) = \sum_{k=1,..,C,G} \lambda_k \prod_{i=1}^{T} \quad (\theta_k^s P_k^s(w_i|w_{i-1},w_{i-2}) + \theta_k^c P_k^c(w_i|w_{i-1},w_{i-2})) \tag{5.8}$$

where P_k^s represents the static model parameters of component k, P_k^c represents the cache model parameters of component k, and θ_k^s and θ_k^c represent the static model and cache model weights for the component k respectively. The static model parameters for the kth component can be expanded as for the single LM case:

$$\theta_k^s P_k^s(w_i|w_{i-1},w_{i-2}) = \theta_k P_k(w_i|w_{i-1},w_{i-2}) + \theta_k^{G'} P_{G'}(w_i|w_{i-1},w_{i-2})) \tag{5.9}$$

where $P_k(\cdot|\cdot)$ is the kth component model trained on the kth topic data and $P_{G'}(\cdot|\cdot)$ is a general model, as defined earlier, trained on all the data from the verbalized punctuation training set.

The independent weight estimation data set is labeled by topics and the mixture weights for the kth topic, static θ_k^c and θ_k are estimated on the corresponding kth subset of the data in the same manner as explained for the $m = 1$ model. Together with the static model, the cache is finally weighted by the probability of the topic for the sentence the word is observed in.

The second solution to the problem of maintaining mixture cache models was to assign each sentence to all the topic-caches weighted by the likelihood of the sentence belonging to that topic, namely z_k^i from Equation 5.3. This implies that we consider weighted counts of the n-grams as opposed to absolute counts, and this allows all the topic-models to take advantage of the partially observed document.

Since implementing an interpolated n-gram cache [20] in the framework of the mixture model, requires simple variations to our back-off n-gram model, we implemented an interpolated trigram model instead of a back-off trigram model. In

this cache model, the cache probability $P^c(w_i|w_{i-1}, w_{i-2})$ is estimated as a linearly interpolated sum of the trigram, bigram and unigram probabilities,

$$P^c(w_i|w_{i-1}, w_{i-2}) = \theta^c_{uni} P(w_i) + \theta^c_{bi} P(w_i|w_{i-1}) + \theta^c_{tri} P(w_i|w_{i-1}, w_{i-2}) \qquad (5.10)$$

where $\theta_{uni}, \theta_{bi}$ and θ_{tri} are the interpolation weights for the unigram, bigram and trigram estimates respectively. The probability estimates are simple relative frequencies based on the counts of the words in the cache. The interpolation weights are empirically estimated on the development test data. The weights are selected so as to minimize recognition error in a series of experiments.

In order that the cache component is effectively reflected in the sentence estimate, the λ's can also be re-estimated in the same manner as was suggested in Section 5.1. The important difference here is that $P_k(w_1, \ldots, w_{n_i})$ in Equation 5.3 is now a mixture of the static probability estimates and the dynamic cache estimates as has been explained in this section. In this case, the recursive solution is only an approximation of the ML solution.

During adaptation, the n-gram level and the sentence-level mixture weights, as well as the cache are reset at the end of a session. In the Wall Street Journal ARPA task that we experimented on, sessions are defined to mark the end of an article or the end of speech by a particular speaker. Since the article session boundaries were not available, we handmarked these sessions in both our development and evaluation test data.

5.3 Implementation Details

As pointed out earlier, one of the advantages of the sentence-level mixture model framework lies in the fact that other language modeling advances or new modeling concepts can be easily incorporated in the model. Though the dynamic

modeling techniques required simple variations in the baseline static mixture model at a theoretical level, there were other practical implementation problems to be dealt with.

One of the significant problems was to have the m (in this case 5) topic models on-line to compute the correct transcription probabilities. Here, since we are considering supervised adaptation, the correct transcription implies the original input utterance and not the hypothesized utterance. In addition, we required to re-estimate the n-gram level mixture weights using the component models and the cache models on the WSJ1 acoustic transcriptions. We developed our dynamic modeling software such that all static probabilities (the N-best and the weight estimation sentences) were pre-computed. This proved to be a big saving, both in terms of memory requirements as well as in terms of computation time for development of the dynamic language model, though this is not an option for real time speech recognition. Most of the experiments used the Stochastic Segment Model as the acoustic component for weight estimation, unless otherwise stated. In some of the experiments presented at the end of this chapter, we have combined all the knowledge sources at hand, which include the HMM scores and the Segmental Neural Net scores along with the BBN trigram language model scores.

5.4 Experiments

In this section, we discuss the experimental results for the various adaptation techniques investigated. The results obtained are task-dependent; in our case, we worked on the Wall Street Journal ARPA H2-P0 task, which is not specifically designed for language model adaptation and has short sessions. Average session length for this data is 3-4 sentences. However, since the language model adaptation test data

and acoustic scores were not available at the time, we used the above mentioned data set. Moreover, the session boundaries were not marked. Hence, we hand-labeled both the development and evaluation test sets into different topic sessions or articles. All the experiments with dynamic modeling techniques were supervised adaptation, where the correct sentence transcription is available for adaptation after recognition. In all the results reported in this section, no BBN scores are used.

5.4.1 Dynamic Mixture Weight Adaptation

We first experimented with dynamic topic adaptation, adapting the sentence-level mixture weights. The number of mixture components used was five. The initial sentence-level weights were computed by maximizing the likelihood of the WSJ1 acoustic transcriptions. The mixture weights were adapted using Equation 5.4 on the partially observed data. The mixture weights were reset to their initial values at the beginning of every session. The results obtained during the dynamic mixture adaptation are reported in the Table 5.1 given below.

