CONTINUOUS LANGUAGE MODEL ADAPTATION

Diploma Thesis

MICHALIS PAPAKOSTAS

Supervisor : Prof. Vassilis Digalakis

TECHNICAL UNIVERSITY OF CRETE
Chania, October 2013

Approval

Name:
Michalis Papakostas

Degree:
Diploma in Electronics and Computer Engineering

Thesis Title:
Continuous Language Model Adaptation

Examine Committee:

---

1. Dr. Vassilis Digalakis
   Professor at Technical University of Crete
   Supervisor

---

2. Dr. Michail Lagoudakis
   Associate Professor at Technical University of Crete

---

3. Dr. Michail Paterakis
   Professor at Technical University of Crete

Acknowledgments

Initially I would like to mention that I owe my gratitude to my professor Dr. Vasilis Digalakis for confiding this topic to me and for giving me the chance to concern myself with such an interesting part of science as Language Modeling.

Moreover I should thank my senior supervisor Dr. Vassilis Diakoloukas for being always inclined to help me and for providing me with his invaluable advises, his guidance and his great experience .

Finally I need to thank all my friends and my family for their emotional and mental support during the last year.

Michalis Papakostas

Contents

List of Tables

List of Figures

Abstract

Statistical Language Models (LMs) are widely used in many applications such as speech recognition systems, automatic translation systems etc.

However every statistical model is based to the domain of the training data and thus it can not perform well when tested in out-of-domain data. Moreover collecting and processing data in order to train a new statistical model is always a time consuming and expensive procedure given the large amounts required. Hence adaptation techniques have been developed in order to adapt an existing LM to a new domain, using significantly smaller amount of data.

N-grams, which is the dominant technology for language modeling, are very difficult to be adapted due to the large amounts of parameters. Thus LMs in Continuous-Space have been implemented in order to make language models more robust and easier to be adapted.

This study is an initial approach to continuous LM adaptation. We take advantage of some widely used algorithms from the field of speech recognition and we try to adapt an initial LM ,trained in a corpus from Wall Street Journal, with data from Air Travel Information System. We examined different approaches and techniques and came up to some useful conclusions which can feed many future works.

Thesis Outline

In the first section we introduce the part of Artificial Intelligence science, known as Natural Language Processing. We explain what is a statistical model in general, how such a model can be used in NLP and why nowadays investigating adaptation techniques for such models is thought to be a necessity. At the end of the chapter we report the most widely known adaptation techniques and we present the purpose of this work.

In section two, the differences between discrete and continuous language models are described. We analyze each approach and focalize on their benefits and drawbacks. Additionally the mechanism on which our work is based is being explained in detail. At the end of the section we describe how we evaluate our model, introducing terminologies from information theory, such as entropy and perplexity.

In section three we introduce the main adaptation algorithms. We explain why these techniques are so widely known and we decompose their mechanisms and their theoretical background.

In chapter four the results of our experimental work are illustrated. We describe our baseline experiments and the complete reasoning course until we reached our final goal. All the experimental tries are presented in great detail and supported with the appropriate charts and tables.

In the final chapter we summarize our best results, we make some crucial comparisons and we reach a final conclusion. At last we propose some suggestions for future work which I believe they worth experimenting on.

Chapter 1

Introducing Language Modeling and Statistical Model Adaptation

1.1 Introduction

Developing a speech recognition system for a new domain is costly, primarily due to the collection and preparation of the data required to train the system. Generally speaking, fairly large amounts of manually annotated data (tens of hours of data at a minimum for a large vocabulary system) are needed, which are very labor intensive to obtain.

Language model (LM) and acoustic model (AM) adaptation attempt to obtain models for a new domain with little training data, by leveraging existing (out-of-domain) models. AM adaptation in particular has been studied extensively, with many related researches and the achieved performances are already extremely high.In contrary to AM adaptation, LM adaptation has received much less attention and the potentials to improve our results are still great.

In this study we try to adapt a LM structured with continuous distributions by using well known adapting algorithms which are widely used in speech recognition. We use an initial LM trained in Wall Street Journal (WSJ) domain and we propose some adaptation approaches in order to construct an efficient LM on Airline Travel Information System (ATIS) domain using a small amount of adaptation data. Global and clustering methods have been tried and all our results are compared with baseline experiments made with discrete and continuous LMs as well.

1.2 Natural Language Processing (NLP)

Natural language processing is the technology for dealing with our most ubiquitous product: human language, as it appears in emails, web pages, tweets, product descriptions, newspaper stories, social media, and scientific articles, in thousands of languages and varieties. In the past decade, successful natural language processing applications have become part of our everyday experience, from spelling and grammar correction in word processors to machine translation on the web, from email spam detection to automatic question answering, from detecting people's opinions about products or services to extracting appointments from your email.

The development of NLP applications is challenging because computers traditionally require humans to speak to them in a programming language that is precise, unambiguous and highly structured or, perhaps through a limited number of clearly-enunciated voice commands. Human speech, however, is not always precise – it is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context.

Current approaches to NLP are based on machine learning, a type of artificial intelligence that examines and uses patterns in data to improve a program's own understanding. Most of the research being done on natural language processing revolves around search, especially enterprise search.

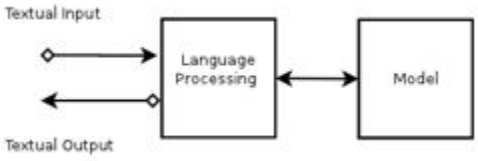

Figure 1.1: Language Processing

1.3 Statistical Modeling

A statistical model is a probability distribution constructed to enable inferences to be drawn or decisions made from data. This idea is the basis of most tools in the statistical workshop, in which it plays a central role by providing economical and insightful summaries of the information available.

The key feature of a statistical model is that variability is represented using probability distributions, which form the building-blocks from which the model is constructed. Typically it must accommodate both random and systematic variation. The randomness inherent in the probability distribution accounts for apparently haphazard scatter in the data, and systematic pattern is supposed to be generated by structure in the model. The art of modelling lies in nding a balance that enables the questions at hand to be answered or new ones posed. The complexity of the model will depend on the problem at hand and the answer required, so dierent models and analyses may be appropriate for a single set of data.

Statistical modeling can be applied in many applications. Language Modeling is exclusively based in statistical models and pattern matching. The fundamental idea is that Language Models assign probabilities to words of a word sequence using stochastic models.

1.3.1 Statistical Language Modeling

The goal of Statistical Language Modeling is to estimate the distribution of natural language as accurate as possible. A statistical language model (SLM) is a probability distribution P(s) over strings S that attempts to reflect how frequently a string S occurs as a sentence.

By expressing various language phenomena in terms of simple parameters in a statistical model, SLMs provide an easy way to deal with complex natural language in computer.

The original (and is still the most important) application of SLMs is speech recognition, but SLMs also play a vital role in various other natural language applications as diverse as machine translation, part-of-speech tagging, intelligent input method and Text To Speech system.

N-gram model is the most widely used SLM today. Without loss of generality we can express the probability of a string S , P(S) , as:

$$P(S) = P(w_1, w_2, \cdots, w_n) \tag{1.1}$$

$$P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\cdots P(w_k|w_1, w_2\cdots, w_{k-1}) = \tag{1.2}$$

$$\prod_{i=1}^{k} P(w_i|w_1, w_2, \cdots, w_{i-1}) \tag{1.3}$$

In bigram models, we make the approximation that the probability of a word only depends on the identity of the immediately preceding word, hence we can approximate P(S) as:

$$P(S) = \prod_{i}^{k} p(w_i|w_{i-1}) \tag{1.4}$$

If the word depends on more, than the previous two words, we can have a trigram: $P(w_i|w_{i-1}, w_{i-2})$, a fourgram $P(w_i|w_{i-1}, w_{i-2}, w_{i-3})$ etc. Similarly, we can have a unigram: $P(w_i)$ if the word does not depend on history words.

The parameters in a traditional N-gram model can be estimated with Maximum Likelihood Estimation (MLE) technique:

$$P(w_i|w_{i-1}) = \frac{Count(w_{i-1}, w_i)}{Count(w_{i-1})} \tag{1.5}$$

1.4 Adaptation Of Statistical Model

The generalization properties of most current statistical learning techniques are predicated on the assumption that the training data and test data come from the same underlying probability distribution.

Unfortunately, in many applications, this assumption is inaccurate. It is often the case that plentiful labeled data exists in one domain (or coming from one distribution), but one desires a statistical model that performs well on another related, but not identical domain. As every statistical model is being tuned to the domain of the training data and hand labeling data in the new domain is a costly procedure, adaptation techniques should be used before the model can be used to recognize data in a different domain.

For that reason statistical model adaptation techniques are becoming indispensable by helping us fit the distributions of training "in-domain" data to the "out-of-domain" test data using only a small amount of information from the new domain.

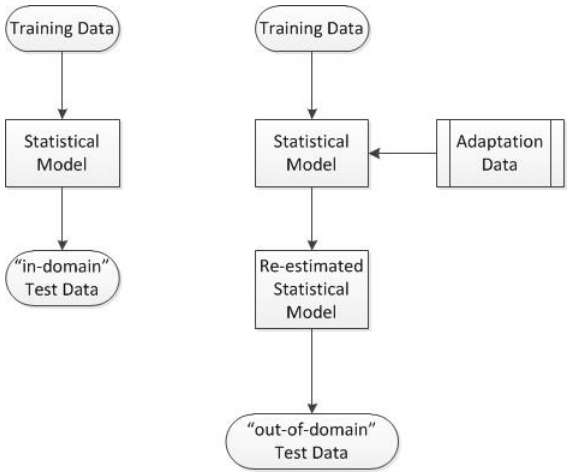

Figure 1.2: Statistical Model Adaptation

1.4.1 Adaptation Of Statistical Language Model

Regarding a SLM the definition of adaptation follows the same guidelines as mentioned before. Given a LM which is well trained in a specific domain (usually with more than 50.000 sentences) we try to reestimate the model's distribution in order to match the "out-of-domain" test data. The key here is that usually a much smaller amount of adaptation data is needed in order to achieve high efficiency (usually less than 20.000 sentences)

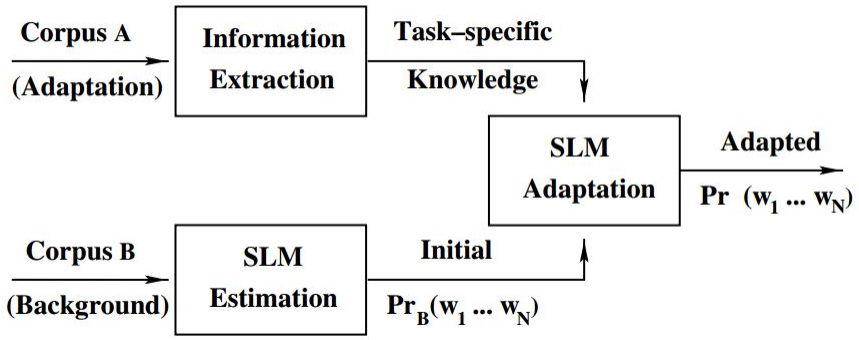

Figure 1.3: Statistical Language Model Adaptation [12]

The most widely techniques for adapting an N-Gram Language Model are the following [12]:

1. **Model Interpolation**

   In interpolation-based approaches, the corpus is used to drive a task-specific SLM, which is then combined with the background SLM.

   Model interpolation can be divided into three kinds:

   (a) **Model Merging**
   Because of the limited amount of adaptation data involved the adapted SLM tends to be poorly trained, most of the time resulting in a rather inaccurate estimate
   However for certain word sequences,particulary frequent in the current task, may the new SLM outperform the initial estimate SLM.
   The simplest way to apply Model Merging is via linear interpolation

   (b) **Dynamic Cache Models**
   A special case of linear interpolation, broadly used for within-domain adaptation, is commonly knows as: Dynamic Cache Memory Adaptation

   The underlying idea of this method is that a language model exploited short-term shifts in word-use frequencies might perform significantly.

   (c) **MAP Adaptation**
   In this approach, the MAP-optimal model M^* is computed as:

   $$M^* = argmax_M P(A|M)P(M) \tag{1.6}$$

   where *P(M)* is prior distribution over all models

   This framework leads to a solution of the following form:

   $$P(w_q|h_q) = \frac{\varepsilon \cdot Count_A(h_q w_q) + Count_B(h_q w_q)}{\varepsilon \cdot Count_A(h_q) + Count_B(h_q)} \tag{1.7}$$

   where ε is a constant factor which is estimated empirically

2. **Constraint Specification**

   In such approaches, the corpus is used to extract features that the adapted SLM is constrained to satisfy. This framework is thought to be more efficient

as a different weight is likely to be assigned separately for each feature.
Constraint Specification adaptation technique can be divided into three subcategories:

(a) **Exponential Model**
In this approach we train exponential models using the maximum entropy criterion.
Despite computing the conditional probability $P(w|h)$ directly we assume the associated event of the joint distribution and we consider that this joint distribution is constrained by K linearly independent constraints.
Finally we reach to a solution for the joint distribution $P(w|h)$ which has an exponential parametric form as shown bellow.

$$P(w|h) = \frac{1}{Z(h,w)} \exp^{\{\lambda_k I_k(h,w)\}} \tag{1.8}$$

(b) **MDI Adaptation**
MDI (Minimum discrimination information estimation) considers the features extracted from the adaptation corpus as important properties.
The solution has to be close to the joint initial distribution $P_B(h,w)$, taken as prior distribution. This is achieved by minimizing the Kullback-Leibler distance from the joint background distribution.
The final model has the following form:

$$P(h,w) = \frac{P_B(h,w)}{Z(h,w)} \prod_{k=1}^{K} \exp^{\{\lambda_k I_k(h,w)\}} \tag{1.9}$$

(c) **Unigram Constraints**
MDI adaptation is widely used with unigram constraints. Given a small amount of adaptation data we can use only unigram features to estimate a reliable adaptation model on corpus A.
The resulting solution reduces to the close form:

$$P(h,w) = P_B(h,w) \frac{a_A(w)}{P_B(w)} \tag{1.10}$$

where $a_A(w)$ represent the empirical unigram probability

3. Mixture Model

The simplest approach of Mixture modeling is based on a generalisation of linear interpolation to include several pre-defined domain

Assuming that the initial n-gram model is composed of a collection of K sub-models, each trained on a seperate topic, then a mixture LM linearly interpolates these K n-grams in such way that the resulting mixture best matches the adptation data.
The final probability has the form:

$$P(w_q|h_q) = \sum_{k=1}^{K} \lambda_{A,k} \cdot P_{B,k}(w_q|h_q) \tag{1.11}$$

where $\lambda_{A,k}$ refers to the interpolation coefficients and reflects the fact that they are estimated on corpus A.

