Feature-Based Language Identification Using Data-Driven Model Selection

Sonia Pramod Parandekar

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Electrical Engineering

University of Washington

2003

Program Authorized to Offer Degree: Electrical Engineering

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purpose or by any means shall not be allowed without my written permission.

Signature_____

Date_____

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor Prof. Katrin Kirchhoff, not only for providing me with an opportunity to work in this very interesting field, but also for being a source of inspiration, motivation and encouragement throughout my graduate studies. I would also like to thank Prof. Mari Ostendorf, Prof. Jeff Bilmes and all members of the SSLI group, past and present, for their support and co-operation. A heartfelt thank you to Dipanjan for always being there for me. Finally, I want to thank my parents and parents-in-law for their constant support and encouragement.

Chapter 1

INTRODUCTION

1.1 What is Language Identification?

Among the various factors that define different cultures and communities, an important factor is language. The importance of speech and language for human to human communication can not be over-emphasized. Speech would thus be the most natural medium of interaction between humans and machines, too. With advancements in speech technology, scenarios where humans converse with computer-backed systems to accomplish complex tasks (such as airline ticket reservation, information retrieval), have become a reality. Of course, such systems assume that the system and the human are using the same language. A desirable feature of such systems would be the ability to converse with any user in his or her native or most-preferred language. At the interface of such a multi-lingual system and human users would be a device that can quickly and accurately identify the language the user is speaking. Such a device is called an automatic language identification (LID or language ID) system. An LID system is used to hypothesize the language of an unknown speech utterance.

Multi-lingual speech processing systems are likely to be found in places like information kiosks in airports and hotels, as well as in multi-lingual telephone based systems such as tele-shopping, travel reservations etc. Internet technologies are playing a defining role in the expansion and enhancement of commercial businesses. With rapid advancements in speech technologies, it is possible for their web-based systems to support speech as an alternative medium of interaction between users and the system. If such a system is scaled to support multiple languages, language identification would be the essential first

step before the system can cater to the users' requirements. Personal Digital Assistants (PDAs) and other hand-held devices are gaining in popularity. A desirable feature of such devices would be to support speech in multiple languages; which would require a front-end language identification system. Apart from the front-end of multi-lingual speech recognition systems, many other applications to LID can be cited. LID systems have been used as an interface between users and human operators, for example to route emergency calls to the appropriate operator; such systems could be extremely useful in police and fire departments in areas having highly diverse communities. Such systems are commercially available, for example, the *AT&T Language Line* provides interpreter services in up to 140 languages [3].

Several different approaches to language identification have been studied and a review of research work shows that accurate language identification, while minimizing system complexity and other system requirements, continues to be a challenging research problem. LID accuracy improves with long utterances which provide more information about the language being spoken. However, LID accuracy degrades significantly for utterances with short duration. Considering the nature of the application of LID systems, accurate identification with short utterance lengths is a key requirement. In general, increasing the complexity of the system may lead to improvements in LID results. However, the smaller the system complexity and resource requirements the better. Low system resource requirements is especially important for LID systems on hand-held devices. Obtaining a good LID accuracy rate with a small system continues to pose a challenge.

1.2 Language Identification Systems

The problem of language identification first attracted the attention of researchers in the early seventies, and in the past three decades this problem has been addressed by several researchers in many different ways. LID systems based on spectro-temporal features extracted from speech utterances, prosodic features, differences in phonetic inventories of languages etc. have been studied. The most widely studied approach to LID is the phono-

tactic approach. Phonotactic systems are based on the assumption that each language is characterized by certain statistical patterns in phone sequences that are more likely to occur in utterances of that language as compared to other languages. Speech utterances are mapped to a sequence of phones using some acoustic decoding technique. N-gram language models are trained to encode these statistical regularities of the sequences and scores from these models for an unknown utterance are used to hypothesize its language. The work in this thesis develops a new approach to LID based on articulatory-phonetic features.

1.3 Feature-based LID

Articulatory features of an utterance symbolize properties of different components of the articulatory apparatus when the utterance is being spoken. Each distinct articulatory dimension can be defined as a feature group. For example, *lip-rounding* and *consonantal place of articulation* can be defined as two separate feature groups. In a feature-based system, the speech sequence is mapped to parallel sequences of such distinct feature groups. Analogous to phonotactic systems, language hypotheses are based on statistical patterns observed in sequences within each feature group or stream. Language models are trained to encode these regularities in the form of conditional probabilities between features at different positions in the sequence. For example, a feature value at a given time position can be conditioned on the feature value at the previous time position or two time positions before it in that feature stream. Since feature groups represent different dimensions of articulation of the same speech utterance, it might be advantageous to model such dependencies *across* the feature groups too. It has been shown that it is possible to achieve language identification accuracy with an articulatory-phonetic feature-based system which is comparable to or better than state-of-the-art phonotactic systems [16, 17]. Besides, a feature-based system has the advantage of having much lower system complexity and the parallel framework of a feature-based system offers several potential advantages. Apart from providing a richer encoding of the speech utterance, it allows for easy integration of other sources of information that can be represented as a sequence of discrete symbols,

for example, a stream of prosodic symbols. Furthermore, language-specific information in dependencies between all the feature groups or streams under consideration can be exploited by modeling cross-stream dependencies.

Identifying the set of cross-stream dependencies that maximizes language identification accuracy poses a challenging search problem. Two specific aspects of this search problem that make it challenging are:

- The search space defined by all possible combinations of within-stream and cross-stream dependencies is huge and cannot be searched exhaustively.

- The ideal set of dependencies for LID is a complex function of several factors, such as the nature of the LID task (type and number of languages being considered), speakers, signal quality, utterance lengths etc.

For these reasons, selecting dependencies in a data-driven fashion seems to be appropriate. The search space defined by all combinations of dependencies is multi-dimensional and bound to be multi-peaked. Conventional search techniques, when applied to this problem are likely to converge on one of the local optima. Hence, we chose to apply the Genetic Algorithm (GA) to this problem. GAs are highly robust search algorithms that have been successfully applied to complex search problems across a wide range of fields. Since they simultaneously process several solutions in the search space and are based on probabilistic transition rules, they tend to be highly robust and are not likely to converge on local peaks in the search space. We use a GA based search to identify the ideal set of within-stream and cross-stream feature dependencies for language identification. The primary focus of this thesis is to develop a feature-based LID system using data-driven model selection.

1.4 Experimental results and summary

We have developed a novel approach to language identification based on articulatory-phonetic features. LID performance of such a system is comparable to that of the standard

phonotactic approach. Furthermore, an AF based system has a much smaller parameter set. This could be especially significant for LID systems with limited power, memory and computational resources.

We have shown that significant improvements in language ID performance are possible by explicitly modeling dependencies across different feature streams. Identifying the ideal model-set of within-stream and cross-stream dependencies for language ID is an NP-hard search problem. Due to the huge search space defined by all possible combinations of dependencies, an exhaustive search is not possible. We have shown that a good set of dependency models for language ID can be discovered by employing a genetic algorithm based search. A GA based search out-performs other heuristic search methods. In general, the ideal model set for the given data is a complex function of several factors such as languages in the set, n-gram models, size of the data set and its characteristics, utterance lengths, combination function etc. and it is not possible to make a knowledge-based estimate of this model-set.

We have developed a generalized framework in which parallel streams of information about a speech utterance are combined together to perform language ID. The system is scalable to further incorporating any other sources of information that can be expressed as sequences of discrete symbols, such as prosodic symbols. The approach we have developed here can be applied to any problem that uses parallel streams of information and can be used to effectively model the dependence relationships between these streams, given that the streams of information are not independent. For example, such a system can also be applied to a speaker identification system based on a similar approach.