Table 5.1: *Perplexity measurements and recognition results on the 1993 ARPA 5K development and evaluation test sets using dynamic adaptation of sentence-level mixture weights.*

LM	Perplexity		% Word Error	
	Dev	Eval	Dev	Eval
Baseline static model	73.79	62.87	7.0	5.7
5-mixture λ adapted BU LM	73.49	62.54	7.0	5.9

There was no significant decrease in perplexity or improvement in the recognition

error. Typically, the sentence-level weights favored the general model, and in many cases almost entirely based the likelihood of the sentence on the general model. A possible explanation for this problem could be that the general model has more robust estimates than the topic models, since it has almost twice the amount of training data as compared to the topic models. This problem may be rectified, with additional training data.

5.4.2 Dynamic Cache Adaptation

In all the experiments with cache adaptation, we used the session boundaries and the correct transcriptions after recognition. In addition, the sentence-level mixture weights are not dynamically adapted unless otherwise stated. This gives us a clear indication of the gain, if any, obtained from the dynamic cache adaptation. We tried several techniques for dynamic cache adaptation for both the single-component mixture model, $m = 1$, as well as the five-component, $m = 5$, mixture model. The results obtained from these experiments are summarized in Table 5.2 and Table 5.3.

It can be seen from the tables that the single-mixture component showed a greater improvement due to dynamic cache adaptation as compared to the five-mixture component. The results obtained are discussed below. Most of the conclusions hold for both the $m = 1$ and the $m = 5$ case.

We initially implemented a dynamic trigram cache built on the partially observed document. A separate cache was maintained for each topic. The trigram cache probabilities were interpolated with the component and VP model probabilities. The topic and VP n-gram mixture weights were estimated by maximizing the likelihood of an independent data set, namely the WSJ1 acoustic transcriptions. The cache was initially assigned an arbitrary small initial weight of 0.0001, and the n-gram mixture weights were normalized to sum to unity.

Table 5.2: *Perplexity and recognition results on the 1993 ARPA 5K development and evaluation test sets for a single component LM.*

LM	Perplexity		% Word Error	
	Dev	Eval	Dev	Eval
Baseline 1-mixture model	73.34	62.81	7.3	5.8
Conditional bigram/trigram	70.19	59.29	7.1	5.6
Interpolated trigram	70.29	58.54	7.1	5.6
Rare word unigram+dynamic θ^c	69.00	58.5	7.0	5.6
Content word unigram+dynamic θ^c	68.15	57.86	7.0	5.5
Interpolated+content word	66.03	55.21	7.0	5.5

During adaptation, the correct transcription was cached after scoring the N-best list for that transcription. For the five component mixture model, the sentence was first labeled according to Equation 5.7 and separate counts were maintained for each topic. In the case of the single component model, a cache was maintained for the general model.

In order to deal with this problem of sparse data in the caches, we implemented a conditional bigram/trigram cache as suggested by [24]. Instead of re-estimating the n-gram weights, we assigned a fixed weight of 0.09 to the cache, in the case of a bigram/trigram cache hit and 0.0 otherwise. The cache weight of 0.09 was decided empirically by reducing the recognition error on the development test set. We renormalized the mixture weights in the case of a cache hit.

Although there was a 4% reduction in perplexity, there was no significant improvement in recognition using the conditional bigram/trigram caches. In order to

Table 5.3: *Perplexity and recognition results on the 1993 ARPA 5K development and evaluation test sets for a five-component LM. The combined model comprises fractional counts instead of absolute counts*

LM	Perplexity		% Word Error	
	Dev	Eval	Dev	Eval
Baseline 5-mixture model	73.79	62.87	7.0	5.7
Conditional bigram/trigram	70.65	59.75	7.1	5.6
Interpolated trigram	70.43	58.58	7.0	5.6
Rare word unigram+dynamic θ^c	69.40	58.89	7.0	5.6
Content word unigram+dynamic θ^c	68.62	58.31	7.0	5.5
Interpolated+content word	66.13	55.39	7.0	5.5

estimate the extent of adaptation, we computed the number of times the cache was used, namely the number of times there was a cache hit. This number was only 9% in the case of the conditional bigram/trigram cache. Besides the dependence of the low cache hit rate on the small duration of the sessions, an interesting observation here was that the hit rate could have improved if the cache had been able to link certain words such as possessives or adjectives. For example, if "JAPAN" had been previously observed in the context, and if the new context observed comprised "JAPANESE", it was treated as an entirely new word. In addition, most of the cache hits were due to the frequent occurrence of function words (such as "A", "THE"). The conditional bigram/trigram cache, due to the backoff technique used, assigns a probability to a word pair, as long as each word has been previously observed as a unigram.

In order to compare the conditional bigram/trigram cache with an interpolated cache, we experimented with an interpolated cache model suggested by [20]. The

interpolation weights were 0.4, 0.5 and 0.1 for unigram, bigram and trigram cache probabilities respectively, experimentally obtained by [20]. The cache weight was a fixed weight of 0.09 as in the case of the conditional bigram/trigram cache. The recognition error increased using the interpolated cache. This was due to the high weight given to the unigram cache, which mainly benefited the frequent function words. After a few experiments, we decided upon interpolation weights of 0.0, 0.9 and 0.1 for the unigram, bigram and trigram cache respectively. The cache weight was 0.2 in case of a cache hit and 0.0 otherwise.

Since a rare-words unigram cache can also provide significant information about the short-term fluctuations in the word frequencies, we implemented a unigram cache of words whose static general model frequency was lower than a certain threshold. Observing that the more frequent words had a count of approximately 60,000 or more in the training base, we set the threshold count at 35,000. We experimented with other values and found that the most significant gain was obtained at this threshold count. The unigram cache weight was directly proportional to the size of the cache, beginning with a weight of 0.05. The cache weight saturated at 0.1.

Another classification of a "rare" word can also be based on whether a word is a content or a function word. Using this classification, we implemented a unigram cache for content words. The unigram cache weight was proportional to the size of the cache, beginning at 0.05 and saturating at 0.1. There were two advantages observed on using the unigram cache discounting function words. There was approximately a 7% decrease in perplexity and a slight improvement in the recognition accuracy on the evaluation test set. In addition, the percentage of cache hits increased to around 13% as compared to 4% using the rare word unigram cache.