We will focus in more detail on Mixture Language Models in the next chapters as they are the key tool in this search.

4. **Semantic Knowledge**

Approaches taking advantage of semantic knowledge usually purpose to exploit the entire semantic fabric of the adaptation corpus.
This approach is much more difficult to be specified as the semantic relation between two words is a very questionable subject and a lot of conflicting views can be found.

5. **Syntactic Infrastructure**

Approaches leveraging syntactic knowledge make the implicit assumption that the initial and the recognition tasks share a common grammatical infrastructure. The background SLM is then used for initial syntactic modeling, and the adaptation corpus to re-estimate the associated parameters.

6. **Multiple Sources**

Of course very popular is to combine more than one of the above techniques in order to achieve higher efficiency. The corpus is used to extract information about different aspects of the mismatch between training and recognition conditions.

1.4.2 Purpose Of the Thesis

In this work we tried to adapt a continuous-space Language Model into a new domain. This LM was created by Mr. Konstantinos Tsiakas (PhD Candidate at Demokritos - Greek Center For Scientific Research) as his Bachelor Thesis.

The initial models are trained with samples from Wall Street Journal (WSJ) domain, on a vocabulary of the 2700 most frequent words. Each in-vocabulary word has a corresponding trained Gaussian Mixture Model which represents the significance and the relation of this word with the others.

Our task is to re-estimate these background models by using a significantly smaller amount of adaptation data from Air Travel Information Service (ATIS) domain in order to match our new testing conditions.

In order to achieve our goal we exploited some widely used adaptation algorithms from speech recognition, such as MAP, MLLR and MLLR's implementation, Constrained MLLR. We examined each adaptation technique alone by applying various transformations to the background models and we worked with combinatory adaptation approaches as well.

Furthermore as we worked in continuous space, data reduction techniques needed to be used in order to save computational cost and to make data management easier without the loss of useful information. Hence we took advantage of techniques and methods for mapping and dimensionality reduction from the field of linear algebra and Pattern Recognition such as Singular Value Decomposition (SVD) and Linear Discriminant Analysis (LDA).

As this work is the first which targets to adapt a Continuous Language Model, we examined plenty of baseline experiments by trying to train both discrete and continuous models, with the same data used for adaptation but without performing adaptation techniques.

Finally we present the tools which were used to implement our baseline experiments, the word clustering and the GMMs parameters estimation which are SRILM tool and HTK toolkit.

Chapter 2

Building a Language Model in Continuous Space

2.1 Discrete Statistical Language Models

As mentioned before, statistical language models use statistical techniques to estimate models from text data sets by assigning probabilities in each word of the data. One of the most dominant technologies is N-gram models. N-gram models regard each word as a discrete variable. They are significant for many applications such as speech recognition, optical character recognition, machine translation, even dictation correction. Generally, the N-gram model has good results, when there is a satisfying set of data for a specific task.

2.1.1 Disadvantages

As covered before, N-gram models are the dominant technology in Language Modeling. In spite of their success, discrete N-gram models suffer from two basic drawbacks. We can refer to them as *generalization* and *adaptability*. These two problems refer to the N-grams with zero probability and to the parameters of N-gram model. Usually, an N-gram model has a huge number of parameters. Thus, it is very difficult to adapt it using a relatively small amount of data.

2.2 From Discrete To Continuous Language Models

Aiming to overcome the problems mentioned before and to make the adaptation of the final model easier and more efficient we prefer to build a Language Model using continuous-space distributions. Continuous representation of our training features needs an incredibly lower number of parameters to be estimated and that makes it much easier to adapt these parameters to fit to another-new distribution.

Moreover we overtake the need of having a trained model for every new word sequence we meet while testing. It is possible to ascribe the new-untrained word sequence in a class which concludes a variety of words which are for example semantically common and share the same distribution.

In the upcoming paragraphs we will discuss step by step how we manage to move from discrete word sequences to continuous space representation of each word and its history vectors. The following analysis is based on Mr.Tsiakas implementation and is the basis for the adaptation data editing as well.

2.2.1 Word Mapping

As is understood when talking about words means that we are dealing with discrete entities. Our purpose is to project each word of our data in a new continuous space. Due to the large number of words and the difficulty to process parameters in a high- dimensional space, we try to represent the *V-1* most frequent words from our train data. Adding to these words the $<$ *unk* $>$ tag which represents all the "out-of-vocabulary" words we create a model vocabulary with the V most frequent words.

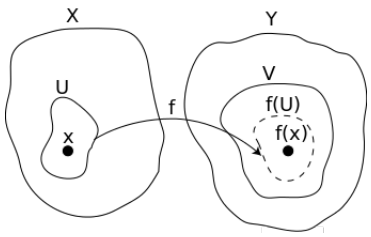

Figure 2.1: Mapping to another value domain

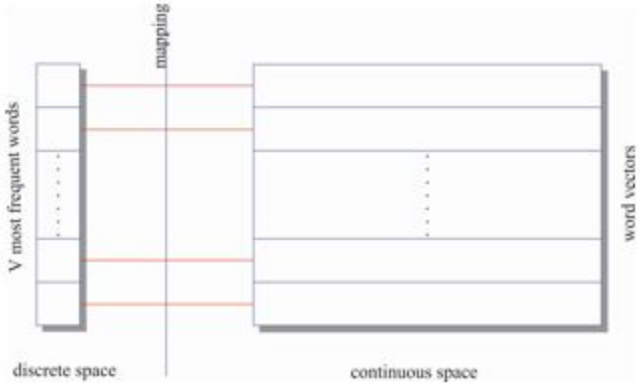

Figure 2.2: Word Mapping

2.2.2 Word Co-occurrences

The first step in order to achieve a reliable mapping for each word is to see how each word correlates with the rest. For that reason we build a co-occurrence matrix E that depicts this relation between words. Each element e_{ij} is the number of times that word i follows word j, in the training data. Each row is the word's vector and the columns represent the histories.

E_{ij}	V_1	V_2	V_3	V_4	V_5	....	V_V
V_1	0	0	0	92	34	....	57
V_2	10	0	0	0	12	....	0
V_3	0	0	0	37	53	....	156
V_4	12	78	0	0	64	....	0
V_5	0	35	118	0	0	....	745
....	....	....	....	....	....	....	....
V_V	45	65	0	64	0	....	0

Figure 2.3: Co-occurrence Matrix

As it can easily be observed this matrix consists of V^2 elements. This means that as the size of the vocabulary rises, the size of the matrix rises exponentially. Especially, in our initial model, where the vocabulary comprises 2700 words, the matrix E contains $2700x27000 = 729 \cdot 10^4$ elements. In order to save computing power and make our problem more easily manageable we try to map each vector to a lower dimension, concerning the word's frequency and how each word behaves with the others.

2.2.3 Singular Value Decomposition (SVD)

Singular value decomposition (SVD) can be looked at from three mutually compatible points of view. On the one hand, we can see it as a method for transforming correlated variables into a set of uncorrelated ones that better expose the various relationships among the original data items. At the same time, SVD is a method for identifying and ordering the dimensions along which data points exhibit the most variation. This ties in to the third way of viewing SVD, which is that once we have identified where the most variation is, it's possible to find the best approximation of the original data points using fewer dimensions. Hence, SVD can be seen as a method for data reduction.

The underlying idea is that performing SVD on a matrix A mxn lead us to decomposed form of A where:

$$A = U\Sigma V^T \tag{2.1}$$

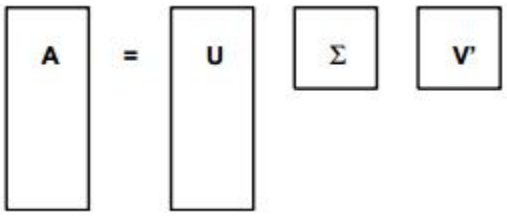

Figure 2.4: Singular Value Decomposition

U: mxm matrix with the eigenvalues of AA^T as its columns

V: nxn matrix with the eigenvectors of A^TA as its columns

Σ: mxn diagonal matrix consisting of the square roots of the eigenvalues of AA^T and A^TA (both matrices have the same eigenvalues but different eigenvectors)

Now notice matrix Σ, which contains the singular values. These singular values multiply only certain columns of U and V. The singular values determine how these columns of U and V influence the matrix. If a value is small enough then its column or row is not added to the new matrix. Many of the values are zero. That means that the corresponding columns and rows have not useful information. So we can transform the matrix to another one that has the valuable information.

Often, singular values are sorted by the significance of their rows and columns. That is the reason that SVD is used for matrix decomposition and data reduction. If we want to maintain the basic information of a matrix by reducing its dimensionality, we can keep the M greatest singular values and build another matrix Σ^T, containing only these M singular values. So we can obtain a transformation matrix Mxn.

2.2.4 History Mapping

Based on N-gram models, each word's history consists of the previous $N-1$ words (1.3.1) . Choosing the value of N depends on the model designer. If $N = 1$, each word is considered as an independent entity. For $N = 2$, each word depends on the previous one and if $N = 3$, each word depends on the previous 2 words. In natural language, it seems that each word has strong dependence on its previous two, so trigram models are trained in our case, for $N = 3$.

Following this approach we try to collect for each word in our vocabulary its $N-1$ previous. Replacing every history word with its SVD vector lead us to history vectors which are considered as the concatenation of the $N-1$ SVD history vectors.

After having the history vectors collected, we are able to model our parameters by training one Gaussian Mixture Model (GMM) for each word. However, as mentioned before, modeling becomes difficult in large dimensions. Assume that we have a trigram model ($N = 3$), a vocabulary of size $V = 3K$ and we perform SVD for $M = 100$ singular values. That means that each history consists of a $100 \cdot (N-1) = 200$ elements.This is still a high-dimensional space to handle.

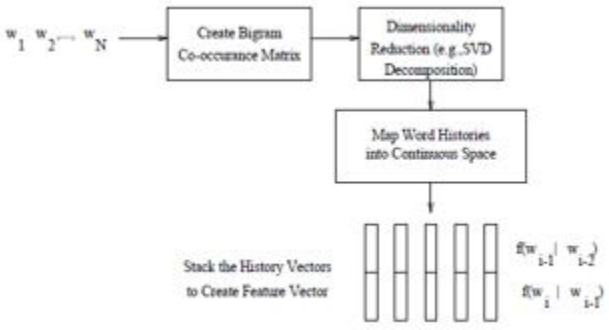

Figure 2.5: History Mapping

Aiming to make our history vectors even smaller we decide to apply to them a linear transformation of the form:

$$y_i = B \cdot h_i \tag{2.2}$$

where y_i is the projection of the history vector h_i in the new- dimensional space. The estimation of the transformation matrix B is described right below.

2.2.5 Linear Discriminant Analysis (LDA)

Feature reduction uses statistical methods to reduce the dimension of the features, while maximizing the information that is preserved in the reduced feature space. Mathematically we can express this by applying the linear transformation $y_i = B \cdot h_i$ which we already defined. The goal of all feature reduction techniques is to find the optimal B with respect to some optimization criterion. Linear Discriminant Analysis (LDA) is, according to the bibliography, one of the most reliable and effective methods.[7]

LDA seeks to reduce dimensionality while preserving as much of the class discriminatory information as possible. Because we deal with multiple classes we need to have our data labeled according to the class they belong. We associate words with class labels and assign each observed history vector to the corresponding class. LDA estimates the between and within class scatters in order to find the optimal transformation matrix.

Firstly in order to apply LDA to our history vectors, the *mean vectors and the co-variances of each class* need to be estimated.

$$\overline{m}_v = \frac{1}{N_v} \sum_{i=1}^{N_v} x_i \tag{2.3}$$

$$\Sigma_v = \frac{1}{N_v} \sum_{i=1}^{N_v} (x_i - \overline{m}_v)(x_i - \overline{m}_v)^T \tag{2.4}$$

The corresponding values for *all the classes (complete data set)* :

$$\overline{m} = \frac{1}{N}\sum_{i=1}^{N_v} x_i \tag{2.5}$$

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{m})(x_i - \overline{m})^T \tag{2.6}$$

,where N denotes the total number of training tokens and N_v stands for the number of training tokens in class v. Naturally (there are V classes), and thus:

$$\sum_{v=1}^{V} N_v = N \tag{2.7}$$

According to the above definitions the optimization criterion can be formulated as:

$$\widehat{B} = argmax_{B^L}\frac{|B_L^T\overline{\Sigma}B_L|}{|B_L^T W B_L|} \tag{2.8}$$

,where

$$W = \frac{1}{N}\sum_{v=1}^{V} N_v\Sigma_v \tag{2.9}$$

Decomposing equation 2.8 we note that the numerator represents the co-variance of the pooled training data in the transformed feature space, while the denominator represents the average co-variance within each class in the transformed feature space. Hence, the criterion really tries to maximize the "distance" between classes while minimizing the "size" of each of the classes simultaneously. This is exactly what we want to achieve because this criterion guarantees that we preserve most of the discriminant information in the transformed feature space.

In order to estimate the projection matrix B, we estimate the statistics that LDA uses to compute the scatter matrices. We need to estimate between-class scatter S_B and within-class scatter S_W. At first, we estimate two sufficient statistics for the matrices computation which are:

$$t_{1i} = \sum_{n\in i} x_n \tag{2.10}$$

$$t_{2i} = \sum_{n\in i} x_n x_n^T \tag{2.11}$$

, where i is the word/class i.e. $i = 1, \cdots, c$ and x_n is the history vector.
After estimating t_{1i} and t_{2i} for all history vectors, we compute the mean vector for each word/class.

$$m_i = \frac{1}{n_i} \sum_{n \in i} x_n = \frac{1}{n_i} t_{1i} \tag{2.12}$$

, where n_i is the amount of history vectors in each class.