Chapter 2 provides an overview of different approaches to LID that have been studied so far. That is followed by a discussion of articulatory-phonetic feature-based LID in chapter 3. In chapter 4 we discuss Genetic Algorithms, followed by an explanation of our baseline system in chapter 5 and a description of our experiments and results in chapter 6.

Chapter 2

INTRODUCTION TO LID

2.1 Introduction

A language identification system is used to hypothesize the language of an unknown speech utterance. Many different factors can be considered for differentiating between languages or identifying a language. These range from spectral features of the acoustic speech signal to language-specific words or phrases. Research in language identification first started over thirty years ago, and since then, different approaches based on all these factors have been developed. The nature and design of LID systems has also been greatly influenced by advancements in technology in general and speech technology in particular. While advancements in technology have made more and more complex systems feasible, some LID systems are based on techniques that have proved effective in related speech problems, such as speech recognition and speaker identification. The remainder of this chapter provides an overview of the most important approaches to LID.

2.2 Different approaches to LID

There are several different factors that can be considered when identifying different languages. These include:

- spectro-temporal acoustic features extracted from the speech signal
- prosodic features such as the pitch, duration and intensity of speech
- the phonetic inventory, i.e. the list of most commonly occurring phones
- variations in acoustic realizations of common phonemes across languages

- phonotactic patterns, i.e commonly occurring patterns in symbolic phone sequences
- vocabulary, i.e. words used in the language

Human beings possess an inherent capability to instantly identify a language that they know and also of guessing the identity of languages that they do not know, depending on their background. Automatic LID might benefit from insights into the type of information that humans use for language identification. Such studies in human language identification have been reported and bring to light some interesting aspects of human LID [1, 2]. The study in [2] investigates the relationship between human LID performance and factors like prior exposure to languages or academic linguistic knowledge. Contrary to speculations, the study concludes based on experimental results, that previous exposure to a language does not necessarily translate into success in identification of that language. Furthermore, a high level of linguistic expertise does not lead to better performance in language ID. It is interesting to note that the majority of errors in human LID can be attributed to confusions between pairs of related languages, such as Amharic and Arabic, which both belong to the class of Semitic languages, which includes languages spoken in the Middle Eastern region. In [1] different cues used by individuals to identify languages were studied. The choice of cues varied across individuals and across languages and it was difficult to determine the set of cues used by humans to identify languages with which they were completely unfamiliar. Some of the cues used by humans to identify languages included word-spotting as well as phonetic and prosodic cues. For example, some subjects used the the word *watashiwa* to identify Japanese and *iminida* for Korean. Some subjects reported using the "*sing-song*" intonation of Vietnamese as a cue while some used the frequent occurrence of phoneme /sh/ for identifying Farsi. Encoding these cues into a formal paradigm could benefit an automatic language identification system.

The core of every LID system consists of a set of language dependent models. The realization of these models and any pre-processing and post-processing steps that may be required depend primarily on the choice of language-discriminating features to be ex-

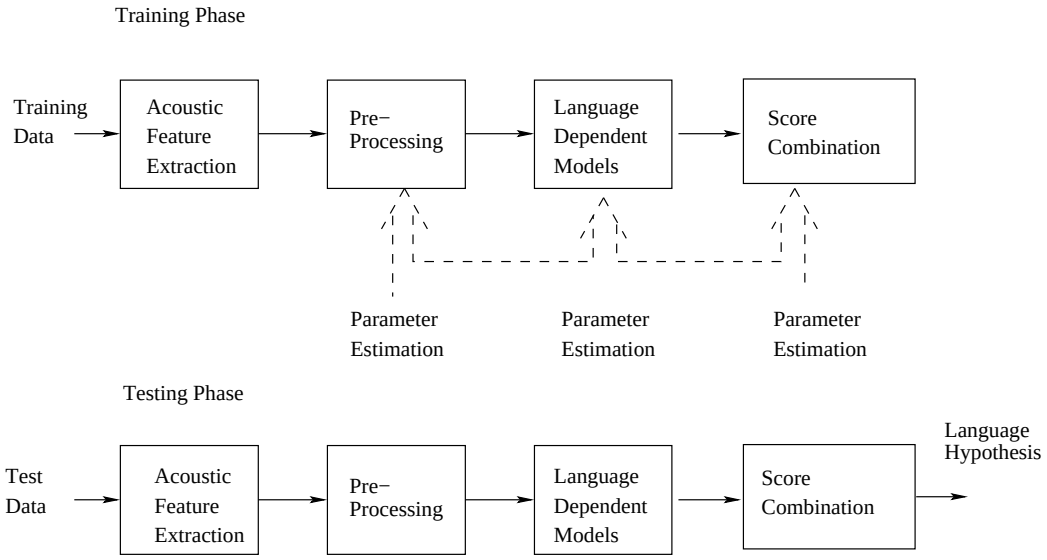

Figure 2.1: Phases in implementing an LID system

ploited by the system. Secondary considerations include desired system complexity and availability of resources. In the first step, model parameters are estimated using training data for the different languages. In the second step, these models are used to score a set of test utterances, and language of the test utterance is identified based on the the scores output by these models. Usually, a held-out development set is used to optimize performance of the LID system, before applying it to the test set, also known as the evaluation set. These two phases in implementing LID systems are illustrated in Figure 2.1. In the following sections we will look at various realizations of LID systems in greater detail.

2.2.1 Acoustic LID systems

The earliest studies in LID were reported in the early '70s [4] and relied on differences in acoustic features to perform LID. This system was based on a template matching algorithm, where spectral features from the test speech were compared with prototypical spectral features from the training data of different languages. More advanced systems based on acoustics used LPC (linear predictive coding) features for LID [5, 6]. In [6], LID experiments using variations of Vector Quantization (VQ) techniques on LPC cep-

strum coefficients are reported. Two LID systems, one based on language-specific VQ codebooks and the other based on a universal VQ codebook and language specific occurrence probability histograms were studied and the latter technique was shown to be superior. The study also investigated different LPC spectral distortion measures such as LPC cepstrum distance, weighted likelihood ratio, frequency weighted cepstrum distance etc. Other studies based on formant feature vectors extracted from the acoustic signal have also been reported [7]. Another approach to language identification based on acoustic feature vectors using Gaussian Mixture Models (GMMs) for classification can be found in [8]. In this study, feature vectors formed by 12 mel-cepstral and delta mel-cepstral coefficients each were considered. RelAtive SpecTrA (RASTA) pre-processing was applied to remove channel effects. Every language was assumed to be characterized by multi-variate Gaussian density functions and the feature vectors were assumed to be drawn from the weighted sum of these densities. For each language, GMM parameters, namely, mixture weights, means and variances were estimated from the training speech data for that language using the Expectation Maximization (EM) algorithm. During recognition, the log likelihood of a test utterance with the GMM models was calculated and its language was identified using the maximum-likelihood hypothesis, i.e. the language of the GMM model that assigned the highest probability to the utterance was hypothesized as the language of the unknown utterance.

2.2.2 Phonotactic LID systems

The first study in exploiting phonotactic patterns for language identification was reported in the late seventies in [9]. Phonotactic constraints define the legal sequences of phones that can occur in a language. Phonotactic constraints vary greatly across languages. For example, the phone cluster /sr/ is very common in some Indian languages (like Tamil) but is not allowed in English. A phone can be defined as the acoustic realization of a phoneme. A phoneme is the smallest phonetic unit in a language that is capable of conveying a distinction in meaning, for example, the *d* in dog and *f* in fog are two different phonemes. The study in [9] investigated the feasibility of LID systems based on phonotactics and

concluded that statistical constraints on sequences of broad phonetic classes of a language can be used successfully in language identification. Due to the lack of resources at that time, the study worked with artificial data (orthographic transcriptions not derived from acoustic decoders) and the same data was used for training as well as testing. Utterances were decomposed into five broad phonetic classes, such as *vowel* and *stop consonant*. Hidden Markov Models (HMMs) were trained for different languages, one per language and models having different numbers of hidden states were investigated. During testing, the log probability of an utterance for each model was computed and it was identified to belong to the language whose model obtained the maximum score. Though the same data set was used for training and testing, it was shown that for any utterance, no other language scored higher than the true language of the utterance. Since this study, there have been several successful attempts in building phonotactic LID systems, some of which are discussed below.