On a closer analysis of the component-caches, we realized that one of the major problems with the dynamic caches was that most of the test utterances were assigned

to the general model. This can be attributed to the fact that the general model has at least twice the training data as compared to the topic models, hence comprising more robust n-gram estimates. We tried to rectify this problem and deal with sparse trigram caches by assigning each utterance to every topic component model, weighted by the relative likelihood of the sentence. The cache was again assigned a fixed weight of 0.09 in the case of a hit and zero otherwise. The conditional bigram/trigram cache with weighted counts gave a small improvement in perplexity and recognition error when compared to the conditional bigram/trigram cache with absolute counts.

We finally combined all the different dynamic components to examine the improvement in recognition accuracy and perplexity provided by language model adaptation. As a "best-case" experiment, we considered two caches at the n-gram level. One was a unigram cache of content words, and the second was an interpolated bigram/trigram cache. The unigram cache was weighted proportional to the size of the cache, beginning at 0.05 and saturating at 0.1. The interpolated bigram/trigram cache was weighted 0.2 in the case of a cache hit and zero otherwise. The interpolation weights remained the same at 0.9 for the bigram cache and 0.1 for the trigram cache. The caches stored fractional counts in the case of the $m = 5$ mixture model. In addition to the cache component, the sentence-level mixture weights were also adapted as described in Section 5.1. These results are reported at the end of Tables 5.2 and 5.3.

5.5 Summary

We experimented with two different dynamic adaptation techniques using a single component mixture model and a five component mixture model. The dynamic adaptation of the sentence-level mixture weights did not improve performance and

the dynamic n-gram adaptation gave only a small improvement, essentially due to the task at hand. Both the development and evaluation data had small articles for adaptation, typically comprising three sentences. This implied that before the mixture weights could adapt significantly to the topics, they were reset at the session boundaries. Another significant problem with the topic adaptation lay in the fact that the general model had more robust probability estimates as compared to the topic models. Due to this reason, most of the sentences were assigned to the general model cache.

The cache adaptation gave a significant 9-11% decrease in perplexity and slight improvement in recognition results on the evaluation test set. On observing the N-best list for each transcription, it could be seen that most of the differences between sentences in this list were function words inserted or deleted at different places. The cache adaptation in general resulted in improved probability estimates for content words that had already been correctly recognized. With the small sessions available, there was no significant cache adaptation. Another important reason for the low cache hit rate was the fact that the caches did not consider any links between words (e.g. "JAPAN" and "JAPANESE"), as explained earlier.

We observed that dynamic adaptation of a single n-gram gave slightly better performance than the static mixture, but there was no further improvement using dynamic adaptation of mixtures. One might conjecture that the mixture model will be more appropriate for short term speech events and dynamic language models better for longer documents. However, further data is needed to test this hypothesis empirically. These dynamic mixture model techniques might provide a significant improvement in recognition accuracy, if the sessions for adaptation are reasonably long and additional training data is available for the topic models.

We conclude this chapter with the results obtained by combining the "best case"

Table 5.4: *Perplexity measurement and recognition results on the 1993 ARPA 5K development and evaluation test (dynamic cache adaptation with content word unigram cache and interpolated bigram/trigram cache, dynamic sentence-level mixture weight adaptation, includes all knowledge sources).*

LM	Perplexity		% Word Error	
	Dev	Eval	Dev	Eval
Baseline 5-mixture BU LM	73.79	62.87	6.2	5.3
1-mixture adapted BU LM	66.02	55.21	6.3	5.2
5-mixture adapted BU LM	66.13	55.39	6.1	5.2

dynamically adapted mixture model with the HMM, SNN and the BBN LM scores in addition to the SSM scores. With the adaptation, we achieve the best reported BU results for the task, 5.2% word error, as shown in Table 5.4.

Chapter 6

Conclusion

In this thesis, the basic concepts for a topic dependent sentence-level mixture model have been developed, which can be used with a speech recognition system to provide significant improvement in recognition accuracy. We have also investigated methods of dynamically adapting the mixture model to the partial document observed. The key results and contributions are summarized in the next section, followed by a discussion of possible directions for future work.

6.1 Summary

This thesis presents a new approach to statistical language modeling, which offers the potential for capturing both topic-dependent effects and long range sentence level effects in a conceptually simple variation of statistical n-gram language models. The research has theoretical contributions (development of a new language model with associated training and recognition algorithms), as well as practical insights gained through implementation and experiment. These contributions are briefly reviewed below, beginning with the general modeling contributions.

Since topic-dependent effects in language cannot be captured by the standard n-gram language models, we introduced a mixture of topic-dependent models as an alternative to the standard n-gram model. Though this work is similar to the work done by [22], the most important difference between the two approaches is that we try to capture the topic-dependent effects at the sentence level, since there will not be a shift in the topic within a sentence.

To implement a topic-dependent mixture model, we needed different topic-labeled data. An automatic clustering algorithm was developed, which could automatically classify the text as belonging to one of m topics. This helped us take advantage of the large Wall Street Journal training corpus, which did not have topic demarcations.

We investigated several techniques of robust parameter estimation, which are extremely important in the light of the sparse data problems in language modeling. We initially used an iterative Viterbi-style estimation procedure for estimating the n-gram probabilities of the topic-dependent models. In addition, we introduced a general model, trained on all the data, as one of the mixture components in order to smooth the topic-dependent estimates. The weights for the the mixture components were estimated by maximizing the likelihood of unseen data. Realizing that the Expectation-Maximization (EM) algorithm [17] might be a more powerful method for estimating the topic-dependent model parameters, we developed an EM algorithm for n-gram probability estimation combined with a Witten-Bell type back-off, which helped us reduce the effect of sparse data for the topic models. The advantage of using the EM algorithm lies primarily in the fact that the topic n-gram probabilities can be estimated over the entire training data, which is also reflected in improved recognition accuracy.