Now we are finally ready to estimate the between-scatter matrix as:

$$S_B = \sum_{i=1}^{C} n_i (m_i - m)(m_i - m)^T \tag{2.13}$$

$$where, \quad m = \frac{1}{n} \sum_{i=1}^{C} n_i m_i \tag{2.14}$$

and the within-class scatter as:

$$S_i = \sum (x - m_i)(x - m_i)^T = \tag{2.15}$$

$$= \sum [xx^T - xm_i^T - m_i x^T + m_i m_i^T] = \tag{2.16}$$

$$= \sum xx^T - (\sum x) m_i^T - m_i \sum x^T + n_i m_i m_i^T = \tag{2.17}$$

$$= \sum xx^T - n_i m_i m_i^T \tag{2.18}$$

for each word/class i.

The within-class scatter for the total amount of class is given as:

$$S_W = \sum_{i=1}^{C} S_i \tag{2.19}$$

By estimating the projection matrix B we are able to project each history vector to the new-dimensional space. In our initial model history vectors are projected in the reduced feature space $y \in \mathbb{R}^L$ for $L = 50$.

2.3 Model Training

After having our training data converted to continuous space vectors it is time to choose a model training method. In this work Gaussian Mixture Models were selected. GMMs and their variations are the most widely known technology for statistical model training.

GMMs are often used in biometric systems, most notably in speaker recognition systems, due to their capability of representing a large class of sample distributions. One of the powerful attributes of the GMM is its ability to form smooth approximations to arbitrarily shaped densities. The classical uni-modal Gaussian model represents feature distributions by a position (mean vector) and a elliptic shape (covariance matrix) and a vector quantizer (VQ) or nearest neighbor model represents a distribution by a discrete set of characteristic templates. A GMM acts as a hybrid between these two models by using a discrete set of Gaussian functions, each with their own mean and covariance matrix, to allow a better modeling capability. So the GMM not only provides a smooth overall distribution t, its components also clearly detail the multi-modal nature of the density.

A GMM can also be viewed as a single-state HMM with a Gaussian mixture observation density, or an ergodic Gaussian observation HMM with fixed, equal transition probabilities. Assuming independent feature vectors, the observation density of feature vectors drawn from these hidden classes is a Gaussian mixture

In the upcoming paragraphs we discuss the structural elements of a GMM and a TGMM and we explain how we can use these structures in order to achieve our aim.

2.3.1 Multivariate Gaussian distribution & Multivariate Gaussian LM

In probability theory and statistics, the multivariate normal distribution or multivariate Gaussian distribution, is a generalization of the one-dimensional (univariate) normal distribution to higher dimensions. The univariate distribution is described by the know density function:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \tag{2.20}$$

where, $-\infty < x < \infty$

, μ is the mean of distribution and σ^2 is the variance.

One possible definition is that a random vector is said to be p-variate normally distributed if every linear combination of its p components has a univariate normal distribution. Thus the multivariate normal distribution can be formulated as:

$$f(x) = \frac{1}{\sqrt{(2\pi)^{\frac{p}{2}}\Sigma^{-1}}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \tag{2.21}$$

Within the mean vector μ there are p (independent) parameters and within the symmetric co-variance matrix Σ there are $\frac{1}{2}p(p+1)$ independent parameters. $\frac{1}{2}p(p+3)$ independent parameters in total.

However, its importance derives mainly from the Multivariate central limit theorem. The multivariate normal distribution is often used to describe, at least approximately, any set of (possibly) correlated real-valued random variables each of which clusters around a mean value.

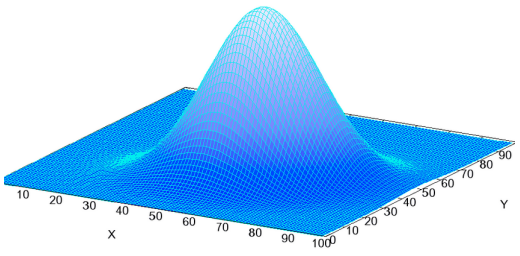

Figure 2.6: Multivariate Gaussian

The multivariate normal distribution of a k-dimensional random vector $X = [X_1, X_2, \ldots, X_k]$ can be written in the following notation:

$$X \sim \mathbb{N}(\mu, \Sigma) \tag{2.22}$$

where, $\mu = [E[X_1, X_2, \ldots, X_k]]$ and $\Sigma = [cov[X_i, X_j]]$, with $i, j \in [1, k]$

Multivariate normal distribution describes variables that tend to cluster around their mean value. Based on the Multivariate central limit theorem, any random variable can be described by the normal distribution if it has a large set of observations. That is why Gaussian distributions are often used for statistical modeling and language modeling.

In such LMs first we assume that each word is described by one distribution and we calculate each word's mean vector and co-variance matrix. With this distribution we evaluate the probability of each history given the word. With our model we want to evaluate the probability of the word given its history. Using Bays rule we have:

$$P(w|y) = \frac{P(w)p(y|w)}{p(y)} = \frac{P(w)p(y|w)}{\sum_{u=1}^{V} P(u)p(y|u)} \tag{2.23}$$

, where $P(w)$ is the unigram probability of the word.

We must consider that $P(w|y)$ must sum up to 1 for each $w \in V$.

2.3.2 Gaussian Mixture Language Model (GMLM)

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs are commonly used as a parametric model of the probability distribution of continuous measurements or features in a biometric system, such as vocal-tract related spectral features in a speaker recognition system. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm. [2]

A Gaussian mixture model is a weighted sum of M component Gaussian densities as given by the equation:

$$p(x|\lambda) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i) \tag{2.24}$$

where x is a D-dimensional continuous-valued data vector (i.e. measurement or features), w_i , $i = 1, \ldots, M$, are the mixture weights, and $g(x|\mu_i, _i)$, $i = 1, \ldots, M$, are the component Gaussian densities. Each component density is a D-variate Gaussian function of the form:

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma_i|}} \exp\left\{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right\} \tag{2.25}$$

, with mean vector μ_i and co-variance matrix Σ_i . The mixture weights satisfy the constraint that their sum is 1. The complete Gaussian mixture model is parameterized by the mean vectors, co-variance matrices and mixture weights from all component densities. These parameters are collectively represented by the notation:

$$\lambda = w_i, \mu_i, \Sigma_i \tag{2.26}$$

,with $i \in [1, M]$

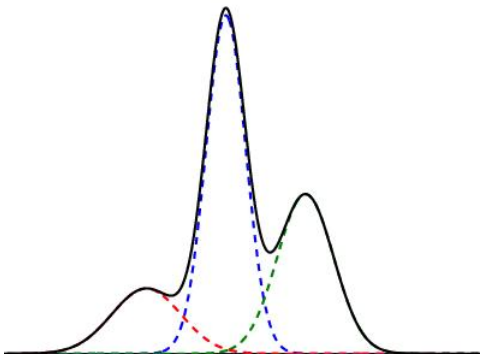

Figure 2.7: Gaussian Mixture Model

The co-variance matrices, Σ_i, can be full rank or constrained to be diagonal. Additionally, parameters can be shared, or tied, among the Gaussian components, such as having a common co-variance matrix for all components. The choice of model configuration (number of components, full or diagonal co-variance matrices, and parameter tying) is often determined by the amount of data available for estimating the GMM parameters and how the GMM is used in a language model application. It is also important to note that because the Gaussian components are acting together to model the overall feature density, full co-variance matrices are not necessary even if the features are not statistically independent. The linear combination of diagonal co-variance basis Gaussians is capable of modeling the correlations between feature vector elements. The effect of using a set of M full co-variance matrix Gaussians can be equally obtained by using a larger set of diagonal co-variance Gaussians.

Regarding our LM the model parameters for GMLM are the SVD output matrix A of the co-occurrence matrix, the LDA projection matrix B and the mixture parameters, mean vectors and covariances matrices for each word and the priors for each mixture.

2.3.3 Tied Mixture Language Model (TMLM)

After trying a variety of different experiments on simple Gaussian Mixture Models (GMMs) with various components the best results for our initial model came after training a Tied Gaussian Mixture Model (TMLM) with 64 Gaussians per GMM.[5]

Tied GMM is a specialization of GMMs that instead of having separate sets of Gaussian distributions for each word, a common set of distributions can be used for all words with different weights.

Assume that we have a set of distributions, we may refer to it as a Gaussian pool, which is common for all words. Tied Gaussian mixture model is a weighted sum of J component Gaussian densities as given by the equation,

$$P(y|w) = \sum_{k}^{K} c_{w,j} N(\mu_j, \Sigma_j) \qquad (2.27)$$

T-GMM is been used in pattern recognition and statistical modeling applications, such as acoustic modeling. Their advantage is that they use a small set of parameters for large amount of data, and, consequently, the model training is more efficient.

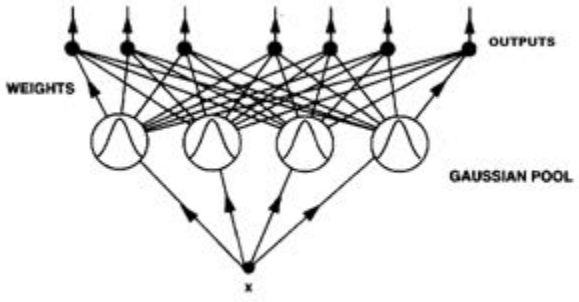

Figure 2.8: Tied GMM

Working with TMLMs helps us overcome not only N-gram problems such as generalizability and adaptability but GMLM's disadvantages as well by avoiding the big amount of parameters that a GMLM requires. TMLM provides a great deal of parameter tying across words, hence achieves robust parameter estimation. TMLM can theoretically estimate the probability of any word that has as few as two occurrences in the training data (however the probability estimation of such words is not always reliable). Moreover by tying our GMM parameters we accomplice to avoid data-over-fitting problem which rise from the small variances between some history data.

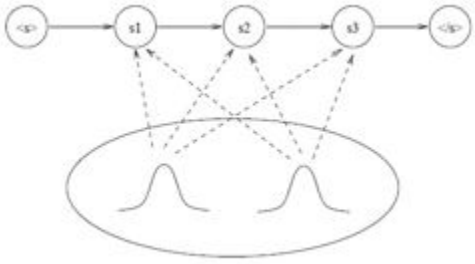

Figure 2.9: Parameter Tying

In our initial model we have tied the variance vector for each word, in order to train our model. So, all the words use the same variance vector for each distribution and different weights for each mixture.

The model parameters for TMLM are the SVD output of the co-occurrence matrix, the LDA projection matrix B and the mixture parameters, mean vectors and covariances matrices for the common set of distributions and the weights for each word.

2.4 Estimating a Model

Statistical Model Estimation deals with the estimation of parameters for a statistical model and the supply of informative model summary statistics. In our case we take as a measure the test data logarithmic probability and perplexity using the Bayes rule. Both definitions are part of Information Theory and they are the most common measures for estimating a Statistical Language Model.

Language models are used to estimate word-sequence probabilities. For a word sequence of N words, $P(W)$ has information about the sequence's probability and accuracy. Our task is using the above notions to decide on the quality of a model according to $P(W)$ of each data sequence.

Language can be thought of as an information source whose outputs are words wi belonging to the vocabulary of the language. The most common metric for evaluating a language model is the word recognition error rate, which requires the participation of a speech recognition system. Alternatively, we can measure the probability that the language model assigns to test word strings without involving speech recognition systems. This is the derivative measure of cross-entropy known as test-set perplexity.

2.4.1 Information Theory

Information theory is a branch of mathematics that overlaps into communications engineering, biology, medical science, sociology, and psychology. The theory is devoted to the discovery and exploration of mathematical laws that govern the behavior of data as it is transferred, stored, or retrieved. In this work we use information theory in order to estimate the accuracy of a Statistical Language Model.

2.4.2 Entropy

Given a language model that assigns probability $P(W)$ to a word sequence W, we can derive a compression algorithm that encodes the text W using $-\log_2 P(W)$ bits. The cross-entropy $H(W)$ of a model $P(wi|wi-n+1,\cdots,wi-1)$ on data W, with a sufficiently long word sequence, can be simply approximated as:

$$H(W) = -\frac{1}{N_W}\log_2 P(W) \tag{2.28}$$

,where N_W is the length of the text W measured in words.

2.4.3 Perplexity

The perplexity $Perplexity\{P(W)\}$ of a language model $P(W)$ is defined as the reciprocal of the (geometric) average probability assigned by the model to each word in the test set W. This is a measure, related to cross-entropy, known as test-set perplexity.

$$Perplexity\{P(W)\} = 2^{H(W)} \tag{2.29}$$

The perplexity can be roughly interpreted as the geometric mean of the branching factor of the text when presented to the language model. The perplexity defined has two key parameters: a language model and a word sequence. The test-set perplexity evaluates the generalization capability of the language model. The training-set perplexity measures how the language model fits the training data, like the likelihood. It is generally true that lower perplexity correlates with better language modeling. This is because the perplexity is essentially a statistically weighted word branching measure on the test set.

For each language model, it is possible to calculate the perplexity for test data. Perplexity minimum value is 1 that would mean that all words of a sequence have probability equal to 1. On the other hand, if a word has zero probability, then the probability of the whole sentence will be zero and perplexity will be infinite. So, we can assume that the challenge of a language model is to avoid zero probabilities. A well- built model should assign small perplexity for large test data sets. Perplexity value is a quality measure of different models for common test data.

In our occasion after the training of the models , we estimated the test data set perplexity for each language model as:

$$\log P(test\_data) = log[P(S_1)\cdot P(S_2)\cdots P(S_T)] \tag{2.30}$$

For each test data sentence, we estimate its log-probability as:

$$\log P(S_k) = \log[P(w_1| < s >< s >) \cdot P(w_2| < s > w_1) \cdots P(w_n|w_{n-2}w_{n-1})] \quad (2.31)$$

$$= \log[P(w_1| < s >< s >)] + \log[P(w_2| < s > w_1)] + \cdots + \log[P(w_n|w_{n-2}w_{n-1})] \quad (2.32)$$

Using the Bayes rule we estimate each trigram log- probability as:

$$\log[P(w_k|w_{k-2}w_{k-1})] = \log \frac{P(w_k) \cdot P(w_{k-2}w_{k-1})}{\sum_V \cdot P(u) \cdot P(w_{k-2}w_{k-1}|u)} \quad (2.33)$$

,where V is the vocabulary length and u is every possible history bigram.

Then, we estimate the test data perplexity as:

$$PPL = \exp\left\{\frac{-logprob}{T + W - OOVS}\right\} \quad (2.34)$$

, where T is the number of test data sentences, W the number of the words and $OOVS$ the number of outof–vocabulary words.