The phonotactic approach continues to be the most popular approach to language identification. Much more complex schemes have been reported [10] and several enhancements have been proposed [11] to phonotactic systems. Various techniques to improve pre-processing of the speech signal, acoustic decoding, language modeling and score combination have been studied.

The acoustic signals are pre-processed to compensate for noise and channel effects and feature vectors are extracted. The feature vectors may be in the form of LPC coefficients, PLP (Perceptual Linear Prediction) coefficients or MFCCs (Mel Frequency Cepstral Coefficients). Techniques such as cepstral mean subtraction and RASTA provide an efficient way to remove the effects of the transmission channel (such as telephone line and microphone). Variance normalization is another technique that improves noise robustness.

Phone recognizers based on HMMs or artificial neural networks (ANNs) [21] are used to map these feature vectors to a sequence of phone symbols. Individual HMMs are trained for each phone from labeled training data; since an HMM models the temporal

realization of a phone, left-to-right topologies in state sequence structures with possible skips are typically used. Since sufficiently large amounts of transcribed data may not be available, these models may be initially bootstrapped on available transcribed data. Using these models, automatic transcriptions of remaining training data are generated and the models retrained on the automatic transcriptions [16]. During recognition, the Viterbi search algorithm is used to map test speech signals into phone sequences. The number of inputs in an ANN classifier typically equals the number of coefficients in the acoustic feature vector, while the number of outputs equals the number of phone classes. The ANN architecture is multi-layered, with input and output layers and one or more hidden layers.

Phone-based LID systems can be broadly categorized based on whether they use language-dependent (LD) or language-independent (LI) recognizers. In language-independent or multi-lingual recognizers, training data from all languages is pooled together to train recognizer parameters. When large amounts of training data per language are available, language-dependent recognizers may be opted for. Though the latter need significantly more resources during training and testing, such systems can exploit the acoustic-phonetic differences in languages to a higher degree. A phonotactic system using LD phone recognizers (called P-PRLM - Parallel Phone Recognition followed by Language Modeling) was compared to an LID system using a single phone recognizer trained on only one language (called PRLM - Phone Recognition followed by Language Modeling) in [10] and the former was shown to outperform the latter over all test utterances. However, a study comparing phone-based LID using LD recognizers with one using LI recognizers showed that though the former outperformed the latter for short test signals (10 sec.), the LI based system gave better LID accuracy for long utterances (45sec.) [13]. Furthermore, appending the LI recognizer to the bank of LD recognizers improved LID performance for all utterances. Output of the acoustic decoders is fed into a bank of language models that encode language-specific phonotactic constraints. Language models are used for scoring test utterances. Language hypotheses by the system are based on the scores output by these language models, hence they are a crucial component of the LID system.

Different categories of language models include [44]:

- n-gram models
- decision trees
- linguistically motivated models
  - context free grammars (CFGs)
  - link grammars
- exponential models

Of these, n-gram models have been the most commonly used models in almost all aspects of speech technology. In all our experiments, we have used only n-gram language models and the two terms are henceforth used interchangeably. An n-gram language model is a collection of probability distributions that predict the probability of occurrence of an utterance in that language. Parameters of n-gram models are estimated based on counts for different n-grams computed from the training data for that language, and constraints based on grammatical structure etc., are not considered. An n-gram model predicts the probability of a word (or phone for a phone-based n-gram model) given the previous $n-1$ words (or phones) in the utterance and is based on the assumption that speech generation is a time-invariant Markov process. Thus, the probability of a phone sequence $\phi_1\phi_2....\phi_N$ given an n-gram language model L is computed as:

$$P(\phi_1, \phi_2, ..., \phi_N|L) = \prod_{i=n}^{N} P(\phi_i|\phi_{i-1}, ..., \phi_{i-n+1}, L) \tag{2.1}$$

The language model that assigns the highest probability to the test utterance is hypothesized as its true language, i.e.

$$L^* = argmax_L P(\phi_1, \phi_2, ..., \phi_N|L) \tag{2.2}$$

In [11], an alternative phonetic-phonotactic LID system is also presented, which uses integrated LD acoustic-phonotactic models, i.e. the n-gram model for a language is integrated

with a language-dependent phone recognizer for that language and log likelihood scores from this system are directly used for LID.

The feasibility of an LID system based on a single LI recognition module has also been studied in [14]. Phonemes from all languages under consideration were clustered together based on minimum Euclidean distance and from amongst these clusters, a subset of language-independent broad phonemes was manually selected based on criteria such as frequency of occurrence, number of languages of occurrence and degree of confusion with other phonemes. A neural network based classifier was used to identify the broad phoneme classes. Language hypotheses are based on phoneme occurrence statistics of the phoneme sequence output by the neural network classifier. The motivation behind using a single LI recognizer was:

- Many phonemes are shared across languages, thus creating redundancy in training separate recognizers.

- Though a system based on LI recognizers does not model language specific phonemes, many language specific statistical patterns are observed in the sequences of broad phonemes that can be shared across languages.

- An LID system based on language independent recognizers is much smaller and has far fewer parameters than an LID system with language dependent recognizers.

2.2.3 Segment-based and prosodic LID systems

Another approach to LID motivated by [9] is segment-based language ID. Instead of fine phonetic symbols used in phonotactic systems described above, the segment-based approach extracts and identifies segments belonging to broad phonetic classes from the acoustic signal. An interesting feature of these segment-based system is that it integrates language-specific acoustic, phonotactic and prosodic models in a probabilistic framework [18, 19]. In [18, 20], a phonetic recognizer is used to find the best phonetic hypothesis

and segmentation from the pre-processed acoustic signal. The fine-grained phonetic symbols output by the recognizer were then collapsed back into broader phonetic classes. In an earlier study, a neural-network based segmentation and broad phonetic classification algorithm is described in detail [19]. The phonotactic component in [18, 20] is similar to the one previously described in phonotactic systems. The acoustic model for a language is characterized by a set of GMMs. The prosodic model looks at prosodic information specific to a segment and gathers information about fundamental frequency and duration of the segment. The model is further simplified by assuming the two to be uncorrelated. During language identification, a weighted sum of individual scores from the three models is considered. The LID performance of individual systems and the combined system were studied with the conclusion that though the phonotactic model significantly outperforms all other models, combining all three models is advantageous, especially for short duration utterances. In another segment-based system, more complex prosodic features such as fundamental frequency variation across segments and syllabic timing were considered and were found to be only marginally useful [21]. In general, it is believed that prosodic cues could be a vital source of information for language identification, but they have not been very successfully implemented in LID systems, mainly because of the unavailability of techniques for reliably and usefully extracting prosodic features from acoustic signals [1, 20]. This could be because prosodic cues such as stress, rhythm and intonation are encoded in physical properties of speech such as pitch, amplitude and duration in a complicated manner [22].

Studies employing only prosodic features for LID have also been reported [22, 23]. In [22] pitch and amplitude contour information is extracted from the speech signal at a syllable level; inter-syllable pitch and amplitude statistics are also computed. A total of 224 prosodic features are defined to capture the pitch and amplitude contour information at the syllable level. Smaller subsets of individual features or feature pairs are used during training and recognition. In [23] a recurrent neural network is used to determine the subset of features to use for LID. Both studies conclude that pitch is a more effective factor than

amplitude contour for LID. The studies also confirm that prosodic features are ideal for enhancing an LID system based on other approaches rather than building a stand-alone prosody-based LID system.