In order to provide additional smoothing for the topic models, as well as to account for the verbalized punctuation words, we introduced a general model trained

on verbalized punctuation data at the n-gram level. The n-gram level mixture weights were also estimated by maximizing the likelihood of unobserved data.

Dynamic language model adaptation, which makes use of the previous document history to tune the language model to that particular topic, easily fits into the mixture model framework in two ways. First, the sentence-level mixture weights can be recursively adapted according to the likelihood of the respective mixture components in the previous utterance, as in [22] for n-gram level mixture weights. Second, the dynamic n-gram cache model [20, 21] can be incorporated into the mixture language model. However, in the mixture model, it is possible to have component-dependent cache models, where each component cache would be updated after each sentence according to the likelihood of that component given the recognized word string. Alternatively, we can also have component-dependent fractional counts for each sentence in the caches. We also introduced a new variation of cache language modeling, where we use a selective unigram cache including only content words. This unigram cache was used alone and in addition to an interpolated bigram/trigram cache at the n-gram level.

Besides the theoretical contributions, there are important experimental contributions of this thesis. The experiments include our work with the different back-off techniques, the two similarity measures for clustering, and the two robust parameter re-estimation techniques. We also investigated different dynamic language modeling techniques and these contributions will be discussed next.

As an introduction to our work in the area of language modeling, we developed the software required to implement the standard n-gram language models, using the Katz back-off technique. Since we did not consider the Katz back-off technique to be extremely robust, we implemented the back-off technique suggested by [4]. We observed that a consistent improvement in recognition accuracy was obtained with

the second technique, in addition to faster computations of the back-off probabilities. This initial work proved to be extremely useful, since the mixture language model was built on this foundation.

We implemented two different similarity measures for clustering and observed that there was no significant improvement in the recognition accuracy. This clustering algorithm can also be made use of in other areas of language processing such as word-spotting or text-classification. The number of topics was restricted to 5 or 8, since the greater the number of fragments the training data was split into, the greater were the sparse data problems encountered. We investigated several techniques of robust parameter estimation of the mixture model, which are extremely important in light of the sparse data problems in language modeling, and found a small improvement in performance of the EM algorithm over the Viterbi algorithm.

The static mixture language model comprises the following details: a five component-mixture model, automatic clustering using the inverse document frequencies [40] as the similarity measure for clustering, parameter estimation of the topic n-gram probabilities using the EM algorithm, mixture weights estimated on unseen data, namely WSJ1 speaker independent speech transcriptions. The results for our baseline static mixture model and dynamically adapted mixture model that are reported below in Table 6.1 and Table 6.2, are based on the Wall Street Journal ARPA H2-P0 task [42]. These experiments mainly use only the SSM acoustic scores and in this case improved performance by 7%. Combining the SSM acoustic scores with the HMM scores, the SNN scores, and the static BU and BBN language models gives us a best case result of 5.3% which is close to the best reported result of 4.9% as obtained by Cambridge University (HTK group) using a standard trigram language model.

We experimented with different techniques of incorporating the cache in the mixture model, including conditional bigram/trigram cache, interpolated cache, selective

Table 6.1: *Recognition results for static language models: 1993 ARPA 5K development and evaluation test sets.*

Acoustic Model	Language Model	% Word Error	
		Dev	Eval
SSM	Trigram (BBN)	7.4	6.1
SSM	(5-mixture) trigram (BU)	7.0	5.7
SSM	BBN+BU	6.9	5.7
All	All + BU + BBN	6.3	5.3

unigram cache with rare words and selective unigram cache with content words. Of these, the selective content word unigram cache gave the most benefit for adaptation with short sessions. Dynamic language modeling provided small improvements in the recognition accuracy and about 9-11% decrease in the test set perplexity. The results using a dynamic language model are reported in Table 6.2. Here the model includes both sentence-level weight adaptation as well as component-cache models, namely an unigram cache for the content words and an interpolated bigram trigram cache. In the case of the five-component mixture model, each sentence is assigned to all the topics according to the likelihood of the sentence belonging to that topic. Besides the SSM scores, we also combine the HMM, SNN and the BBN language model scores to get our best result of 5.2% word error.

As a part of the research undertaken, we have also developed our own optimization techniques to reduce the heavy computation required for training and accessing the trigram models as well as to reduce the memory requirements of the model as the length of the vocabulary and the amount of training data increases.

Table 6.2: *Perplexity measurement and recognition results on the 1993 ARPA 5K development and evaluation test (dynamic cache adaptation with content word unigram cache and interpolated bigram/trigram cache, dynamic sentence-level mixture weight adaptation.)*

Acoustic Model	Language Model	Perplexity		% Word Error	
		Dev	Eval	Dev	Eval
SSM	Baseline 5-mixture model	73.79	62.87	7.0	5.7
SSM	1-mixture adapted BU LM	66.02	55.21	7.0	5.6
SSM	5-mixture adapted BU LM	66.13	55.39	7.0	5.6
All	5-mixture adapted BU LM	66.13	55.39	6.1	5.2

6.2 Suggestions for Future Work

This work can be extended in several ways. It would be interesting to see whether further performance gains can be achieved by increasing the number of clusters. Although we did not observe such an improvement in our experiments, the results may have been limited by the amount of available training data and the question should be re-evaluated since additional training data has been recently made available. It might also be useful to consider other metrics for automatic topic clustering, such as a multinomial distribution assumption with a likelihood clustering criterion, for obtaining more constrained topic models.