Chapter 3

Continuous Language Model Adaptation

3.1 Speech Recognition

Speech recognition is the ability of a machine or program to identify words and phrases in spoken language and convert them to a machine-readable format. Rudimentary speech recognition software has a limited vocabulary of words and phrases and may only identify these if they are spoken very clearly. More sophisticated software has the ability to accept natural speech. Speech recognition applications include call routing, speech-to-text, voice dialing and voice search.

The signal of our voice is by nature a signal in continuous space. In speech recognition in order to examine the spoken signal more easily we try to model discrete parts of the signal by cutting it into "pieces". Now each sub-part of the signal contains significantly less information than the original one and is much easier to train an efficient model for each one of these sub-parts.

After having discriminate our parts of speech the collected training-data have to be parameterized. After the parameterization is implemented we have a set of speech vectors which are now ready to use for recognizing new test-data.

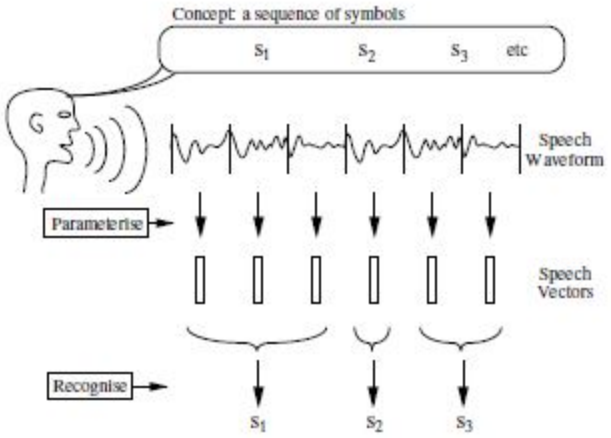

Figure 3.1: Speech recognition process [6]

However speech recognition models are not very robust. A speech recognition system will work perfectly for a single speaker but its efficiency sinks when it is tested on alternative speakers with different accents. This is exactly where the need of statistical adaptation algorithms rises. Collecting training data either in written or in spoken form, is a very uneconomic and time consuming procedure. So the best idea is to be able to modify an already trained model without the need of a huge amount of adaptation data.

Relying on this approach we assume a Statistical Language Model trained in continuous space as an alternate speech recognition model. We introduce the most known adaptation algorithms for speech recognition and we test how these algorithms work on a SLM.

3.2 Using Speech Recognition Techniques for Language Model Adaptation

Assume we are given some initial continuous density hidden Markov models (HMMs) (in our case trained in WSJ domain). We try to improve the modeling of a new domain (ATIS) by updating the HMM parameters. Statistics are gathered from the available adaptation data and used to calculate either a linear regression-based or a maximum a-posterior transformation for the mean vectors. By tying the transformations among a number of distributions, adaptation can be performed for

distributions which are not represented in the training data. It is very important to note that the arbitrary adaptation data can can be easily used and no special-enrollment sentences are needed.

Such adaptation techniques can be used in various different modes. If the true transcription of the adaptation data is known then it is termed *supervised adaptation*, whereas if the adaptation data is unlabelled then it is termed *unsupervised adaptation*. In the case where all the adaptation data is available in one block, e.g. from a speaker enrollment session, then this termed *static adaptation*. Alternatively adaptation can proceed incrementally as adaptation data becomes available, and this is termed *incremental adaptation*. In this work we tried only supervised adaptation.

As these technologies were initially tested on speech recognition systems, today there are many relevant applications which have specific tools for applying the transformations mentioned above in a trained model. We used HTK tool, which is an application developed on Cambridge University, for speech recognition and model training. As we worked with words and not with recordings we had to modify our data properly in order to make them readable from HTK. We used the HERest tool which provides us with a variant of linear and non-linear transformation adaptation techniques.

3.3 Model Adaptation using Linear Transformations

The transformation matrices are all obtained by solving a maximisation problem using the Expectation-Maximisation (EM) technique. Using EM results in the maximisation of a standard auxiliary function.[6]

The basic idea of the EM algorithm is, beginning with an initial model λ, to estimate a new model $\overline{\lambda}$, such that $p(X|\overline{\lambda}) \geq p(X|\lambda)$. The new model then becomes the initial model for the next iteration and the process is repeated until some convergence threshold is reached.

To enable robust transformations to be trained, the transform matrices are tied across a number of Gaussians. The set of Gaussians which share a transform is referred to as a regression class. For a particular transform case W_r, the M_r Gaussian components $\{m_1, m2, \ldots, m_{Mr}\}$ will be tied together, as determined by the regression class tree. The standard auxiliary function shown below is used to estimate the transforms:

$$Q(M, \widehat{M}) = -\frac{1}{2}\sum_{r=1}^{R}\sum_{m_r=1}^{M_r}\sum_{t=1}^{T} L_{m_r}(t)\left[K^{(m)} + \log(|\widehat{\Sigma}_{m_r}|) + (o(t) - \widehat{\mu}_{m_r})^T \widehat{\Sigma}_{m_r}^{-1}(o(t) - \widehat{\mu}_{m_r})\right] \tag{3.1}$$

,where

- $\boldsymbol{M}$ is the model set.
- $\widehat{\boldsymbol{M}}$ is the adapted model set.
- $\boldsymbol{T}$ is the number of observations.
- $\boldsymbol{m}$ is the number of component.
- $\boldsymbol{O}$ is a sequence of *d-dimensional* observations.
- $\boldsymbol{o(t)}$ is the observation at time t.
- $\boldsymbol{\mu_{m_r}}$ is the mean vector for the mixture component m_r.
- $\boldsymbol{\Sigma m_r}$ is the co-variance matrix for the mixture component m_r.
- $\boldsymbol{Lm_r(t)}$ is the the occupancy probability for the mixture component m_r at time t.
- $\boldsymbol{K^{(m)}}$ subsumes all constants.

and $L_{m_r}(t)$, the occupation likelihood, is defined as,

$$L_{m_r}(t) = p(q_{m_r}(t)|M, Q_T) \tag{3.2}$$

and $q_{m_r}(t)$ indicates the Gaussian component m_r at time t, and $O_T = \{o(1), \ldots, o(T)\}$ is the adaptation data. The occupation likelihood is obtained from the forward-backward process.

3.3.1 Maximum Likelihood Linear Regression

Maximum likelihood linear regression or MLLR computes a set of transformations that will reduce the mismatch between an initial model set and the adaptation data.[6]

More specifically MLLR is a model adaptation technique that estimates a set of linear transformations for the mean and variance parameters of a Gaussian mixture HMM system. The effect of these transformations is to shift the component means and alter the variances in the initial system so that each state in the HMM system is more likely to generate the adaptation data. The transformation matrix used to give a new estimate of the adapted mean is given by:

$$\widehat{\mu} = W\xi \tag{3.3}$$

where W is the nx(n+1) transformation matrix (*where n is the dimensionality of the data*) and ξ is the extended mean vector,

$$\xi = [w \quad \mu_1 \quad \mu_2 \quad \dots \quad \mu_n]^T \tag{3.4}$$

where w represents a bias offset whose value is fixed at 1.
Hence W can be decomposed into:

$$W = [bA] \tag{3.5}$$

,where A represents an nxn transformation matrix and b represents a bias vector. This form of transform is referred to in the code as MLLRMEAN

3.3.2 Mean Transformation Matrix (MLLRMEAN)

Substituting the for expressions for MLLR mean adaptation

$$\widehat{\mu}_{m_r} = W_r \xi_{m_r} \tag{3.6}$$
$$\widehat{\Sigma}_{m_r} = \Sigma_{m_r} \tag{3.7}$$

into the auxiliary function, and using the fact that the covariance matrices are diagonal, yields

$$Q(M, \widehat{M}) = K - \frac{1}{2} \sum_{r=1}^{R} \sum_{j=1}^{d} \left(w_{rj} G_r^{(j)} w_{rj}^T - 2 w_{rj} k_r^{(j)T} \right) \tag{3.8}$$

,where W_{rj} is the j^th row of W_r and

$$G_r^{(i)} = \sum_{m_r=1}^{M_r} \frac{1}{\sigma_{m_r i}^2} \xi_{m_r} \xi_{m_r}^T \sum_{t=1}^{T} L_{m_r}(t) \tag{3.9}$$

$$k_r^{(i)} = \sum_{m_r=1}^{M_r} \sum_{t=1}^{T} L_{m_r}(t) \frac{1}{\sigma_{m_r i}^2} o_i(t) \xi_{m_r}^T \tag{3.10}$$

Differentiating the auxiliary function with respect to the transform Wr , and then maximising it with respect to the transformed mean yields the following update:

$$w_{ri} = k_r^{(i)} G_r^{(i)-1} \tag{3.11}$$

The above expressions assume that each base regression class r has a separate transform. [6]

3.3.3 Co-variance Transformation Matrix (MLLRCOV)

There are two standard forms of linear adaptation of the variances. The first is of the form:

$$\widehat{\Sigma} = B^T H B \tag{3.12}$$

,where H is the linear transformation to be estimated and B denotes the inverse of the Choleski factor of Σ^{-1}. Thus :

$$\Sigma^{-1} = CC^T \tag{3.13}$$
$$B = C^{-1} \tag{3.14}$$

This form of transform results in an effective full co-variance matrix if the transform matrix H is full. This makes likelihood calculations highly inefficient. This form of transform is only available with a diagonal transform and in conjunction with estimating an MLLR transform. The MLLR transform is used as a parent transform for estimating H.

An alternative more efficient form of variance transformation is also available. Here, the transformation of the covariance matrix is of the form:

$$\widehat{\Sigma} = H \Sigma H^T \tag{3.15}$$

,where H is the nxn co-variance transformation matrix. This form of transformation can be efficiently implemented as a transformation of the means and the features.

$$N(o; \mu, H\Sigma H) = \frac{1}{|H|} N(H^{-1}o; H^{-1}\mu, \Sigma) = |A| N(Ao; A\mu, \Sigma) \tag{3.16}$$

,where $A = H^{-1}$. Using this form it is possible to estimate and efficiently apply full transformations. [6]

3.3.4 Constrained MLLR Transformation Matrix (CMLLR)

Constrained MLLR forces the transform to be the same for both mean and variance parameters. If we assume the for expressions for CMLLR adaptation

$$\widehat{\mu}_{m_r} = \mu_m r \tag{3.17}$$
$$\widehat{\Sigma}_{m_r} = H_r \mu_m r H_r^T \tag{3.18}$$

and we substitute them into the auxiliary function, with respect to the fact that the co-variance matrices are diagonal yields:

$$Q(M, \widehat{M}) = K + \sum_{r=1}^{R} \left[\beta \log(p_{ri} w_{ri}^T) - \frac{1}{2} \sum_{j=1}^{d} (w_{rj} G_r^{(j)} w_{rj}^T - 2 w_{rj} k_r^{(j)}) \right] \tag{3.19}$$

,where $W_r = [-A_r \widehat{b}_r H_r^{-1}] = [bA]$

w_r^i is i^{th} row of W_r, the 1xn row vector p_{ri} is the zero extended vector of co-factors of A_r and $G_r^{(i)}$ and $k_r^{(i)}$ are defined as:

$$G_r^{(i)} = \sum_{m_r=1}^{M_r} \frac{1}{\sigma_{m_{ri}}^2} \sum_{t=1}^{T} L_{m_r}(t) \zeta(t) \zeta(t)^T \tag{3.20}$$

$$k_r^{(i)} = \sum_{m_r=1}^{M_r} \frac{\mu_{m_{ri}}}{\sigma_{m_{ri}}^2} \sum_{t=1}^{T} L_{m_r}(t) \zeta(t)^T \tag{3.21}$$

Differentiating the auxiliary function with respect to the transform W_r , and then maximising it with respect to the transformed mean yields the following update:

$$w_{ri} = (a p_{ri} + k_r^{(i)}) G_r^{(i)-1} \tag{3.22}$$

,where a satisfies the following quadratic equation:

$$a^2 p_{ri} G_r^{(i)-1} p_{ri}^T + a p_{ri} G_r^{(i)-1} k_r^{(i)T} - \beta = 0 \tag{3.23}$$

There are thus two possible solutions for a. The solutions that yields the maximum increase in the auxiliary function (obtained by simply substituting in the two options) is used. This is an iterative optimization scheme as the co-factors mean the estimate of row i is dependent on all the other rows.[6][1]

3.3.5 Adaptation Procedure for MLLR adaptation using Baum-Welch Algorithm

1. Initialization of $A_o(g) = I$, $b_o(g) = 0$, $g = 1, \cdots, N_g$ for all transformation

2. **E-step**: Perform one iteration of the forward-backward algorithm using Gaussians transformed with the current value of the transformations $W_k(g) = [A_k(g), b_k(g)]$. For all components and all GMMs collect the sufficient statistics:

(a) $\overline{\mu}_i(g)$
(b) $\overline{\Sigma}_i(g)$
(c) $n_i(g) = \sum_x p(\varpi_i|A_o, b_o, x)$

The re-estimation formulae and the sufficient statistics which are calculated difference according to the type of MLLR adaptation.

3. **M-step**: Compute the new transformation parameters $[A_{k+1}(g), bk + 1(g)]$
4. If another iteration, goto **(2)**

[1]

3.4 Model Adaptation using Maximum A-Posteriori Probability (MAP)

Model adaptation can also be accomplished using a maximum a-posteriori (MAP) approach. This adaptation process is sometimes referred to as Bayesian adaptation. MAP adaptation involves the use of prior knowledge about the model parameter distribution. Hence, if we know what the parameters of the model are likely to be (before observing any adaptation data) using the prior knowledge, we might well be able to make good use of the limited adaptation data, to obtain a decent MAP estimate. This type of prior is often termed an informative prior. Note that if the prior distribution indicates no preference as to what the model parameters are likely to be (a non-informative prior), then the MAP estimate obtained will be identical to that obtained using a maximum likelihood approach.

For MAP adaptation purposes, the informative priors that are generally used are the speaker independent model parameters. For mathematical tractability conjugate priors are used, which results in a simple adaptation formula.