2.2.4 LID systems motivated by ASR and speaker identification technologies

A considerable amount of research effort has been invested in the fields of speech recognition and speaker identification/verification. Parallels can be drawn between these tasks and the task of language identification, and many approaches to LID motivated by research in these fields have been proposed.

LID systems based on higher level lexical information

An important application of LID systems has been as the front-end of multi-lingual speech recognition systems. Such a system possesses word-recognizers, dictionaries, language models etc. for each of the languages it supports. LID systems taking advantage of these higher-level knowledge sources have been proposed [24, 25]. Such systems are commonly classified as Large Vocabulary Continuous Speech Recognition (LVCSR) based LID systems. In [24], word-based lexicons and grammars were integrated into the LID system based on the fact that these components are important for speech recognition performance. Five different systems, starting with language dependent phone-recognizers were studied, each incorporating a greater degree of higher-level knowledge. In the final system, a word-based recognizer including a pronunciation lexicon and word based language model for each language were implemented. Results showed that word-based systems out-performed phone-based systems, and across the five systems, increasing the degree of higher-level linguistic knowledge in the system always resulted in better LID performance. Experiments also illustrated the importance of having similar recording conditions for different languages.

An interesting approach to LID combining phonotactic and lexical information can be found in [26]. In this system, hybrid phone and word n-gram models are integrated into

the language-dependent acoustic decoders to provide linguistic constraints. Only the N most frequent words in each language are considered. Though incorporating higher-level knowledge sources is known to be beneficial to LID systems, such systems are computationally very expensive and also, gathering higher-level knowledge for all languages under consideration is very difficult. Just the N most frequent words in a language were considered because the study found that the most frequently occurring words account for a large subset of the entire data. The hybrid n-gram models were estimated using standard estimation techniques which are discussed in detail in chapter 3. The study also investigated other related issues such as LID performance on spontaneous vs. read speech and LID performance on fixed vocabulary utterances. Results for different values of N ranging from 100 to 500 have been reported. The study concluded that incorporating word-level lexical information in the phone-based system significantly improved LID accuracy. As expected, LID results with read speech were much better than spontaneous speech.

LID based on syllabic features

Language identification based on syllabic features extracted from speech utterances for individual speakers for each language has also been investigated. None of the LID systems discussed above take into account speaker variability. It is assumed that the various model parameters used by these systems are speaker independent. In general, speaker and language differences within an acoustic signal are difficult to separate. In [27], a closer look is taken at this and related issues. A direct correlation between speaker similarities and language identification is assumed and speaker characteristics of unknown utterances, in terms of the syllabic nuclei, are compared with the most similar speakers in the target languages, The baseline system is based on a speaker identification system using the nearest neighbor algorithm. Syllabic nuclei are extracted from the training utterances using an artificial neural network. Spectral coefficients for these nuclei are estimated and stored for every speaker in each language. During testing, spectral coefficients from the test utterance are extracted in a similar fashion and are then compared to coefficients of each speaker in the training set. The average difference with the S most similar speakers in

each language is considered and the language with the smallest difference is hypothesized as the true language. The value of S is optimized for maximizing LID accuracy. The study concludes that syllabic spectral features used in such a framework are suitable for LID.

2.2.5 Enhancing LID systems

In the previous sections, the motivation and basic implementation of different approaches to language identification have been discussed. Strategies and techniques to further enhance LID systems have also been studied, some of which are outlined below. As in the case of LID approaches, a few of these techniques draw inspiration from their success in speech recognition tasks.

Higher-level classifiers for score combination In phonotactic LID systems using N language dependent phone recognizers, each followed by a bank of language dependent n-grams, N log probability scores per language are output for each test utterance. These are summed up to produce the final score for each language and the language which has the highest final score is assigned to the test utterance. However, such a scheme assumes the phone sequences for a language to be independent given the language, which may not be a valid assumption. In [28] and [11], outputs from all the n-gram models in the system are aligned together in a feature vector. Multi-variate Gaussian models are trained on these feature vectors, one per language. During evaluation, a similar feature vector is computed for the test utterance, and the language whose Gaussian model yields the highest likelihood score is assigned to the test utterance. Parameters for the Gaussian models can be optimized on a held-out set. Other higher level models, such as Artificial Neural Networks (ANNs) or decision trees can also be employed.

Gender-dependent models: Gender dependent acoustic modeling leads to considerable improvements in speech recognition performance. There are significant differences in the acoustics of male and female speech and with gender-dependent acoustic decoders,

more reliable tokenization of the input speech may be possible. This idea has been imported into LID systems [11]. Along with gender-independent models, gender dependent acoustic and n-gram models are trained for each language. Scores from each type of n-gram model (i.e. male, female or independent) are weighted according to the acoustic likelihood score from the corresponding acoustic decoder.

Other enhancement techniques that have been successfully been implemented in LID systems include:

- *Phone duration length modeling* [11]. Duration statistics are compiled and phone symbols that are much longer or much shorter compared to the average duration for that phone are tagged accordingly and incorporated into the n-gram models.

- *Speech Non-Speech Segmentation* [26]. Speech detection techniques are used to segment the utterance into speech non-speech units. Using just the speech segments thus identified for LID can be more effective than using the whole speech signal.

2.2.6 Recent studies in LID

A few of the latest research efforts in language identification will be discussed in this section.

LID with GMM tokenization

In [29], the HMM based phone recognizers are replaced by GMM based tokenizers. The proposed LID system is similar to a phonotactic LID system in all other respects. The GMM based tokenizer takes pre-processed acoustic speech signals for input and maps them into the appropriate partition of the acoustic space. The GMM is trained on feature vectors from acoustic signals of just one language. During testing, each frame in the test utterance is assigned an index based on which GMM component yielded the highest score for that language. Sequences of such indices from the training data of a language are used for training n-gram models. Final language ID is based on output from an LDA

classifier rather than using the simple likelihood maximization rule. GMM tokenizers have many advantages over conventional phone-recognizers, such as: they do not require transcribed data, they are computationally inexpensive and are easier to combine with additional tokenizers in parallel. Finally, combining scores from GMM based and phone recognizer based LID systems is shown to outperform either of the two systems considered in isolation.

GMM based LID system using VTLN

In [30] different techniques to enhance performance of a GMM based system are proposed. Most LID systems assume that their language dependent models are speaker independent. However, to make these models truly speaker independent, some form of speaker normalization of the acoustic features is required. Speaker adaptation based on maximum likelihood linear regression (MLLR) technique has been widely studied. MLLR is a transform based technique that adapts the model parameters for individual speakers, where the transformation function is estimated using adaptation data for each speaker. This technique thus requires some amount of adaptation data and transformation of *all* model parameters using the transformation function. Hence, it is not suitable for LID systems since LID systems require a fast response time with short utterances. In comparison, vocal tract length normalization (VTLN) for speaker normalization requires the estimation of just one parameter per speaker, namely α, and normalizes the feature vectors rather than adapting model parameters. Hence, VTLN might be more suitable in the context of LID systems and is applied to a GMM based LID system in [30]. The vocal tract length of a person is inversely proportional to the formant frequencies found in his or her speech. Using VTLN, a normalization factor α is identified for each speaker that normalizes speech for that speaker by re-scaling the frequency axis. Though the normalization factor is not directly related to the physical length of the vocal tract, the normalized speech achieves a higher likelihood score. Furthermore, the universal background model (UBM) technique, which has been successfully applied to speaker identification systems, is applied to the GMM based language ID system. A universal background GMM that

models characteristics of all the languages in the system is trained on training data from all the languages. Language dependent GMMs are trained for individual languages from the UBM with Bayesian adaptation using training data for that language. This technique takes significantly less amount of time as compared to maximum likelihood based training techniques. Finally, scores from such a GMM-UBM system are linearly combined with scores from a P-PRLM phonotactic system. LID results for the GMM-UBM system and the combined system are reported. The GMM-UBM system performs better on long utterances (45 sec.), while the P-PRLM system is more accurate for short utterances (10 sec.). The combined system significantly outperforms both systems for short as well as long utterances.