Much more could also be done in the area of robust parameter estimation. For example, one could use an n-gram part-of-speech sequence model as the base for all component models and topic-dependent word likelihoods given the part-of-speech

label, a natural extension of [21]. This approach would address the problem of sparse data, since it is likely that the part-of-speech sequence probabilities are not topic-dependent and can be based on the entire training data. This would imply that we model only the conditional probability of the word given the class (or given the class and the previous word) based on topics. If we use a dynamic model in this framework, we could maintain only component caches for the probabilities of the words, given part of speech as in [21].

We would also like to repeat our efforts using a much larger vocabulary, for example a 20K dictionary. With the additional training data of 2.0 GB, we should be able to exploit the full potential of the mixture model framework, with a larger number of more robust topic models.

The simple static mixture language model can also be useful in applications other than continuous speech transcription. For example, topic-dependent models could be used for topic spotting. In addition, as mentioned earlier, the notion of topic need not be related to subject area, it can be related to speaking style or speaker goal. In the ATIS task, for example, the goal of the speaker (e.g. flight information request, response clarification, error correction) is likely to be reflected in the language of the utterance. Representing this structure explicitly has the double benefit of improving recognition performance and providing information for a dialog model.

From a cursory look at our recognition errors from the recent WSJ benchmark tests, it is clear that topic-dependent models will not be enough to dramatically reduce word error rate. Out-of-vocabulary words and function words also represent a major source of errors. The mixture language model can be extended to incorporate out-of-vocabulary words in supervised adaptation contexts with cache modeling. This list of extensions is only partial, since an important advantage of this framework is that it is a simple modification of existing language modeling techniques that can

easily be integrated with other language modeling advances.

Appendix A

Expectation Maximization Algorithm

Here, we explain the Expectation Maximization algorithm that we developed for robust model parameter re-estimation as referred in Section 4.2.1.

The E-step

Let y_i be an observation, namely the sentence i, X_i represent the complete observation, i.e. y_i and the class it belongs to, and m represent the number of classes. Define,

$$X_i = \begin{bmatrix} y_i \\ z^k \end{bmatrix}$$

where

$$z^k = \begin{bmatrix} 0 & 0 & 1 & \cdots & 0 \end{bmatrix}$$

is an m-dimensional vector with a 1 in the k^{th} place, referring to the kth class. Let θ_1 represent all the parameters of the m classes and θ_2 represent the class probability

distributions. Define,

$$u(y_i|z^k) = \log P^{\theta_1}(y_i|c_k)$$

and

$$v(z^k) = \log P^{\theta_2}(c_k) = \log q_k$$

where c_k represents the kth class and q_k represents the probability of the kth class. The class probabilities sum to one, namely

$$\sum_{k=1}^{m} q_k = 1.$$

The log probability of a single complete observation X_i, given θ_1 and θ_2, can then be written as,

$$\begin{aligned} \log P(X_i|\theta_1, \theta_2) &= \log P^{\theta_1,\theta_2}(y_i, z^k) \\ &= \log P^{\theta_1}(y_i|z^k) + \log P^{\theta_2}(z^k) \\ &= z^k(U(y_i|\theta_1) + V(\theta_2)). \end{aligned}$$

where $U(y_i|\theta_1)$ is an m-dimensional vector with elements $u(y_i|z^j)$ and $V(\theta_2)$ is a vector with elements $v(z^j)$, $j = 1, \ldots, m$ which are defined earlier. Assuming every sentence is an identical and independent distributed observation, the likelihood of the entire set of n sentences, denoted by χ, can be given as,

$$\log P(\chi|\theta_1, \theta_2) = \sum_{i=1}^{n} \log P(X_i|\theta_1, \theta_2). \tag{A.1}$$

If p represents the current estimate, let $Q(\theta|\theta^{(p)})$ be the expected likelihood of the observations. Then,

$$Q(\theta|\theta^{(p)}) = E[\log P(\chi|\theta_1, \theta_2)|Y, \theta_1^{(p)}, \theta_2^{(p)}] \tag{A.2}$$

$$= \sum_{i=1}^{n} E[z^k(U(y_i|\theta_1) + V(\theta_2))|y_i, \theta_1^{(p)}, \theta_2^{(p)}]. \tag{A.3}$$

Since the data is incomplete because we do not know what class z^k the observation y_i belongs to, the expectation is over z^k. Therefore,

$$Q(\theta|\theta^{(p)}) = \sum_{i=1}^{n} E[z^k|y_i, \theta_1^{(p)}, \theta_2^{(p)}](U(y_i|\theta_1) + V(\theta_2)). \quad (A.4)$$

Let us call the expected class vector for the observation y_i as $\hat{z}_i$, which is an m-dimensional vector. A single element of this vector is,

$$\hat{z}_{ik}^{(p)} = P(z_k|y_i, \theta_1^{(p)}, \theta_2^{(p)}) \quad (A.5)$$

$$= \frac{P^{(p)}(y_i|z_k)P^{(p)}(z_k)}{\sum_{j=1}^{m} P^{(p)}(y_i|z_j)P^{(p)}(z_j)} \quad (A.6)$$

where $P^{(p)}(y_i|z_k)$ represents the probability of the observation y_i as given by the class c_k using parameters $\theta_1^{(p)}$ and $P^{(p)}(z_k)$ similarly uses $\theta_2^{(p)}$. Here, the probability of the sentence y_i, which comprises the sequence of words $w_1, w_2, \ldots, w_l$, using a trigram model k is given as,

$$P^{(p)}(y_i|z_k) = P_k^{(p)}(w_1)P_k^{(p)}(w_2|w_1)\prod_{q=3}^{l} P_k^{(p)}(w_q|w_{q-1}, w_{q-2}). \quad (A.7)$$

The initial estimates of the probabilities are obtained using labeled data, where the labels are given by clustering.