The update formula for a single stream system for state j and mixture component m is:

$$\widehat{\mu_{jm}} = \frac{N_{jm}}{N_{jm} + \tau} \cdot \overline{\mu_{jm}} + \frac{\tau}{N_{jm} + \tau} \mu_{jm} \tag{3.24}$$

where τ is a weighting of the a priori knowledge to the adaptation speech data and N is the occupation likelihood of the adaptation data, defined as,

$$N_{jm} = \sum_{r=1}^{R} \sum_{t=1}^{T_r} L_{jm}^r(t) \tag{3.25}$$

,where μ_{jm} is the speaker independent mean and $\overline{\mu_{jm}}$ is the mean of the observed adaptation data and is defined as,

$$\overline{\mu_{jm}} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T_r} L_{jm}^{r}(t) o_t^T}{\sum_{r=1}^{R} \sum_{t=1}^{T_r} L_{jm}^{r}(t)} \tag{3.26}$$

As can be seen, if the occupation likelihood of a Gaussian component (N_{jm}) is small, then the mean MAP estimate will remain close to the speaker independent component mean. With MAP adaptation, every single mean component in the system is updated with a MAP estimate, based on the prior mean, the weighting and the adaptation data. Hence, MAP adaptation requires a new speaker-dependent model set to be saved.

One obvious drawback to MAP adaptation is that it requires more adaptation data to be effective when compared to MLLR, because MAP adaptation is specifically defined at the component level.

When larger amounts of adaptation training data become available, MAP begins to perform better than MLLR, due to this detailed update of each component (rather than the pooled Gaussian transformation approach of MLLR).

In fact the two adaptation processes can be combined to improve performance still further, by using the MLLR transformed means as the priors for MAP adaptation (by replacing μ_{jm} in equation 3.6 with the transformed mean of equation 3.3). In this case components that have a low occupation likelihood in the adaptation data, (and hence would not change much using MAP alone) have been adapted using a regression class transform in MLLR.

Chapter 4

Experiments & Model Evaluation Results

4.1 Introduction

During this work we tried a big variation of different adaptation approaches. We performed six baseline experiments in total. Half of them were made with a discrete LM, built by SRILM toolkit [4] and the rest with a continuous LM. As Continuous LM adaptation has never been tried before and thus we had no other similar mechanism to compare our work, our baseline experiments were focused on training a LM with the same data we used for adaptation.

Regarding our adaptation experiments we tried both MLLR and MAP adaptation techniques. Initially we experimented on global/domain HMM transformations and afterwards we tried to adapt our models by organizing the adaptation data into clusters and applying a different transformation to each background model. At last we combined linearly our best experimental tries and we examined which combination improved our results most.

All the experiments were made with various amounts of adaptation data and with a constant vocabulary of 782 words. As covered before, these words are the common words from the WSJ initial model's vocabulary and the ATIS domain. In this section we will refer to it as *common vocabulary*.

All the experiments were performed on GMMs with tied variances as, as proved experimentally, they are capable of modeling language much better than the common GMMs.

4.2 Baseline Experiments

We examined three different baseline experiments both in discrete and continuous LM. We trained each LM by combining data both from WSJ and ATIS domain with respect to the adaptation vocabulary.

The training and test data information of the two domains is presented in the following two tables:

Table 4.1: Training Data Statistics from WSJ and ATIS

	Number of Sentences	Number of Words	Number Different Words
ATIS	18688	224566	782
WSJ	150000	4447740	2700

Table 4.2: Test Data Statistics from ATIS

	Number of Sentences	Number of Words	Number Different Words
ATIS	4000	50424	492

4.2.1 Discrete Language Model

To create the discrete Lm we used,as we said before the SRILM toolkit. SRILM is a collection of C++ libraries, executable programs, and helper scripts designed to allow both production of and experimentation with statistical language models for speech recognition and other applications. The toolkit supports creation and evaluation of a variety of language model types based on N-gram statistics, as well as several related tasks, such as smoothing and class-based models.

At first it generates the n-gram count file from the corpus, then it trains the language model from the n-gram count file and it calculates the test data perplexity using the trained language model. Also it can perform word clustering.

In particular we give an example of how SRILM works:

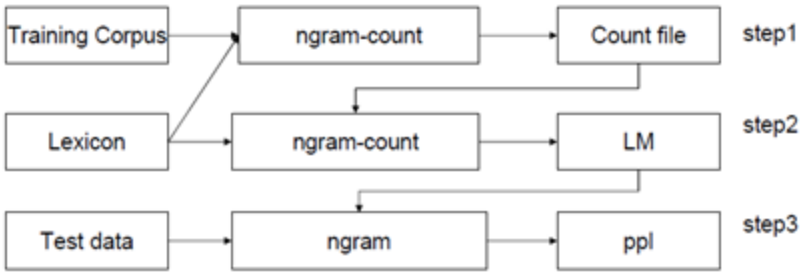

Figure 4.1: SRILM toolkit

This command generates and manipulates N-gram counts, and estimates N-gram language models from them.

- **ngram -count**
  -vocab Lexicon.file -text train.txt -order 3 -write *train_3gram* -unk

  ,where:

- -vocab file: reads a vocabulary from file.
- -text filename: train data set
- -order: sets the maximal length of N-grams
- -write filename: output file that contains N-gram counts
- -unk: sets any unknown words with oov (out-of-vocabulary)

The output file has the following format and it presents the N-gram counts:

vestavia 1
vestavia suburb 1
vestavia suburb comma 1
⋮
fortys line 1
fortys line twenty 1

The next step is to create the language model file. To do so we have to execute the bellow command:

- **ngram -count**
  -vocab Lexicon.file -read $train\_3gram$ -order 3 -lm langmod.lm

  ,where:

- -vocab file: vocabulary file.
- -read countfile: N-gram counts file
- -order: N-gram length
- -lm langmod.lm: language model file

The output file has the following format:

\data
ngram 1=57789
ngram 2=849565
ngram 3=375899
\1-grams:
-6.604123 1 -0.1859108
-6.604123 2 -0.1859108
-1.428032 < \s >
⋮
\2-grams:
-0.4572527 1 rightparen
-0.4572527 2 rightparen
⋮
\3-grams:
⋮
-0.893048 mr zyman said -0.893048 mr zyman says
\end

At last we have to evaluate our model. To do so we execute the the command:

- **ngram -ppl test.txt**
  -order 3 -lm langmod.lm

  ,where:

- -ppl file: test data perplexity file.

- -order: N-gram length
- -lm langmod.lm: language model file

The output file contains the logarithmic probability of each sentence and its perplexity. It also evaluates the perplexity of the whole test data, the number of words and the out-of-vocabulary words.

p(banks — < unk > ...) = [1gram] 0.000475788 [-3.32259]
p(need — banks ...) = [1gram] 7.3968e-05 [-4.13096]
p(more — need ...) = [2gram] 0.0191286 [-1.71832]
p(period — < unk > ...) = [1gram] 0.0436764 [-1.35975]
p(< /s > — period ...) = [2gram] 0.925196 [-0.0337663]
1 sentences, 6 words, 2 OOVs
0 zeroprobs, logprob= -10.5654 ppl= 129.741
ppl1= 437.869
file test.txt: 4000 sentences, 50424 words, 350 OOVs
0 zeroprobs, logprob= -127767 ppl= 589.524 ppl1= 1080.5

The final SRILM results for all the experiments are presented in the table bellow:

Table 4.3: SRILM baseline experiments

TRAIN DATA	TEST DATA	NUM OF TRAINING DATA	VOCABULARY FROM	PERPLEXITY
ATIS	ATIS	18688	common vocabulary	15
WSJ	ATIS	150000	common vocabulary	589
WSJ+ATIS	ATIS	168688	common vocabulary	29

4.2.2 Continuous Language Model

We tried the same baseline experiments with the continuous LM. To do so we had to modify the code of Mr.Tsiakas. The main problem was that the only option was to extract the vocabulary from the given training corpus. We wrote a Perl script which gives us the option to train a Language Model with a given vocabulary and

modifies the training corpus according to it.

Moreover we checked all the given programming code parts in order to make it more parametric. Our target was to make it work with every possible amount of training data and vocabulary.

Finally we modified the model estimation code which was written in Matlab aiming to make it faster. The initial code needed almost 9 days in order to estimate the perplexity for 4000 sentences of test data. After our editing we managed to take results for each experiment after 8 hours.

As covered before, on model training chapter, we edit our training sentences by transforming them into indecies. We split them into train and test sets. Then, we construct the co-occurrence matrix. We perform Singular Value Decomposition for dimensionality reduction. Using this word mapping, we collect our history vectors for each word and map them to a lower dimension using Linear Discriminant Analysis. The projection matrix B, is used for the projection of each history matrix to our new continuous space, with lower dimension.

The EM algorithm is used to train mixture parameters, such as mean, variance vectors and mixture weights. Language model parameters are estimated and we can use the test data to evaluate the test set perplexity.

The perplexity estimation for each test sentence was computed as mentioned in paragraph 2.4.3

The final Continuous LM results for all the experiments are presented in the table bellow:

Table 4.4: Continuous LM baseline experiments

TRAIN DATA	TEST DATA	NUM OF TRAINING DATA	VOCABULARY FROM	PERPLEXITY
ATIS	ATIS	18688	common vocabulary	750
WSJ	ATIS	150000	common vocabulary	1543
WSJ+ATIS	ATIS	168688	common vocabulary	825

4.3 Adaptation Experiments

As already mentioned we performed various adaptation techniques. We applied MLLR-MEAN, Constrained MLLR and MAP adaptation to all models and we experimented with various collections of history vectors for adaptation.

Before applying the adaptation techniques, we had to carry our adaptation data exactly in the same form as the training data were before the training of the initial model.

This is necessary for two main reasons. The first one is that we deal with NLP problem in continuous space. Hence the initial word and history vectors that Singular Value Decomposition confers us (1x100) suffer from the problem of dimensionality. Practically that means that the memory and computational cost is very high to achieve the desired results. The second reason is that our results have to be completely comparable with our baseline experiments which were made with the initial continuous LM.

4.3.1 Preparation of the Adaptation Data

Until now we have explained how these adaptation methods are being applied to our initial models. However in order to adapt our models the adaptation data need a curious pre-processing.

Firstly we have to clean our adaptation data from all punctuation. Thus we replaced every punctuation with its periphrastic description. For example "**.**" became "**period**", "**,**" became "**comma**" , ""**"** became "**quote**" etc.

Secondly we had to clean the data from empty lines , trash undefined/incorrect words and special tags (ex. @fragment-reject@ or @noise-reject@). Moreover we replaced every abbreviation with the real word sequence (ex. "what's" to "what is")

After that we had to turn all data into lowercase and replace every digit or number with its alphabetic synonym (ex. "1" to "one").

In the next step we inserted $< s >$ and $< /s >$ symbols at the beginning and the end of each sentence, consequently.

At last we had to define our vocabulary. The background model was trained with a vocabulary of 2700 most frequent words. In our approach we decided to

define our vocabulary as the collection of the common words from the two domains. According to this assumption we concluded to a vocabulary consisting 782 words. Every word from the adaptation data which was not in the vocabulary was replaced by the tag $< unk >$. This tag was considered as one (and the most frequent) of the vocabulary words.

After the pre-editing mentioned above a training sentence would be transformed as shown below:

Initial Sentence

What's the seating capacity for US air flight 975

Final Sentence

$< s >$ what is the $< unk >$ capacity for us air flight nine seven five $< /s >$

The next step was to turn our data into index sequences. Each index shows the position of each word in the vocabulary. The vocabulary words are listed in descending order according to their occurrences. For example the tag $< unk >$ which is the most frequent word in our adaptation data is represented by the number *1*.

So the above sentence was transformed to a sequence of numbers separated with a space char.

Indexed Sentence

3 8 12 6 1 247 27 46 50 88 7 61 190 34 2

After pre-editing was completed we had to separate a sub-part of the adaptation data in order to test our re-estimated models. We had in total 22.800 adaptation sentences. We kept 4000 sentences for testing and the other 18800 (or sub-parts of them) were used as training data.

The complete information about adaptation data is summarized in the table bellow:

ATIS	Training Data	Test data
Num of Sentences	18688	4000
Num of Words	224566	50424
Num of different words	782	492

Table 4.5: Adaptation Data Information

4.3.2 Before Adapting the Background HMMs

To apply the adaptation algorithms we used the HTK speech recognition toolkit. However HTK is by default a tool for speech modeling. In order to work with written entities an extra modification of our data was needed. Fortunately HTK provides us an option to write our data in an HTK readable format.

Another problem we had to solve was to match the words from the adaptation vocabulary with the initially trained models. For each word in adaptation data we had to find its corresponding background model in order to perform adaptation. Thus we wrote a Perl script which made the word matchmaking and saved the results in a file. This file informs us that the i_{th} word from ATIS vocabulary is related with the j_{th} word/HMM from the initial WSJ model.

In all experiments we re-estimated only the mean vector of each GMM's component. All the variances where tied between the Gaussians and we used only diagonal variance matrices.

4.3.3 Initial Adaptation Approaches

As covered before the adaptation data had to be transformed in the same way as the training data before re-estimating the background HMMs.

Therefore, in our first tries, targeting to make adaptation data to fit the background model's training data we found the co-occurrences between words in our corpus. Hence we created an (782x782) co-occurrence matrix which comprises the above information as explained in detail in paragraph 2.2.2 .

After the co-occurrence vectors had been collected we applied on them the SVD algorithm (2.2.3) and create an 1x100 vector for each word. As already mentioned with SVD we manage simultaneously to reduce the vectors dimensions and to expose better the various relationships among the adaptation data.

Moving to the next step we collected for each word the previous two history vectors and we concatenated them. Hence we had an 1x200 vector for each word's history bigram (2.2.4).

The last step before model adaptation was to reduce even more the vectors dimensions. Thus we performed the LDA algorithm 2.2.5. LDA algorithm manages to reduce vectors dimensionality (*from 200 to 50*) while preserving as much of the class discriminatory information as possible. We estimate a projection matrix B (200x50) which gives us the ability to project each history vector (equation 2.2) to the new-dimensional space.

Unfortunately, because of lack of adaptation data for many in-vocabulary words the final results were quite disappointing . The main problem was that the majority of the extracted SVD vectors were quite small and hence unable to represent an efficient and reliable model. The training samples were influencing our models efficiency and we were observing huge differences between the perplexity according to the collection of the adaptation data.

In the following charts we illustrate some characteristic results of this approach.