2.3 Resources, benchmarks and evaluation methods

In the previous sections, many different LID systems have been discussed and compared, however, LID accuracies of none of these systems have been reported. The main reason for this is that most of these studies have worked with different LID tasks. LID tasks can differ in various respects, such as:

- size of the corpus
- number of languages
- type of languages
- recording conditions (such as noise and channel effects)
- speaking styles: spontaneous vs. read speech
- average utterance lengths

Just as it would be unfair to compare two LID systems trained on corpora with disparate sizes, similarly, recording conditions (speech recorded in a laboratory vs. speech collected

Table 2.1: LID accuracy (%) and other features for some LID systems

LID System	Languages + Corpus	Utt. Length	% LID Acc.
Acoustic features [6]	20 languages	8 sec.	36
Segment based [19]	5 languages	4 sec. (average)	81.5
GMM based [8]	OGI-TS	10 sec.	50
PRLM [10]	OGI-TS	10 sec.	54
P-PRLM [10]	OGI-TS	10 sec.	63
P-PLM + duration modeling + gender modeling [10]	OGI-TS	10 sec.	79
Lexical info. based [26]	IDEAL (4 languages)	5 sec. min.	87.6
LVCSR based [24]	SST (4 languages)	-	84

over telephones), speaking styles (read speech vs. spontaneous speech containing more disfluencies, cross-linguistic terms, co-articulation effects etc.) and language type also play an important role. LID results for systems built using the different approaches discussed in the previous sections are summarized in table 2.1. The table contains the best results reported for the respective systems. Other relevant information about the system, such as number of languages, average utterance length etc. are also provided.

In general, languages can be broadly classified as syllable-timed or stress-timed. In syllable-timed languages, similar amount of time is taken to articulate each syllable, hence,

the amount of time taken to say a sentence is proportional to number of syllables in the sentence. In a stress-timed language, syllables may be stressed or unstressed. The amount of time required to say a sentence depends only on the stressed syllables. Examples of syllable-timed languages include Japanese, French, Italian, while stress-timed languages include English, German and Russian. Considering this and other similar categorization, LID tasks over different sets of languages will vary in difficulty. In many of the studies discussed above, it has been reported that LID performance improved significantly with increase in utterance length. At the same time, for all LID applications (front-end of multi-lingual systems, emergency call routing etc.) minimal processing time is essential. Thus, utterance length is an important consideration.

2.3.1 The OGI-TS corpus

To make it possible to directly compare and evaluate different approaches to language ID, a multi-language telephone based corpus was collected and made available in the public domain [31]. Known as the OGI-TS corpus, it contains utterances from 10 languages (English, Farsi, French, German, Japanese, Korean, Mandarin, Spanish, Tamil, Vietnamese) collected through telephone calls. The corpus contains fixed vocabulary, topic-specific as well as spontaneous speech with up to 9 calls per speaker. For example, the callers were asked to enumerate days of the week and numbers from zero to ten. In topic-specific questions, callers were asked to speak about the climate in their home town and describe their most recent meal. Finally, each caller was asked to speak on any topic of his or her choice. The utterance lengths vary from 3 sec. to up-to 1 min. The corpus contains approximately the same amount of data per language though the male to female ratio is different for each language. Most studies in LID are based on this corpus or some subset of this corpus.

2.3.2 The Callfriend corpus

In 1996, a larger multi-lingual corpus containing recorded conversations between friends and family in 12 different languages was collected and is available with the Linguistic Data

Consortium (LDC) [32]. The languages in this corpus include Arabic, English, Farsi, French, German, Hindi, Japanese, Korean, Mandarin, Spanish, Tamil and Vietnamese. The conversation lengths range from 5 to 30 minutes. Information about gender of caller and callee, quality of the line, background noise and dialect information is available for most calls. The CallFriend corpus was used in the language identification evaluations conducted by NIST in 1996.

2.4 Summary

In this chapter, different approaches to LID ranging from acoustic systems to LVCSR based LID systems have been discussed. Resources in terms of available corpora and evaluation schemes were also presented. In the next chapter, we discuss the theoretical and probabilistic framework behind an articulatory features based LID system.

Chapter 3

FEATURE-BASED LID: THEORETICAL FRAMEWORK

3.1 Introduction

In the previous chapter, different approaches to language ID have been discussed. This thesis presents a new approach to automatic language identification based on articulatory-phonetic features. These features represent articulatory properties of the speech utterance such as information about consonantal place of articulation, lip rounding etc. A feature-based LID system can be represented in the same probabilistic framework as a phonotactic system, the main difference being that instead of a single phone sequence, parallel streams of feature sequences are considered. In a phone based system, languages are distinguished based on differences in statistical patterns in their phone sequences. In a feature-based system, statistical patterns within a feature stream as well as patterns occurring across different feature streams are considered. We will discuss these ideas in greater detail in the following sections.

3.1.1 Articulation

Articulation, in the present context, can be defined as the act or manner of producing speech sounds. More specifically, articulatory properties of a speech utterance are abstract classes that reflect the actual physical configurations of various articulatory organs during speech production. These organs include lips, teeth, tongue, laryngo-pharynx etc. The movements of these organs relative to different parts of the vocal tract define the articulatory properties of speech. For example, the tip of the tongue touching different parts of the hard palate (alveolar, post-alveolar and palatal regions) and the soft palate (velum and uvula) define different articulatory gestures. Articulatory features that we have considered

are loosely based on the vocal tract configurations because no actual physical measurements are done. Absolute physical measurements would be specific to the characteristics of the articulatory organs of that speaker and it would be difficult to generalize these values across all speakers, since there is considerable variation in the physical dimensions of vocal tracts of people. Instead, articulatory properties are characterized by a small set of quantized values that represent different degrees of movement along specific dimensions of articulation, for example, *+voice* for voiced speech and *-voice* for unvoiced speech.

3.1.2 Articulatory-phonetic features:

Different feature systems have been developed to describe as well as classify speech sounds in languages across the world. The *distinctive feature system* developed by Chomsky and Halle is based on physical movement of speech organs such as the tongue, lips and vocal chords, during speech production [36]. It defines a set of 14 binary valued features, for example, anterior-non anterior (sounds produced with or without an obstruction located in front of the palato-alveolar region. /p, t/ can be classified as anterior while /k/ would be a non anterior phone). The distinctive feature set is used mainly for *classifying* sounds that occur in languages. An alternative feature system that can be used for classifying as well as describing phonetic sounds is the *traditional feature system* described in [36]. It includes articulatory as well as a few acoustic features and allows for multi-variate feature values. The articulatory features included in this system are:

- Glottalic - Rate of upward movement of glottis
- Velaric - Degree of suction of air in mouth
- Voice - Glottal stricture
- Aspiration - Onset of voicing w. r. t. release of articulation
- Place - Distance from glottis to the first constriction of the vocal tract

- Labial - Degree of approximation of centers of the lips
- Stop - Degree of approximation of the articulators
- Nasal - Degree of lowering of the soft palate
- Lateral - Amount of air stream flowing over the side of the tongue
- Trill - Degree of vibration of an articulator
- Flap - Rate of articulatory movement
- Round - Lip rounding
- Wide - Degree of advancement of tongue root

Apart from these features, the traditional feature system also includes features based on acoustic properties such as *sonorant*, which pertains to the amount of acoustic energy and *silibant*, which is based on the amount of high frequency energy in the sound.