The M-step

We can maximize Q for θ_1, θ_2 separately to get the new $(p+1)$ parameters. Maximizing for θ_2 gives us [17],

$$q_j^{(p+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ij}^{(p)}}{\sum_{i=1}^{n}\sum_{k=1}^{m} \hat{z}_{ik}^{(p)}}. \quad (A.8)$$

Consider the trigram estimate for the trigram w_b, w_c, w_d for the j^{th} class, namely $P_j(w_d|w_c, w_b)$. Let $P_j(w_d|w_c, w_b) = \theta_{bcd}^j$. To re-estimate θ_{bcd}^j,

$$\underset{\theta_{bcd}^j}{\text{argmax}}\, Q = \underset{\theta_{bcd}^j}{\text{argmax}} \sum_{i=1}^{n}\sum_{k=1}^{m} \hat{z}_{ik}^{(p)} u(y_i|z^k).$$

Since we are considering only topic j,

$$\underset{\theta^j_{bcd}}{\operatorname{argmax}} Q \quad = \quad \underset{\theta^j_{bcd}}{\operatorname{argmax}} \sum_{i=1}^{n} \hat{z}^{(p)}_{ij} \log P(y_i|z^j) \tag{A.9}$$

where $P(y_i|c_j)$ = probability of sentence y_i as given by the j^{th} class. From Equation A.7,

$$\log P(y_i|c_j) \quad = \quad C + n^i_{bcd} \log \theta^j_{bcd}$$

where n^i_{bcd} is the number of occurrences of the trigram w_b, w_c, w_d in the sentence i and C is a constant that includes the remaining probability terms. Let γ_1 represent a Lagrange multiplier required to satisfy the constraint that all the trigram conditional probabilities observed in the context of w_c, w_b along with the backoff estimate of the bigram w_c, w_b, referred as ϕ^j_{bc}, should sum to unity, and γ_2 represent the Lagrange multiplier required to satisfy the constraint that the back-off estimate ϕ^j_{bc}, in the jth model should be greater than a certain minimum. The first constraint can be given as,

$$\sum_q \theta^j_{bcq} + \phi^j_{bc} = 1. \tag{A.10}$$

where q represents all the words observed in the training context w_c, w_b. In order to understand the second constraint, consider the back-off estimate for a single n-gram (or known topic structure). If $c(w_b, w_c)$ represents the bigram w_b, w_c count, $c(w_b, w_c, w_q)$ represents the trigram w_b, w_c, w_q count and $r(w_b, w_c)$ represents the unique trigrams observed in the bigram context, the probability $P(w_q|w_c, w_b)$ is estimated by the Witten-Bell backoff technique [15] as,

$$\frac{c(w_b, w_c, w_q)}{c(w_b, w_c) + r(w_b, w_c)}$$

and the bigram backoff is estimated as,

$$\frac{r(w_b, w_c)}{c(w_b, w_c) + r(w_b, w_c)}.$$

In an analogous manner, for EM estimation, let us define the "count" of w_b, w_c, w_q in topic j, as $\sum_{i=1}^{n} \hat{z}_{ij}^{(p)} n_{bcq}^{i}$ and the "unique trigram count" for bigram w_b, w_c as $\frac{\sum_{i=1}^{n} \hat{z}_{ij}^{(p)} n_{bcq}^{i}}{\sum_{i=1}^{n} n_{bcq}^{i}}$. Then the back-off constraint can be given as,

$$\phi_{bc}^{j} \geq \frac{R_{bc}^{j}}{R_{bc}^{j} + N_{bc}^{j}} \tag{A.11}$$

where,

$$N_{bc}^{j} = \sum_{q} \sum_{i=1}^{n} \hat{z}_{ij}^{(p)} n_{bcq}^{i}, \tag{A.12}$$

$$R_{bc}^{j} = \sum_{q} \left(\frac{\sum_{i=1}^{n} \hat{z}_{ij}^{(p)} n_{bcq}^{i}}{\sum_{i=1}^{n} n_{bcq}^{i}} \right). \tag{A.13}$$

Including these two constraints in the maximizing step, we have,

$$Q = \underset{\theta_{bcd}^{j}}{\operatorname{argmax}} \sum_{i=1}^{n} \hat{z}_{ij}^{(p)} (C + n_{bcd}^{i} \log \theta_{bcd}^{j}) + \gamma_1 (1 - \sum_{q} \theta_{bcq}^{j} - \phi_{bc}^{j}) + \gamma_2 (f_j(\phi_{bc}^{j})). \tag{A.14}$$

Without this constraint Equation A.11 MLE will cause ϕ_{bc}^{j} to be equal to 0. To maximize Equation A.14, we assign ϕ_{bc}^{j} the minimum possible value, namely

$$\phi_{bc}^{j} = \frac{R_{bc}^{j}}{R_{bc}^{j} + N_{bc}^{j}}. \tag{A.15}$$

Differentiating Q and equating to 0 for estimating θ_{bcd}^{j}, we obtain,

$$\sum_{i=1}^{n} \frac{\hat{z}_{ij}^{(p)} n_{bcd}^{i}}{\theta_{bcd}^{j}} - \gamma_1 = 0$$

Therefore,

$$\theta_{bcd}^{j} = \frac{\sum_{i=1}^{n} \hat{z}_{ij}^{(p)} n_{bcd}^{i}}{\gamma_1}.$$

Now substituting for θ_{bcq}^{j} in Equation A.10,

$$\begin{aligned} \gamma_1 &= \frac{\sum_{q} \sum_{i=1}^{n} n_{bcq}^{i} \hat{z}_{ij}^{(p)}}{1 - \phi_{bc}^{j}} \\ &= \frac{N_{bc}^{j} (R_{bc}^{j} + N_{bc}^{j})}{N_{bc}^{j}} \\ &= R_{bc}^{j} + N_{bc}^{j} \end{aligned}$$

where q represents all the words observed in the context w_b, w_c over the entire training data. Solving for γ_1 and θ^j_{bcd}, we obtain,

$$\theta^j_{bcd} = \frac{\sum_{i=1}^{n} \hat{z}^{(p)}_{ij} n^i_{bcd}}{N^j_{bc} + R^j_{bc}}. \tag{A.16}$$