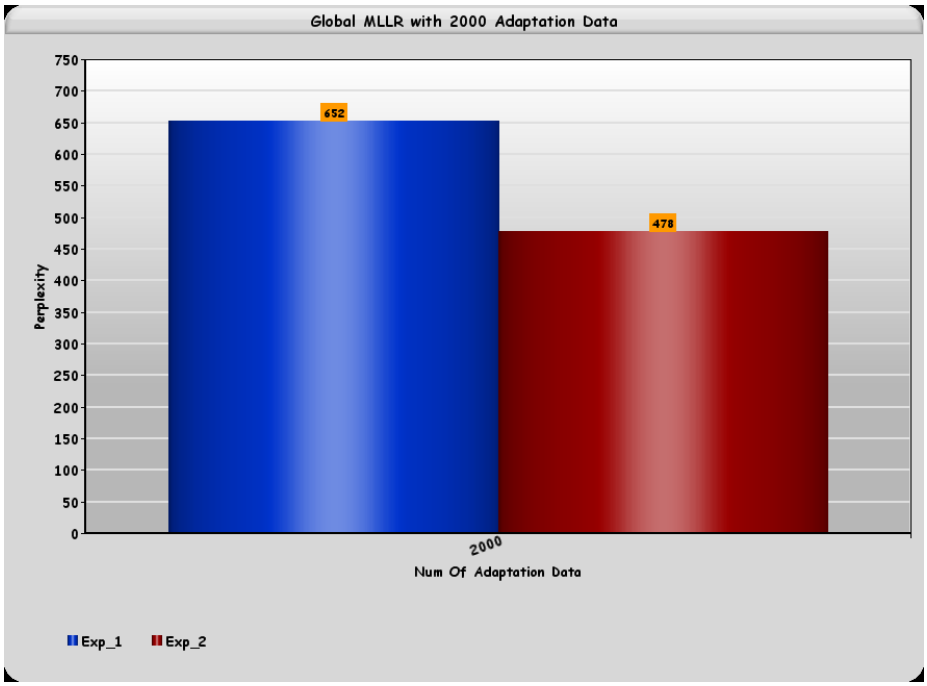

Figure 4.2: Initial experiments-Global MLLR with 2000 Adaptation data

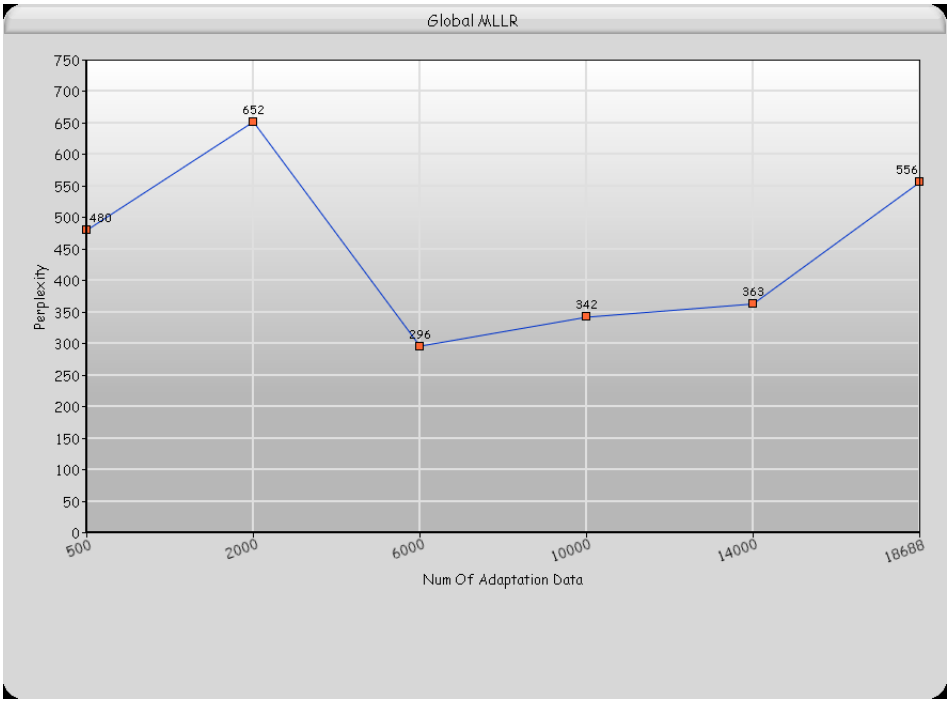

Figure 4.3: Initial experiments-Global MLLR

As it can be easily observed not only our model is fully depended from the adaptation data but despite the theoretical definitions the perplexity varies while increasing the adaptation sentences.

Non-square co-occurrence matrix

In a try to overtake the above problem we tried to work with non-square co-occurrence matrices.

More specifically we handled only the words which had more than N occurrences in our adaptation data and we extracted the SVD vectors only for these words. For the rest in-vocabulary words we used the SVD vector of the $< unk >$ tag when needed.

We examined the same experiments again however with no significant improvement. Conversely the dependence between the collection of adaptation data and the models efficiency was strengthen.

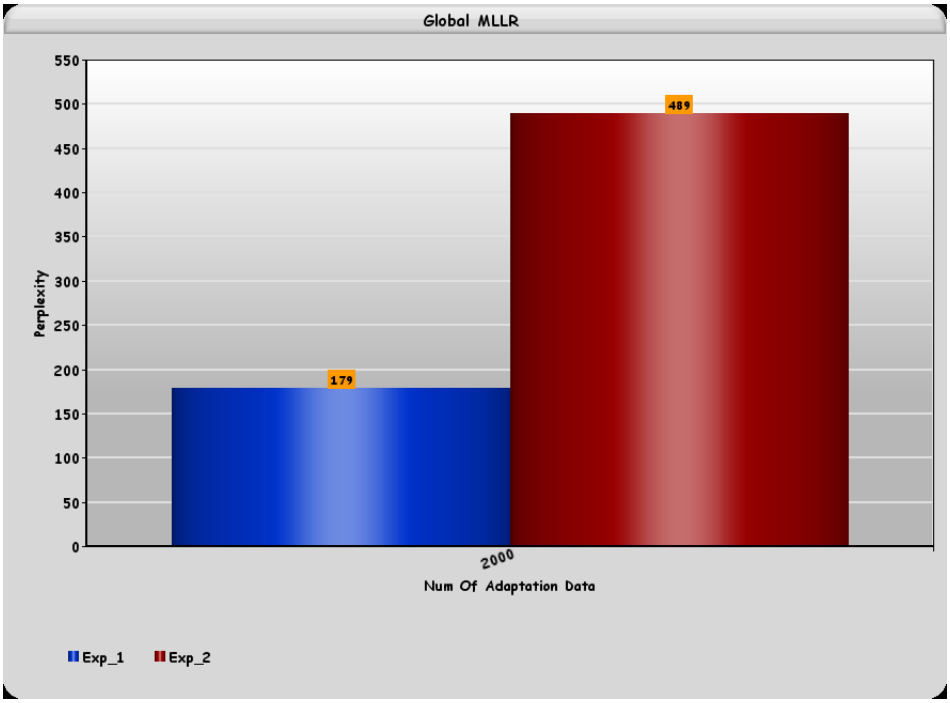

Figure 4.4: Non-Square co-occurrence matrix-Global MLLR

SVD vectors from initial model

Finally we managed to secure our model's stability by taking advantage of the SVD vectors of the initial WSJ LMs. As covered before our initial LM is consisted of 2700 words. None of these words had less than 10 occurrences in the WSJ training corpus and thus the SVD vectors could represent efficiently the required information.

In contrast to the previous tries we started our adaptation procedure by collecting the history bi-grams for each word. In spite of extracting the SVD vectors from the adaptation data we made the word-mapping by assigning each word to its related SVD vector from the initial model. As covered our vocabulary is practically a sub-part of the initial vocabulary as we worked with the common words of the two domains.

It is very important to mention that the prior probabilities for each GMM initially were extracted by modifying the WSJ training corpus with respect to the adaptation vocabulary. Eventually the prior probabilities were not accurate to represent efficiently the frequency of the in-vocabulary words from ATIS domain. However we had manage to build a stable model regarding the collection of adaptation sentences.

The chart bellow illustrates the differentiation of the perplexity as the number of adaptation data rises.

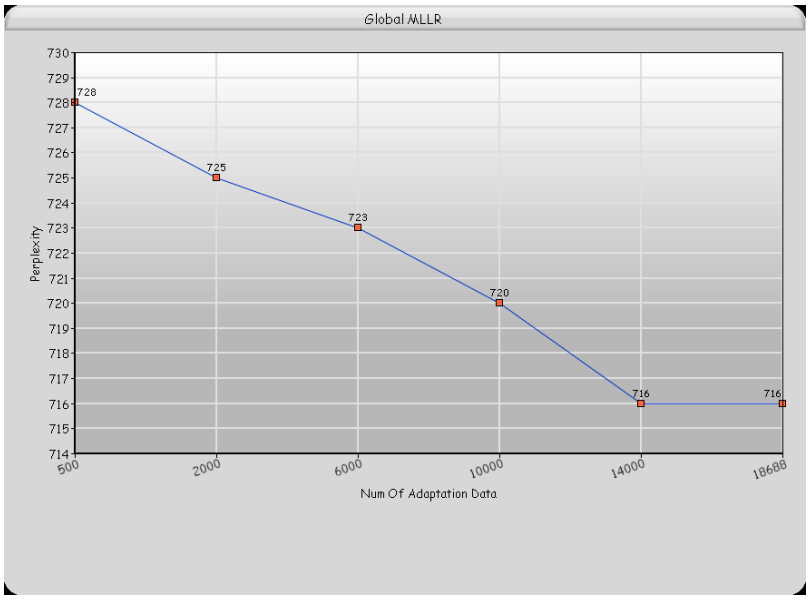

Figure 4.5: Adaptation with SVD vectors and prior probabilities from WSJ initial model

Modifying GMMs prior probabilities

It is obvious however that the perplexities in the previous chart are extremely high. In comparison to our baseline experiments with the Continuous LM we have achieved a small improvement unable though to make our system usable.

The solution came after scaling the words prior probabilities according to the adaptation data.

More specifically, we extracted the prior word probabilities from the adaptation data. For each adaptation data-set we counted the number of occurrences for each word and we divided this amount with the total number of words. When we used a subset of the total adaptation data (500,1000 adaptation sentences etc), where not all the in-vocabulary words occurred, we assumed that the missing words had at least one occurrence.

Through this assumption we managed to weight our models significance with respect to the ATIS vocabulary. As it is shown in the following sections the results were impressively improved.

4.3.4 Adaptation using MLLR Decision Rule

To apply MLLR adaptation with HTK we used the **HERest tool** with the *option -u set to a*. By typing: *HADAPT:TRANSKIND = MLLRMEAN* or *HADAPT:TRANSKIND = CMLLR* to the command's configuration file we let the tool know which type of linear adaptation we want.

The HTK command HERest which was used is the following:

```
%_____________________GMM GLOBAL ADAPTATION_HTK_____________________________________
retval= system([get_htk_path 'HERest '  HTK_OPTIONS ' -C ' configall ' -C ' config_file ...
' -S ' scpfilename ...
' -I ' mlf_file ...
' -H ' hmmacros ...
' -H ' hmmdefs ...
' -H ' global_file ...
' -K ' adapt_models ...
' -u a '...
'-h ' mask ' ' hmm_list  ]);
```

,where *hmmdefs* is the path to background models, *scpfilename* includes the path to the appropriate adaptation data, *configall* is the commands configuration

file and *adapt_models* is the path where the adapted models are saved. All the other input files are there to secure the correct choice of the adaptation data. They are mandatory by HTK but they include no useful information to be reported.

Global Adaptation Experiments

In global adaptation experiments we collect all the history vectors ,from all the vocabulary's words, and we adapt each initial model with the same adaptation data. We perform in total 782 transformations ,one for each word/model.

We made 6 experiments in total. The difference between the experiments is that each time we work with a different number of adaptation data. For each experiment the adaptation data were chosen randomly from the total amount of adaptation sentences.

The Global MLLR adaptation results are presented in the following matrices

Table 4.6: MLLRMEAN Global Adaptation

NUM OF ATIS ADAPTATION SENTENCES	PERPLEXITY
500	144
2000	135
6000	135
10000	134
14000	133
18688	132

We examined the same experiments again but this time instead of using the LDA projection matrix from the adaptation data we used the LDA matrix from the initial LM.

Table 4.7: MLLRMEAN Global Adaptation - LDA matrix from initial model

NUM OF ATIS ADAPTATION SENTENCES	PERPLEXITY
500	144
2000	136
6000	134
10000	134
14000	134
18688	134

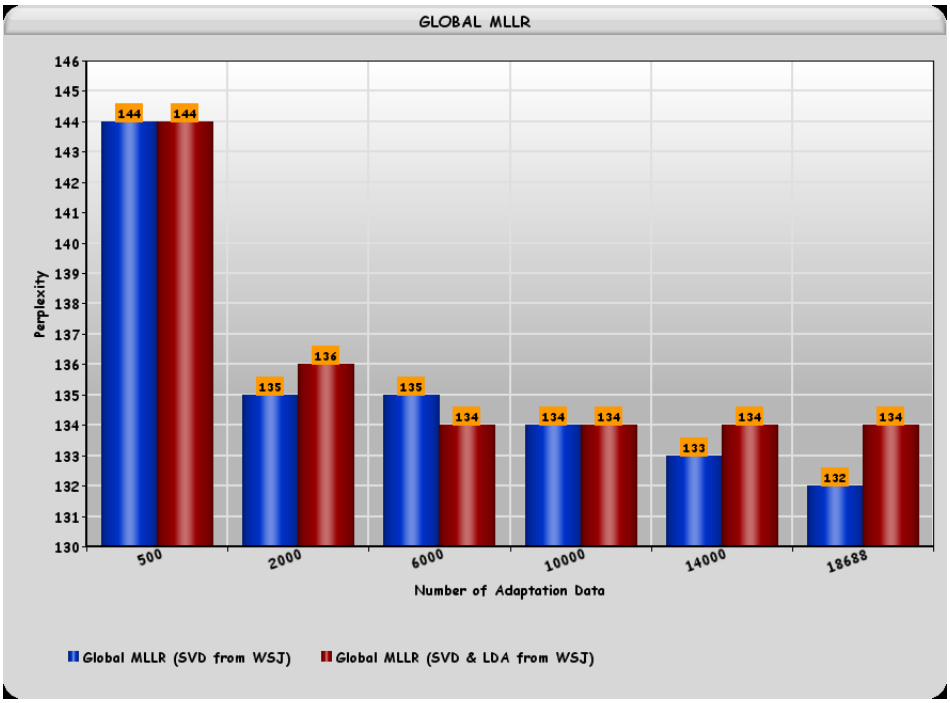

Figure 4.6: (1)Global MLLR

As we can see there are no significant differences between the two approaches. For that reason we continued experimenting by extracting the LDA matrix from the adaptation data.