Though all these features are required to describe sounds from all languages in the world, only a small subset are applicable to most of the languages. For example, the velaric feature is based on the degree of use of the velaric air stream mechanism in sound. It is mainly useful for describing sounds in languages spoken in southern parts of Africa such as Zulu, however it is not applicable to most other languages. In selecting articulatory features for our LID system, our objective would be to select the smallest subset of features that could describe a maximum number of sounds occurring in the set of languages under consideration.

3.1.3 Articulatory features in ASR systems

Several studies in using articulatory feature (AF) representations as an alternative to or in combination with acoustic features for ASR have been reported [37, 38, 39]. The

motivation to integrate articulatory features in Automatic Speech Recognition (ASR) systems comes from the following potential advantages of AF-based systems:

- Differences in a feature representation reflect semantic differences, which is not always the case for acoustic representations. For example, an utterance with heavy co-articulation is likely to be close to the original utterance in the feature representation, but not necessarily in the acoustic space [39].

- The articulatory gestures underlying speech are less sensitive to variation across speakers and also additive noise [33].

- Acoustic and articulatory representations are complementary sources of information. This implies that an articulatory feature-based system is likely to have different (and possibly better) performance than an acoustic system. Also, combining the the two approaches may prove to be very advantageous.

- The set of articulatory features is smaller than the phone set, hence a robust estimation of statistical parameters in an AF-based system is possible. This issue will be discussed in detail in the following section.

The studies in [38, 39] employ the binary-valued *distinctive features* described previously. In each case, the feature set is optimally pruned to a smaller subset that is ideal for ASR, using different strategies. On the other hand, the features in [37] are not based on specific phonetic or phonological theories. They are loosely based on different, partially independent dimensions of human speech production. While some features are binary valued, others are multi-variate, analogous to the *traditional features* described previously. Integration of acoustic and articulatory feature systems at different levels has been investigated. While in [38], acoustic and articulatory feature streams are treated separately, in [39], the articulatory features are concatenated to the acoustic features to form a single feature vector. In [34], combining model scores at the feature-level, HMM state-level and word-level was considered. State-level combination proved to be most profitable in

terms of lowering word error rate (WER), followed by word-level and then feature-level combination. Combining AF and acoustic features was shown to significantly improve speech recognition performance in read and spontaneous clean speech [38], as well as fixed vocabulary noisy speech [39]. Furthermore, an AF based recognition system was shown to significantly outperform the acoustic baseline system on a fixed vocabulary task in noisy and reverberent environments [33].

3.2 Motivation for AF based language identification

As discussed above, several studies have shown the advantages of using an articulatory feature representation, in ASR systems. The work in this thesis applies an AF based approach to the problem of language identification. Processing of speech signals in AF based ASR systems essentially consists of two steps:

- Map the acoustic speech signal to parallel streams of articulatory features
- Map the feature sequences into sub-word units defined by the recognition lexicon for generating the final output.

Since the goal of an LID system is simply to assign a language to input speech utterances, AF based LID systems do not require the feature-to-phone mapping step mentioned above.

Articulatory feature-based and phonotactic LID systems follow the same principle of modeling statistical regularities in feature and phone sequences respectively, for identifying languages. Using an AF representation for language identification has several potential advantages.

- Each feature represents a different dimension of articulation of speech and the set of features together encode complementary information about the utterance. The number of symbols in a feature stream can be larger than the number of symbols in other streams as well as the phone sequence. Thus, the feature streams are capable of a more fine-grained temporal encoding of the utterance, allowing the

system to capture sub-phonemic events. The AF representation thus provides a richer representation of the speech utterance. This might be especially significant for short utterances where a feature-based approach would provide a more powerful modeling of the utterance. Since accurate LID with minimal utterance duration is a key requirement in language ID systems, an AF-based approach may prove to be very effective.

- The total number of possible feature values, across all features is very small compared to the number of values in a phone representation. Approximately 30 articulatory-phonetic feature values can fully represent speech from a comprehensive collection of languages in the world. On the other hand, up-to four times as many phone classes are required for the same purpose. Since a feature corresponds to a particular aspect of speech articulation, the number of feature classes within that feature group is typically very small. Having a large phone set may prove to be useful when scoring long utterances, where large context lengths are available; however, for short utterances, there may not be sufficient context available to accurately score the utterance. Some of the advantages of having a smaller number of feature classes include:

  - Fewer feature models translate to fewer number of parameters to be trained. Due to the parallel architecture in feature representation, training data gets shared across features. This makes it possible to train the system parameters more robustly. This is true for estimating parameters of acoustic decoders such as HMMs and ANNs (which may be language independent), as well as the language specific n-gram models.
  - Fewer parameters translate to a smaller system with a lower requirement of memory and other system resources. This is especially significant for LID systems in low power, low memory applications.
  - Due to fewer possible feature values, the set of all possible feature contexts (or feature combinations) is much smaller than that for the phone system. Hence,

when scaling to a new language or a new task, a phone-based system is much more prone to suffer from problems of unseen contexts or even unknown phones. By the very nature of its application, in any LID system, the flexibility of efficiently upgrading to include more languages in the system is always desirable.

- Differences in languages may occur due to variations in articulatory timings within a phone or across phones, such as degree of vowel nasalization. These differences can be efficiently exploited in a feature-based system by modeling statistical dependencies between feature groups without introducing additional models. However, to exploit these differences in a phone-based system, additional models have to be introduced into the system leading to an increase in system parameters. Modeling dependencies across feature streams would also allow for the modeling of sub-phonemic contexts, which could be beneficial for language identification.

3.3 Articulatory features for LID system

The articulatory features that we have considered in our AF based LID system are very similar to those described in [37]. Some of the criteria that were considered for feature selection are:

- Features should be robustly extract-able from the acoustic signal. This is an essential requirement since the articulatory features have to be extracted from an acoustic representation of the speech utterance.

- The AF system should be able to distinctly symbolize a comprehensive majority of the different speech sounds occurring in all the languages under consideration.

- To minimize computational costs and system requirements, the feature set should be as compact as possible.

Some of the features are binary valued while others are multi-variate. All features can also additionally take up silence and "nil" values to represent pauses and non-speech segments

respectively. They are loosely based on the mechanism of human speech production and they reflect the various orthogonal dimensions of speech production. Since they are not directly derived from any linguistic or phonological theory, they may be termed "pseudo-articulatory" features [35].

The seven articulatory features that we have considered for our language identification system are listed below.

- Consonantal place of articulation (cpl) - Describes the location of the constriction in the vocal tract during consonant production. The constriction could occur at the lips (/p/) or teeth (/t/) or further back at the velum (/k/) etc.

- Front-back tongue position (fb) - Based on position of the tongue along the front back axis, with reference to the palato-alveolar region. Differentiates between sounds made in the front (/p, b/) and those made in the back (/g/) of the mouth.

- Lip rounding (rd) - Based on presence or absence of narrowing of the lip orifice gesture. This feature can be used to classify rounded vowels /o, u/ and non-rounded vowels (/e, i/).

- Manner of articulation (mann) - Used to characterize sounds produced by different methods of speech articulation, for example, laterals (/l/), approximants (/h/), fricatives (/f/) etc.

- Nasalization (nas) - Based on the degree of lowering of the soft palate. This feature is binary-valued and sounds can be categorized as nasalized (/n, m/) or non-nasalized (/d, t/).