From Equation A.12 and Equation A.13 we have,

$$\theta^j_{bcd} = \frac{\sum_{i=1}^{n} \hat{z}^{(p)}_{ij} n^i_{bcd}}{\sum_q \sum_{i=1}^{n} n^i_{bcq} \hat{z}^{(p)}_{ij} + \sum_q \frac{\sum_{i=1}^{n} n^i_{bcq} \hat{z}^{(p)}_{ij}}{\sum_{i=1}^{n} n^i_{bcq}}}. \tag{A.17}$$

and

$$P^{backoff}_j(w_b, w_c) = \frac{\sum_q \frac{\sum_{i=1}^{n} n^i_{bcq} \hat{z}^{(p)}_{ij}}{\sum_{i=1}^{n} n^i_{bcq}}}{\sum_q \sum_{i=1}^{n} n^i_{bcq} \hat{z}^{(p)}_{ij} + \sum_q \frac{\sum_{i=1}^{n} n^i_{bcq} \hat{z}^{(p)}_{ij}}{\sum_{i=1}^{n} n^i_{bcq}}}. \tag{A.18}$$

The above equation holds good for all trigram estimates. The bigram probabilities can similarly be estimated as,

$$\theta^j_{bc} = \frac{\sum_{i=1}^{n} \hat{z}^{(p)}_{ij} n^i_{bc}}{\sum_q \sum_{i=1}^{n} n^i_{bq} \hat{z}^{(p)}_{ij} + \sum_q \frac{\sum_{i=1}^{n} n^i_{bq} \hat{z}^{(p)}_{ij}}{\sum_{i=1}^{n} n^i_{bq}}}. \tag{A.19}$$

The unigram backoff is estimated as,

$$P^{backoff}_j(w_b) = \frac{\sum_q \frac{\sum_{i=1}^{n} n^i_{bq} \hat{z}^{(p)}_{ij}}{\sum_{i=1}^{n} n^i_{bq}}}{\sum_q \sum_{i=1}^{n} n^i_{bq} \hat{z}^{(p)}_{ij} + \sum_q \frac{\sum_{i=1}^{n} n^i_{bq} \hat{z}^{(p)}_{ij}}{\sum_{i=1}^{n} n^i_{bq}}}. \tag{A.20}$$

The unigram probability is estimated as,

$$\theta^j_b = \frac{\sum_{i=1}^{n} \hat{z}^{(p)}_{ij} n^i_b}{\sum_q \sum_{i=1}^{n} n^i_q \hat{z}^{(p)}_{ij}}. \tag{A.21}$$

Bibliography

[1] F. Jelinek, R. L. Mercer and S. Roukos, "Principles of Lexical Language Modeling for Speech Recognition", from *Readings in Speech Recognition*, edited by A. Waibel and Kai-Fu Lee, pp. 651-699, Morgan Kaufmann Publishers, 1990.

[2] L. R. Bahl, F. Jelinek and R. L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 179-190, Vol. PAMI-5, No. 2, March 1983.

[3] G. Gazdar and C. Mellish, "*Natural Language Processing in Lisp- An Introduction to Computational Linguistics*", Addison-Wellesly, 1989.

[4] P. Placeway, R. Schwartz, P. Fung and L. Nguyen "Estimation of Powerful LM from Small and Large Corpora", *Proc. Int'l. Conf. on Acoust., Speech and Signal Proc.,* pp. 33-36 Vol. 2, April 1993.

[5] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proc. of IEEE,* Vol. 77, No. 2, pp. 257-285, February 1989.

[6] V. Zue and D. Goddeau, "Integrating Probabilistic LR Parsing into Speech Understanding Systems", *Proc. Int'l. Conf. on Acoust., Speech and Signal Proc.,* Vol. I, pp. 181-184, 1992.

[7] K. Lari and S. J. Young, " The Estimation of Stochastic Context Free Grammars Using the Inside Outside Algorithm", *Computer Speech and Language,* pp. 35-36, 1990.

[8] J. Kupiec, "Hidden Markov Estimation for Unrestricted Stochastic Context Free Grammars", *Proc. Int'l. Conf. on Acoust., Speech and Signal Proc.,* Vol. I, pp. 177-180, 1992.

[9] J. H. Wright, G. J. F. Jones and E. N. Wrigley, "Hybrid Grammar Bigram Speech Recognizer System with First-Order Dependence Model", *Proc. Int'l. Conf. on Acoust., Speech and Signal Proc.,* Vol. I, pp. 169-171, 1992.

[10] J. Lafferty, "Integrating Probabilistic Finite-State and Context-Free Models of Language", presentation at the IEEE ASR Workshop, December 1993.

[11] M. Meteer and J. R. Rohlicek, "Statistical Language Modeling Combining N-gram and Context Free Grammars", *Proc. Int'l. Conf. on Acoust., Speech and Signal Proc.,* Vol. II, pp. 37-40, 1993.

[12] M. Ferretti, G. Maltese and S. Scarci, "Language Model and Acoustic Model Information in Probabilistic Speech Recognition", *IEEE Transactions on Speech and Audio Proc.,* pp. 707-710, February 1989.

[13] H. Ney and U. Essen, "On Smoothing Techniques for Bigram-Based Natural Language Modeling", *Proc. Int'l. Conf. on Acoust., Speech and Signal Proc.,* pp. 825-828, 1991.

[14] S. M. Katz, "Estimation of Probabilities from Sparse Data for the LM Component of a Speech Recognizer", *IEEE Transactions on Acoust., Speech, and Signal Proc.,* Vol. ASSP-35, Number 3, pp. 400-401, March 1987.