Table 4.8: CMLLR Global Adaptation

NUM OF ATIS ADAPTATION SENTENCES	PERPLEXITY
500	204
2000	174
6000	171
10000	165
14000	158
18688	153

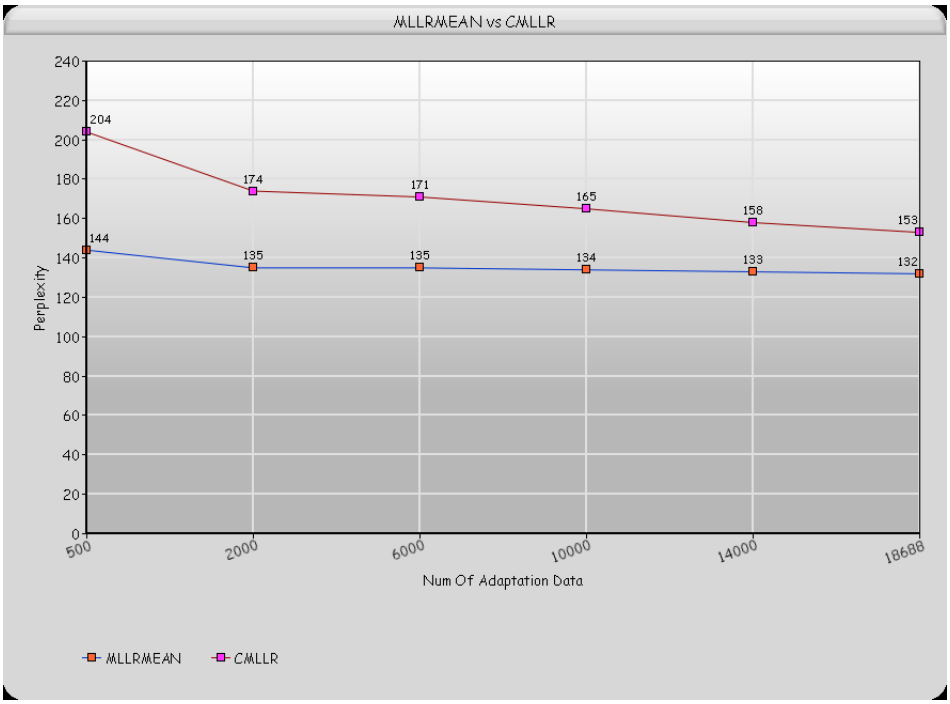

Figure 4.7: MLLRMEAN vs CMLLR

4.3.5 Adaptation using MAP Decision Rule

In MAP adaptation we tried two different types of experiments. The first one is known as domain MAP adaptation and the second one as word-class MAP adaptation. The difference is again on the set of history vectors that we use to adapt each word/model.

To apply MAP adaptation with HTK we used again the **HERest tool** but now with the option *-u set to p*. By setting *-u* to *mp* we ask HTK to update only the mean vectors.

The HTK command HERest which was used is the following:

```
%_____________________GMM GLOBAL ADAPTATION_HTK_____________________________________________
retval= system([get_htk_path 'HERest '  HTK_OPTIONS ' -C ' configall ' -C ' config_file ...
' -S ' scpfilename ...
' -I ' mlf_file ...
' -H ' hmmacros ...
' -H ' hmmdefs ...
' -H ' global_file ...
' -M ' adapt_models ...
' -u mp '...
'-h ' mask ' ' hmm_list  ]);
```

,regarding the input files applies the same as in MLLR adaptation.

Domain MAP

In Domain Map adaptation we approach the problem in the same way as we did with Global MLLR. We collect all the history vectors from all the words and we practice the same adaptation to all initial models.

Again we practiced our experiments two times. In the first try we worked with the LDA projection matrix which was extracted from the adaptation data and in the second one we use the LDA matrix which was extracted from the history vectors of the initial model.

The experimental results are shown in the following two tables:

Table 4.9: Domain MAP Adaptation

NUM OF ATIS ADAPTATION SENTENCES	PERPLEXITY
500	154
2000	139
6000	136
10000	135
14000	134
18688	133

Table 4.10: Domain MAP Adaptation - LDA matrix from initial model

NUM OF ATIS ADAPTATION SENTENCES	PERPLEXITY
500	145
2000	136
6000	134
10000	134
14000	134
18688	134

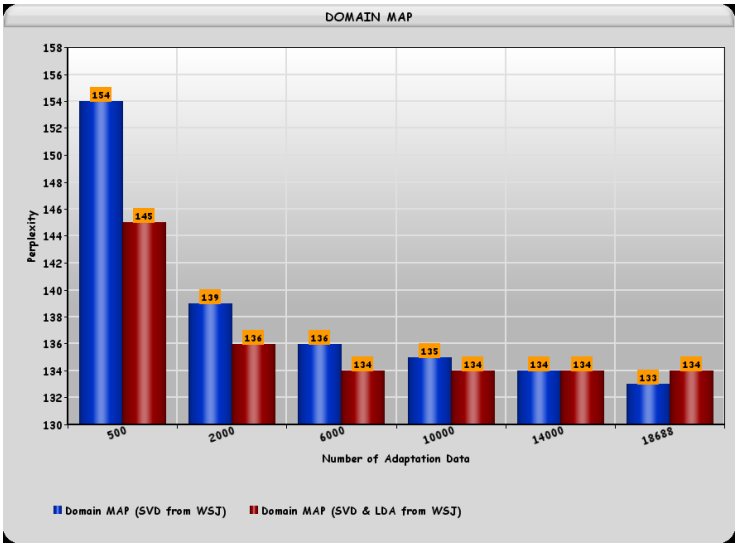

Figure 4.8: Domain MAP

4.3.6 Clustering Adaptation Experiments

To examine a different approach, we tried to adapt each initial LM with a different subset of history vectors. To do so we organized our words in classes. When we wanted to adapt a word/model we only considered the histories of words belonging to the same class with the word for adaptation.

The class-creation was made by the SRILM toolkit. SRILM accepts as input a data-set and a vocabulary and creates as many classes as asked. The SRILM places the words in clusters according to their co-occurrences.

Each word belonged strictly only in one class. We tried in total 11 different clustering experiments by alternating the number of clusters at each one. We performed the same experiments both on WSJ background models and on already, globally adapted models.

As before, for each experiment, we computed 782 transformations in total, one for each word. However each word was adapted with a different collection of history vectors.

MLLR Clustering on WSJ initial models

Table 4.11: (1)MLLR Clustering-500 adaptation sentences

NUM OF WORD-CLASSES	PERPLEXITY
10	175
20	162
30	162
40	162
50	162
60	162
70	162
80	162
90	162
100	162
782	168

Table 4.12: (1)MLLR Clustering-1000 adaptation sentences

NUM OF WORD-CLASSES	PERPLEXITY
10	156
20	168
30	169
40	169
50	169
60	169
70	169
80	169
90	169
100	169
782	172

Table 4.13: (1)MLLR Clustering-18688 adaptation sentences

NUM OF WORD-CLASSES	PERPLEXITY
10	114
20	104
30	99
40	95
50	93
60	93
70	96
80	99
90	100
100	105
782	117

It is obvious that as the adaptation data increase our clustering approach delivers better results. When using all the adaptation data we are able to create up to 60 classes and the results are significantly better than applying a global adaptation. However when less data used clustering does not affect our model the same way. No more than 20 classes can be made and the perplexity is higher than applying global adaptation with the same adaptation data.

MLLR Clustering on globally adapted models

Table 4.14: (2)MLLR Clustering-500 adaptation sentences

NUM OF WORD-CLASSES	PERPLEXITY
10	144
20	144
30	144
40	144
50	144
60	144
70	144
80	144
90	144
100	144
782	144

Table 4.15: (2)MLLR Clustering-1000 adaptation sentences

NUM OF WORD-CLASSES	PERPLEXITY
10	135
20	138
30	138
40	138
50	138
60	138
70	138
80	138
90	138
100	138
782	138

Table 4.16: (2)MLLR Clustering-18688 adaptation sentences

NUM OF WORD-CLASSES	PERPLEXITY
10	114
20	105
30	99
40	95
50	93
60	92
70	95
80	95
90	96
100	98
782	103

These results come with no surprise. As our models are already close enough to the ATIS domain when we apply this clustering approach no great changes are observed. When dealing with a small amount of data we see that the affect of the transform is almost unobserved. However when we use the total amount of data the perplexity falls significantly in contrast to its initial model.

MAP Clustering on WSJ initial models

Table 4.17: (1)MAP Clustering-18688 adaptation sentences

NUM OF WORD-CLASSES	PERPLEXITY
10	117
20	110
30	106
40	104
50	102
60	102
70	101
80	101
90	101
100	101
782	106

MAP Clustering on domain adapted models

Table 4.18: (2)MAP Clustering-18688 adaptation sentences

NUM OF WORD-CLASSES	PERPLEXITY
10	114
20	106
30	101
40	99
50	97
60	96
70	95
80	95
90	95
100	95
782	97

As we already know from the theory, when we have to handle a small amount of adaptation data MLLR reacts better than MAP. From the above charts we can observe that when we use all the adaptation data almost always MLLR performs better than MAP. Thus we decided not to repeat these experiments with less sentences. However when we considered each word as a single class MAP performed better than MLLR.

4.3.7 Combinatory Experiments

After experimenting with MLLR and MAP separately we tried to combine sequentially the best adaptation approaches in order to improve the performance of our LM. We tried a great variety of combinations and right bellow we illustrate the best results.

Table 4.19: Combinatory Experiments

EXPERIMENT	PERPLEXITY
Domain MAP	133
Domain MAP/ 60 classes MLLR	92
Domain MAP/ 60 classes MLLR/ 70 classes MAP	90
Domain MAP/ 60 classes MLLR/ 70 classes MAP/Word-Class MAP	88

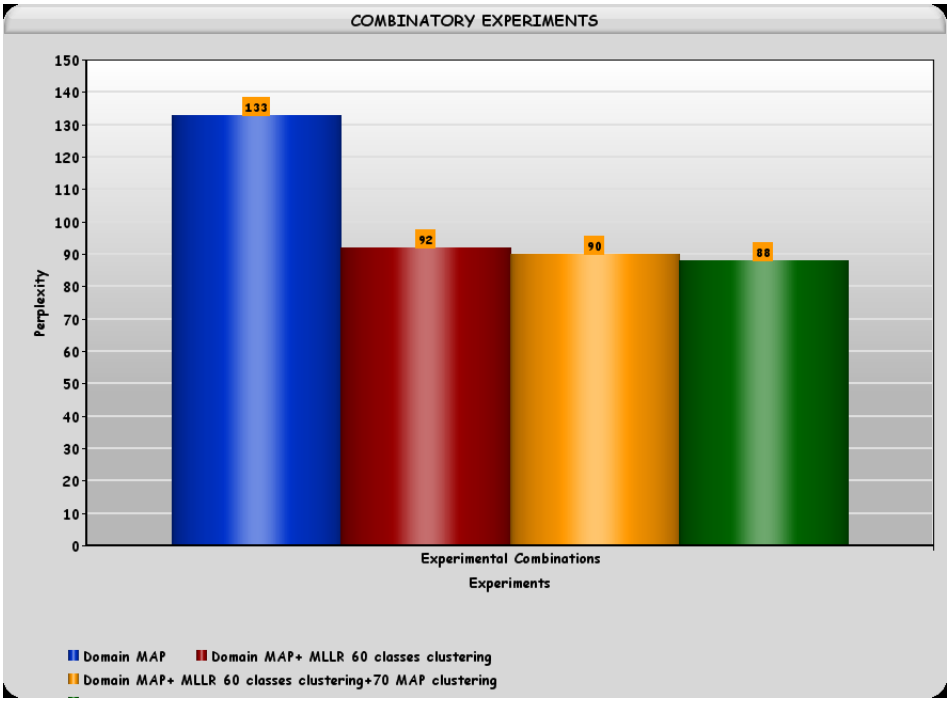

Figure 4.9: Combinatory Experiments

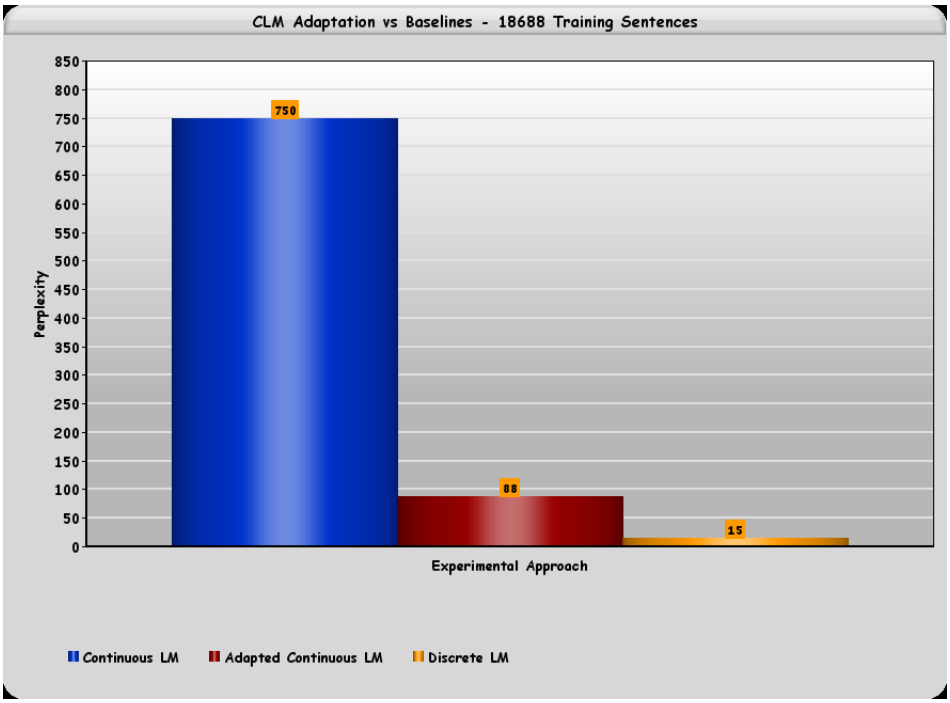

Figure 4.10: Continuous LM Adaptation vs Baseline Experiments

Chapter 5

Conclusion

5.1 Summary of Results & Personal Observations

According to the experimental results, it becomes obvious that adapting a Continuous LM leads to surprisingly better results than training a completely new model with the same amount of data, especially if the available data are extremely limited.