- Voicing (voi) - This feature differentiates between sounds produced with or without the vibration of the vocal chords. Voiced sounds include vowels (/a, e, i/) and voiced consonants (/b/) whereas (/p/) is an unvoiced consonant.

- Vowel place of articulation (vpl) - Describes the height of the tongue during vowel production. For example, /i/ is a *high* vowel, whereas /o/ is a *low* vowel.

Table 3.1 summarizes the different values possible for each feature group.

Table 3.1: Articulatory features for LID system

Feature	Values
cpl	dental, coronal, labial, retroflex, velar, glottal, silence
fb	front, back, nil, silence
mann	vowel, nasal, lateral, approximants, fricative, silence
nas	nasalized, non-nasalized, silence
rd	+round, -round, nil, silence
voi	voiced, voiceless, silence
vpl	high, low, mid, silence

3.4 AF based LID system: Probabilistic framework

Analogous to a phone-based systems, the front end for a feature-based LID system is made up of acoustic decoders, one per feature group, that map a speech signal to parallel sequences of feature values. Different strategies used for acoustic decoding include Hidden Markov Models or Artificial Neural Networks [21]. Consider a sequence of feature values $\mathcal{F} = f_1, ...f_N$ belonging to language L. The language hypotheses of an LID system are based on a set of conditional probabilities, namely, the probability that the given feature sequence belongs to a particular language, as computed for each language. This probability can be represented as $P(L|\mathcal{F})$. The language L for which this probability is the highest is assigned to the unknown utterance. By applying Bayes' rule, this probability becomes:

$$P(L|\mathcal{F}) = P(\mathcal{F}|L)P(L)/P(\mathcal{F}) \tag{3.1}$$

The term in the denominator is language independent,i.e. it has the same value for all languages, hence it can be ignored. The term $P(L)$ represents the *a priori* probability of language L. If all languages are equally likely to occur, this term would have the same value for all languages and can be ignored. The LID decision is based on the class

conditional probability for feature sequence $\mathcal{F}$ given language L and can be stated as:

$$L^* = argmax_L P(\mathcal{F}|L) \tag{3.2}$$

These class-conditional probabilities are estimated using feature n-gram language models. In general, when assigning a probability value to a sequence of words or feature values, each word or feature would be conditioned on *all* the previously occurring words or feature symbols in the sequence, also known as its history. However, this approach becomes impractical for long utterances. N-gram models assume that language is generated by a time-variant Markov process and model each word, phone or feature symbol on just the $n-1$ symbols preceding it. For example, consider a feature stream $\mathcal{F} = f_1, ...f_N$ belonging to language L. Probability of this feature sequence as given by the n-gram model for L for that stream is computed as:

$$P(\mathcal{F}|L) = \prod_{i=1}^{N} P(f_i|f_{i-1}, ..., f_{i-n+1}, L) \tag{3.3}$$

The probability of an ensemble of K feature groups $\mathcal{F}^1, ..., \mathcal{F}^K$ given a language L can then be computed as:

$$P(\mathcal{F}^1, ..., \mathcal{F}^K|L) = C(P(\mathcal{F}^1|L), ..., P(\mathcal{F}^K|L)) \tag{3.4}$$

where, C is some combination function. Based on maximum likelihood, the language with the highest probability score is assigned to the utterance, i.e.

$$L^* = argmax_L P(\mathcal{F}_1, ..., \mathcal{F}_K|L) \tag{3.5}$$

A simple function for combining scores from different streams is the product of individual stream scores.

$$P(\mathcal{F}^1, ..., \mathcal{F}^K|L) = \prod_{k=1}^{K} P(\mathcal{F}^k|L) \tag{3.6}$$

More complicated score combination and language hypothesis schemes based on higher level classifiers such as Artificial Neural Networks or Gaussian Mixture Models can be envisioned. The various steps in a feature-based LID system are outlined in Figure 3.1.

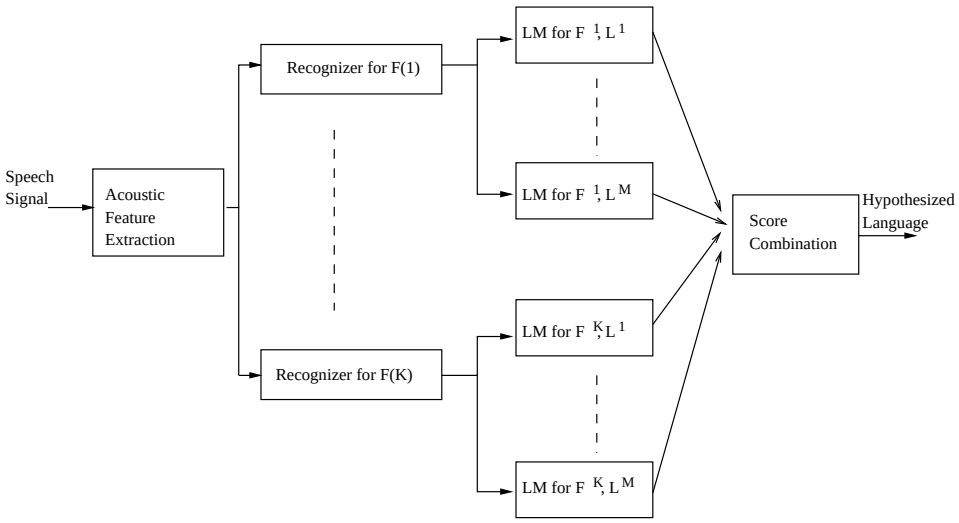

Figure 3.1: Feature-based LID system

3.5 Cross-stream Dependencies

In the previous sections, the feature-based approach to LID has been presented. The idea of modeling statistical dependencies between feature symbols in different groups or streams has been suggested. We will now look at this issue in greater detail.

The product rule stated in 3.6 above assumes that feature groups are independent of each other, given the language. Since sequences for the various feature groups or streams are extracted from the same speech signal, it can be argued that these are not independent of each other. Hence, analogous to modeling dependencies between features at different time positions *within* a feature group, modeling dependencies between features *across* different feature groups might be useful for language identification.

In fact, to completely realize the potential advantages of the parallel architecture of feature streams, modeling statistical dependencies across feature streams is essential. As mentioned previously, modeling such dependencies allows the system to encode differences in articulatory timings such as vowel nasalization or aspirated vs. unaspirated plosive, across languages, which may contain language discriminatory information. While aspi-

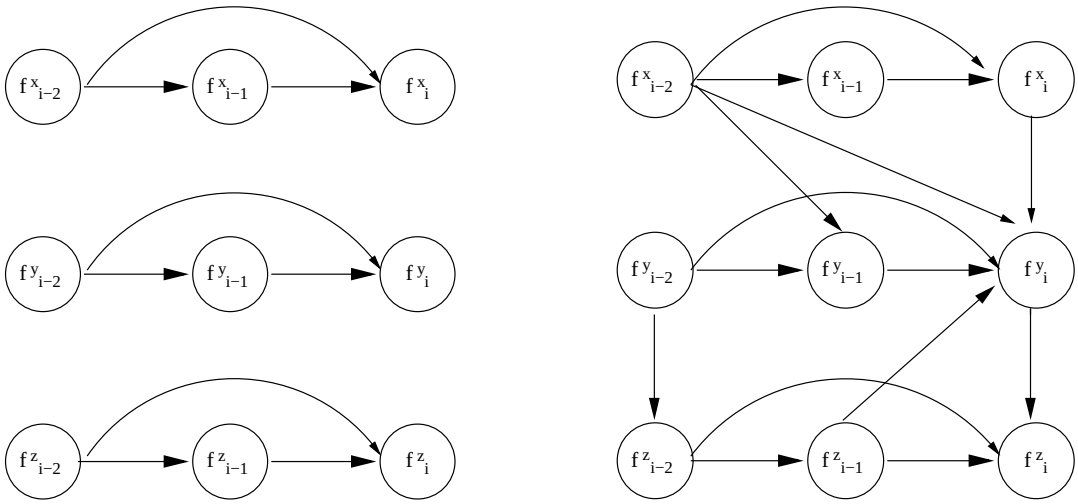

Figure 3.2: Dependency modeling in feature streams. The circles stand for feature variables at different time positions and arrows indicate a dependence relationship