[15] H. Witten and T. C. Bell, "The Zero Frequency Estimation of Probabilities of Novel Events in Adaptive Text Compression", *IEEE Transactions Inform. Theory*, Vol. IT3-7, No. 4, pp. 1085-1094, 1991.

[16] F. Jelinek and R. L. Mercer, "Interpolation and Estimation of Markov Source Parameters from Sparse Data", *Pattern Recognition In Practice*, ed. E. S. Gelsema, L. N. Kanal, 1981.

[17] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society (B)*, Vol. 39, No. 1, pp. 1-22, 1977.

[18] J. Kupiec, "Probabilistic Models of Short and Long Distance Word Dependencies in Running Text", *Proc. ARPA Workshop on Speech and Natural Language,* pp. 290-295, Feb 1989.

[19] R. Kuhn and R. de Mori, "Some Results on Stochastic Language Modeling", *Proc. ARPA Workshop on Speech and Natural Language,* pp. 225-229, Feb 1991.

[20] F. Jelinek, B. Merialdo, S. Roukos and M. Strauss, "A Dynamic LM for Speech Recognition", *Proc. ARPA Workshop on Speech and Natural Language,* pp. 293-295, 1991.

[21] R. Kuhn and R. de Mori, "A Cache Based Natural Language Model for Speech Recognition", *IEEE Transactions PAMI,* Vol. 14, pp. 570-583, 1992.

[22] R. Kneser and V. Steinbiss, "On the Dynamic Adaptation of Stochastic LM", *Proc. Int'l. Conf. on Acoust., Speech and Signal Proc.,* Vol. 2, pp. 586-589, 1993.

[23] R. Lau, R. Rosenfeld and S. Roukos, "Trigger-Based Language Models: a Maximum Entropy Approach", *Proc. Int'l. Conf. on Acoust., Speech and Signal Proc.,* Vol II, pp. 45-48, April 1993.

[24] R. Rosenfeld, "A Hybrid Approach To Adaptive Statistical Language Modeling", to appear in *Proc. ARPA Workshop on Human Language Technology,* March 1994.

[25] L. R. Bahl, P. F. Brown, P. V. deSouza and R. L. Mercer, "A Tree-Based Statistical Language Model for Natural Language Speech Recognition", *IEEE Transactions on Acoust., Speech, and Signal Proc.,* Vol. 37, No. 7, pp. 1001-1008, 1989.

[26] J. H. Wright, G. J. F. Jones and H. Lloyd-Thomas, "A Consolidated Language Model For Speech Recognition", *Proc. EuroSpeech,* Vol. 2, pp. 977-980, 1993.

[27] R. Isotani and S. Matsunaga, "Speech Recognition Using a Stochastic Language Model Integrating Local and Global Constraints", *Proc. ARPA Workshop on Human Language Technology,* 1994.

[28] M. Ostendorf and S. Roukos, "A Stochastic Segment Model for Phoneme-Based Continuous Speech Recognition", *IEEE Transactions on Acoust., Speech, and Signal Proc.,* , Vol. 37, No. 12, December 1989.

[29] S. Roucos, M. Ostendorf, H. Gish, and A. Derr, "Stochastic Segment Modeling Using the Estimate-Maximize Algorithm", *IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 127-130, New York, April 1988.

[30] O. Kimball, M. Ostendorf and I. Bechwati, "Context Modeling with the Stochastic Segment Model,"*IEEE Transactions Signal Processing*, Vol. ASSP-40(6), pp. 1584-1587, June 1992.

[31] A. Kannan, M. Ostendorf and J. R. Rohlicek, "Maximum Likelihood Clustering of Gaussians for Speech Recognition", *IEEE Transactions on Speech and Audio Processing*, Vol. 2, No. 3, pp. 453-455, July 1994.

[32] F. Kubala *et al.*, "BBN BYBLOS and HARC – February 1992 ATIS Benchmark Results", *Proc. ARPA Workshop on Speech and Natural Language,* February 1992.

[33] M. Bates *et al.*, "The BBN/HARC Spoken Language Understanding System", *Proc. Int'l. Conf. on Acoust., Speech and Signal Proc.,* Vol. II, pp. 111-114, 1993.

[34] S. Austin, J. Makhoul, R. Schwartz and G. Zavaliagkos, "Continuous Speech Recognition using Segmental Neural Nets," *Proc. of the DARPA Workshop on Speech and Natural Language,* pp. 249-252, Feb. 1991.

[35] M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz and J. R. Rohlicek, "Integration of Diverse Recognition Methodologies Through Reevaluation of N-Best Sentence Hypotheses", *Proc. ARPA Workshop on Speech and Natural Language,* pp. 83-87, February 1991.

[36] D. B. Paul and J. M. Baker, "The Design for the Wall Street Journal-based CSR Corpus", *Proc. ARPA Workshop on Speech and Natural Language,* pp. 357-361, February 1992.

[37] F. Kubala *et al.*, "The Hub and Spoke Paradigm for CSR Evaluation", *Proc. ARPA Workshop on Human Language Technology,* pp. 40-44, March 1994.

[38] R. Schwartz *et al.*, "On Using Written Language Training Data for Spoken Language Modeling", *Proc. ARPA Workshop on Human Language Technology,* March 1994.

[39] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis,* Wiley-Interscience, 1973.

[40] S. Sekine, "Automatic Sublanguage Identification for a New Text", *Second Annual Workshop on Very Large Corpora, Kyoto, Japan*, pp.109-120, August 1994.

[41] P. Brown, *The Acoustic Modeling Problem in Automatic Speech Recognition*, Carnegie Mellon University Ph.D. Dissertation, 1987.

[42] D. Pallett, J. Fiscus, W. Fisher, J. Garofolo, B. Lund and M. Pryzbocki, "1993 Benchmark Tests for the ARPA Spoken Language Program", *Proc. ARPA Workshop on Human Language Technology,* March 1994.