Moreover, as revealed, it is very important that the background model is a stable, well trained model with a sufficient amount of training data. In this case we may be able to overtake possible adaptation problems due to lack of adaptation data by taking advantage of its statistical information.

Regarding the adaptation mechanisms, MLLR revealed as the most appropriate adaptation algorithm as it can provide high efficiency even if we have less than 1000 adaptation sentences. As the adaptation data increase MLLR and MAP tend to coincide (Figure 5.1).

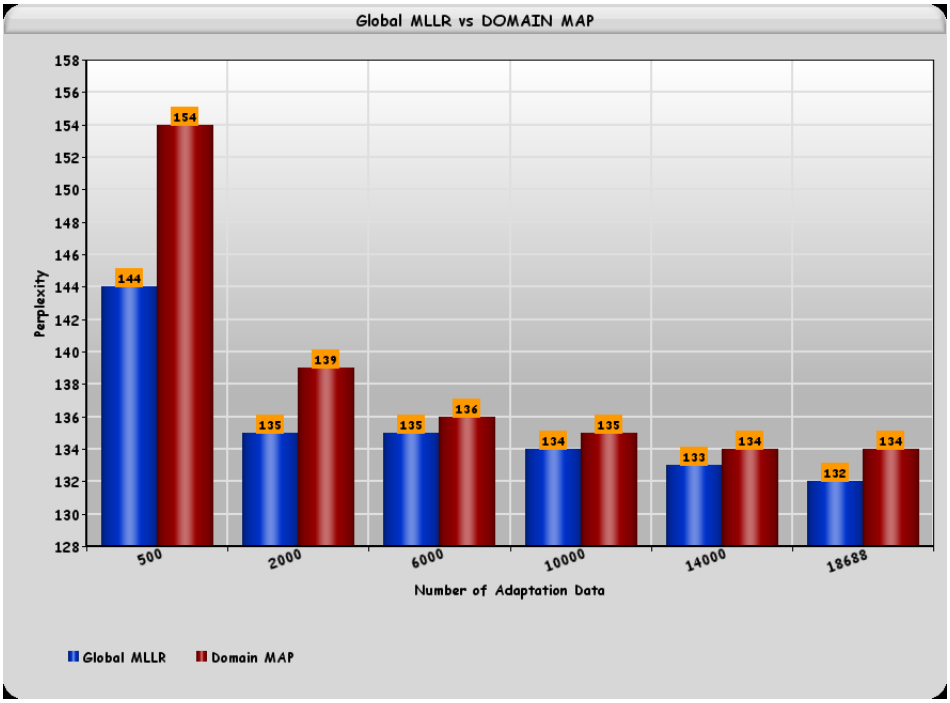

Figure 5.1: Global MLLR vs Domain MAP

As we already knew Tied GMMs model written language more effectively than common GMMs. Thus reestimating only the means of the components and tying the variances together leads to higher efficiency and guarantees extremely faster adaptation as the parameters to be estimated are significantly fewer (Figures 5.2,5.3).

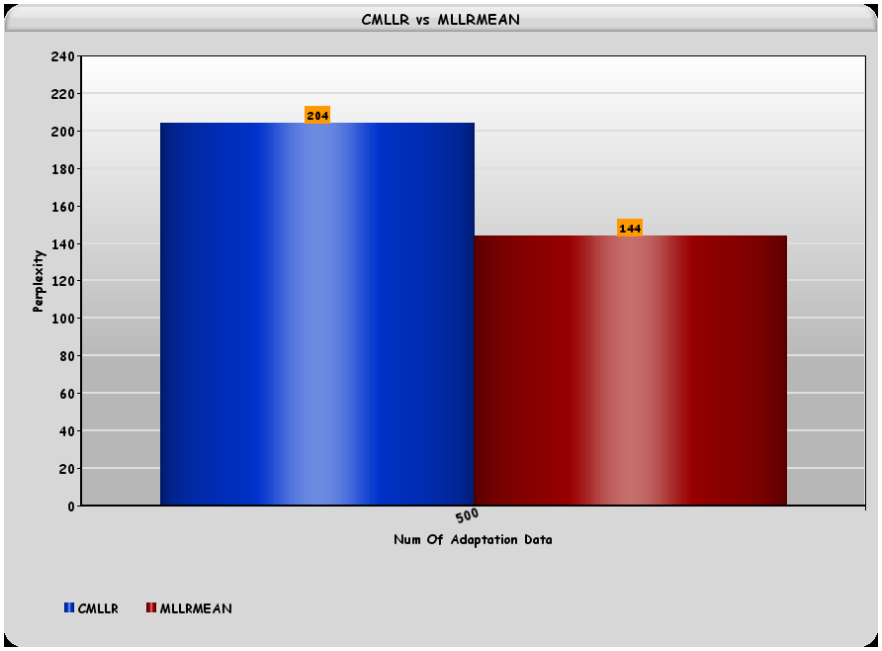

Figure 5.2: (1)CMLLR vs MLLRMEAN - 500 sentences

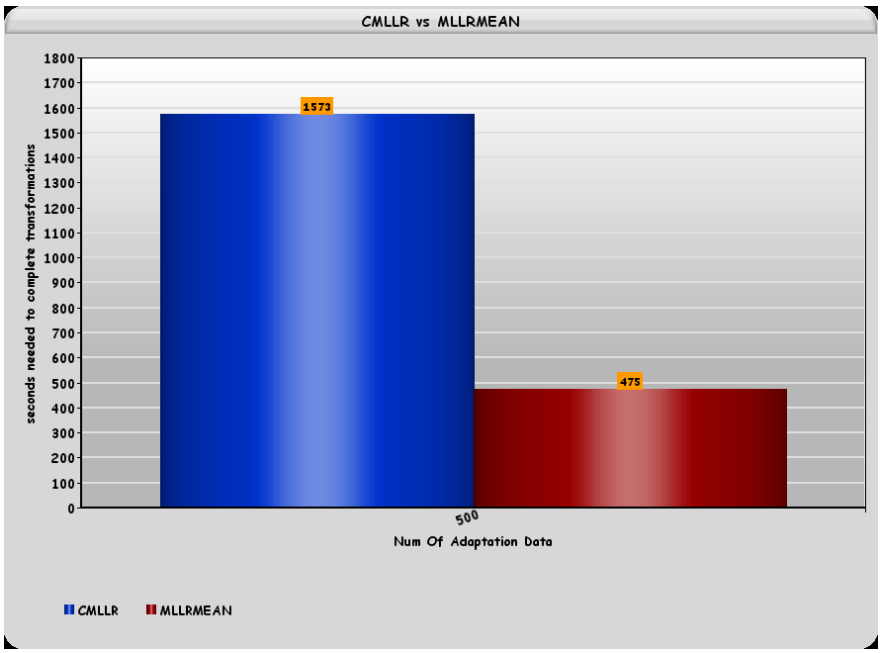

Figure 5.3: (2)CMLLR vs MLLRMEAN - 500 sentences

When the adaptation data are extremely limited the use of the initial model statistics should be favored. Especially if MAP adaptation is preferred. As the amount of adaptation data rises the valuable information that can be extracted from them increases. This keeps happening until a threshold is reached (according to the adaptation domain this threshold has to be found empirically) where the use of the adaptation data statistics favors the adapted model more (Figure 4.8).

Applying different transformations to each GMM proved that can give a great boost to the final models. Especially if each transformation is concentrated to the current word/model. The only condition is that the history vectors which are used for adaptation are plenty enough to improve the model parameters. Performing these separate transformations after applying a global adaptation to all models can lead to even higher efficiency according to the adaptation data available (Figure 5.4).

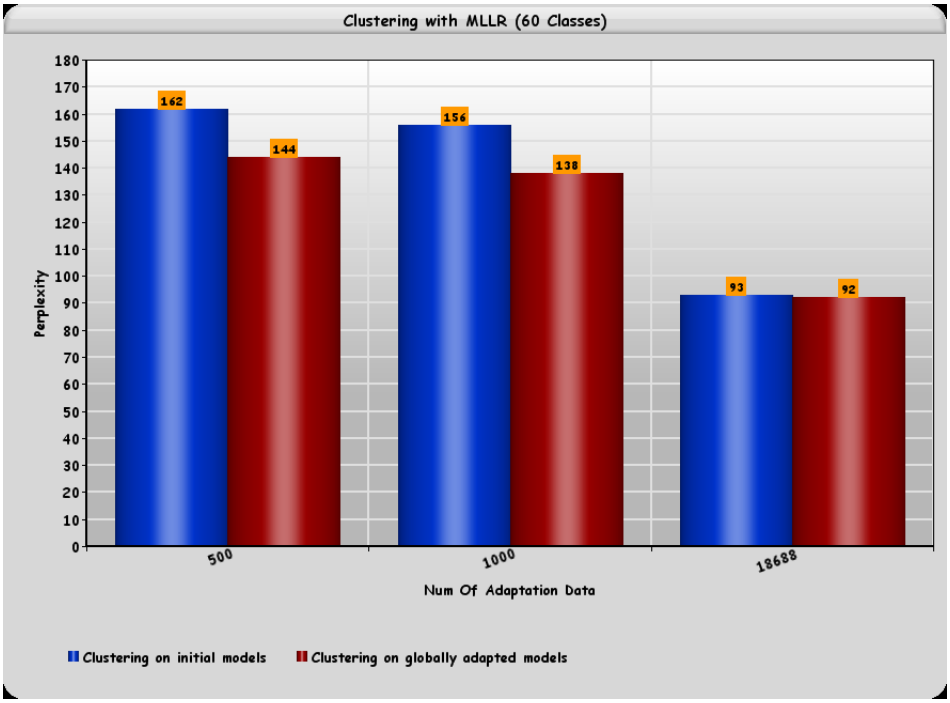

Figure 5.4: Clustering on different initial models

Finally ,as expected , applying serial transformations on a GMM and targeting each time to specify the transformation on the current word/model leads to the best possible results (Figure 4.9).

5.2 Future Work

Unfortunately we did not manage to overtake the efficiency of N-grams which is the dominant technology for language modeling. However we accomplished a great improvement in comparison to the baseline experiments with the Continuous LM.

As already mentioned, this work is an initial approach on this topic. My personal opinion is that higher efficiency can be attained by combining our experimental conclusions with external research.

More specifically I believe that the reliability of the adapted models could be increased by applying the adaptation techniques proposed in this study on alternatively trained initial models.

By organizing the initial models into clusters it is possible to save great computational cost as the transformations needed will be reduced. Under that assumption we could achieve more effective adaptation even for a very small amount of adaptation data as the required parameters to be estimated will be considerably fewer.

Furthermore if we set as first priority the independence between the background and the adapted models, model merging could be a possible solution. By merging the final models we reduce simultaneously the essential parameters. Thus more information can be gathered in less models.

Finally a very interesting scenario is to extend the adaptation vocabulary. This will not cause the increase of the model's efficiency. However it will focalize the final models more to the adaptation domain as there are plenty of words which are very popular in ATIS but do not exist in WSJ and thus in this study they are considered as out-of-vocabulary.

5.3 References

[**1**] *V.Digalakis ,D.Rtischev and L.Neumeyer, "Speaker Adaptation Using Constrained Estimation of Gaussian Mixtures", IEEE Trans. on Speech and Audio Processing, 1995*

[**2**] *M.Afify, O. Siohan and R. Sarikaya, "Gaussian Mixture Language Models for Speech Recognition", ICASSP, Honolulu, Hawaii, 2007.*

[**3**] *Wei Chen, Sanjeev Khudanpur, "Building Language models on continuous space using gaussian mixture models", 2007.*

[**4**] *Berlin Chein, "Introduction to SRILM Toolkit", Department of Computer Science & Information Engineering National Taiwan Normal University, , 2007*

[**5**] *R. Sarikaya, M.Afify, Brian Kingsbury,"TiedMixture Language Modeling in Continuous Space", 2007.*

[**6**] *S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev and P. Woodland, "The HTK Book", Cambridge University Engineering Department, 2006*

[**7**] *S. Geirhofer, "Feature Reduction with Linear Discriminant Analysis and its Performance on Phoneme Recognition", University of Illinois at Urbana-Champaign Department of Electrical and Computer Engineering, 2004*

[**8**] *Holger Schwenk and Jean-Luc Gauvain, "Using Continuous Space Language Models for Conversational Speech Recognition", In ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition, pages 49-53, Tokyo, April 2003.*

[**9**] *Jean-Luc Gauvain and Chin-Hui Lee, "Maximum A Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains", IEEE Trans. on Speech and Audio Processing, vol 2, NO 2, April 1994*

[**10**] *C.J Leggeter and P.C.Woodland, "Speaker Adaptation of HMMs Using Linear Regression", Cambridge University Engineering Department, Cambridge, June 1994*

[**11**] *V.Digalakis and L.Neumeyer, "Speaker Adaptaion Using Combined Transformation and Bayesian Methods", IEEE , November 1994*

[**12**] *Jerome R. Bellegarda, "Statistical language model adaptation: review and perspectives", Speech Communication , 2004*

[**13**] *Jean-Luc Gauvain and Chin-Hui Lee, "Speaker Adaptation Based on MAP Estimation of HMM Parameters", IEEE international conference on Acoustics, speech, and signal processing: speech processing - Volume II Pages 558-561, 1993*

[**14**] *Marc Ferras, Cheung Chi Leung, Claude Barras and Jean-Luc Gauvain, "MLLR Techniques for Speaker Recognition", LIMSI-CNRS & Univ. Paris-Sud, Orsey, France*

[**15**] *C.J Leggeter and P.C.Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models", Cambridge University Engineering Department, Cambridge, Computer Speech and Language, June 1995*