rated plosives are voiceless (/p, t, k/) , unaspirated plosives are voiced (/b, d, g/). These differences can be easily modeled by conditioning the *cpl* feature on the *voi* feature group. Furthermore, many features have a high degree of correlation. For example, for speech articulated in the front portion of the mouth, the *cpl* feature typically has values such as *dental*, *labial* or *coronal* and the *fb* feature always has value *front*. For speech articulated in the back of the mouth, the *cpl* feature has values of the type *alveolar*, *velar* or *uvular* and the *fb* feature always has value *back*. Thus, these two features are highly correlated and conditioning feature values at different time positions between these feature groups may be useful. The concept of modeling within-stream and cross-stream dependencies is illustrated in Figure 3.2. The figure illustrates within-stream trigrams and some cross-stream dependencies in a graphical model framework, which is described in the next section.

3.5.1 Dependency representation using Graphical Models

Graphical models are a powerful tool for depicting, encoding and learning probabilistic relationships among a set of random variables. They are based on the tenets of probability and graph theory. Nodes in the graph structure represent random variables. These nodes are connected with directed or undirected arcs which encode dependence relationships between them. Graphical models are either directed or undirected, based on the presence or absence of directed arrows on the arcs in the graph structure. A directed arc from node A to node B denotes a *dependence* relation between random variables A and B, i.e. B depends upon A. In this case A is the conditioning variable or the *parent* and B is the dependent variable or the *child*. Bayesian networks are a subclass of directed graphical models, also known as directed *acyclic* graphs (DAGs). The condition of acyclicity rules out directed circular paths in the graph structure, which start and end on the same node.

Within-stream and cross-stream dependencies are illustrated in Figure 3.2 in the form of a Bayesian network. Features within each feature group at different time positions correspond to nodes in the graph structure. The directed arcs between nodes represent conditional dependence relationships between the corresponding features. It is essential to avoid circular paths in the graph structure to ensure the validity of resulting probability distributions.

3.5.2 Modeling cross-stream dependencies

As seen previously, the probability of a feature sequence $\mathcal{F} = f_1, f_2, ..., f_N$ given an n-gram language model for language L is computed as:

$$P(\mathcal{F}|L) = \prod_{i=1}^{N} P(f_i|f_{i-1}, ..., f_{i-n+1}, L) \tag{3.7}$$

The probability of a feature f in stream x at position i is represented as $P(f_i^x)$ and can be computed as:

$$P(f_i^x) = P(f_i^x|f_{i-1}^x, ..., f_{i-n+1}^x, L) \tag{3.8}$$

When cross-stream dependencies are integrated into this framework, $P(f_i^x)$ would additionally be dependent on a set of features in streams other than stream x, i.e.

$$P(f_i^x) = P(f_i^x | f_{i-1}^x, f_{i-2}^x, ..., f_{i-n+1}^x, \mathcal{F} \setminus \{x\}, L) \tag{3.9}$$

For each feature stream other than x, the set of features that make the most accurate prediction of $P(f_i^x)$ can be identified and these are represented by $\mathcal{F} \setminus \{x\}$ which is a subset of the set of all features other than those in stream x, i.e.

$$\mathcal{F} \setminus \{x\} \subset (f_{i-n+1}^w, .., f_i^w, f_{i-n+1}^y, .., f_i^y, .., f_{i-n+1}^z, .., f_i^z) \tag{3.10}$$

where w, x, y, z are the feature streams under consideration and n is the context length of interest. Identifying the ideal subset of within-stream and cross-stream dependencies is an NP-hard problem. Consider a system with p feature groups where the context length of interest is n. For any feature at time position i, the total number of possible dependencies or total number of conditioning features (or parents in the graph structure) is $pn - 1$, i.e. all features from other streams, given by $(p-1)n$ and the $(n-1)$ previous features in its own stream. Hence, the total number of different parent sets or combinations of conditioning variables for this feature, is given by:

$$\sum_{i=1}^{(pn-1)} \binom{(pn-1)}{i} \tag{3.11}$$

For a set of 5 feature streams and a context of 3, there are more than 16K possible parent sets of different sizes. Further, such parent sets have to be identified for all feature streams or groups. Figure 3.3 depicts two possible parent sets for feature f_i^y. The search for an ideal subset of within-stream and cross-stream dependencies to be integrated into the LID system needs to traverse the complex search space defined by all these possible combinations across all feature groups and cannot be done exhaustively. This search problem will be discussed in further details in the forthcoming chapters.

Consider a feature value at the current time position i, conditioned on the $n-1$ features within its stream and one feature in a different stream at the same time position. Its

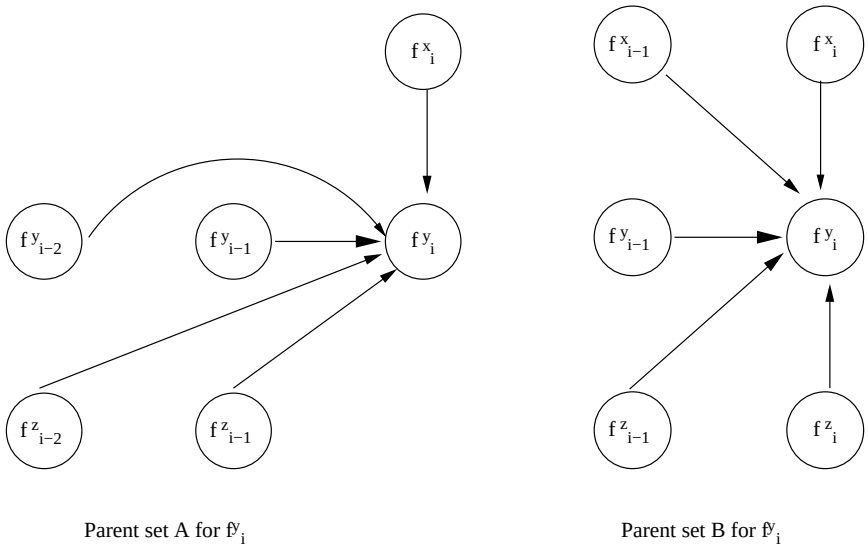

Figure 3.3: Two different sets of dependencies for conditioning f_i^y

probability can be computed as:

$$P(f_i^x) = P(f_i^x | f_{i-1}^x, f_{i-2}^x, ..., f_{i-n+1}^x, f_i^z) \tag{3.12}$$

The additional conditioning variable f_i^z is thus appended to the n-gram context. The number of possible contexts for f_i^x and hence the number of n-gram parameters increases from m_j^n to $m_j^n * m_k$, where m_j and m_k equal number of possible values for features x and z respectively. For robustly estimating probabilities associated with all these different contexts, smoothing procedures used in standard n-gram models are applied, which are discussed in detail in the following sections. Alternatively, by assuming partial conditional independence between different streams, the equation can be approximated by:

$$P(f_i^x) \approx P(f_i^x | f_{i-1}^x, f_{i-2}^x, ..., f_{i-n+1}^x) P(f_i^x | f_i^z) \tag{3.13}$$

With this simplification, the number of n-gram parameters reduces to $m_j^n + m_j * m_k$.

The problem can thus be split-up into two independent problems:

- dependency selection, i.e. finding the ideal subset of dependencies or structure of the Bayesian network of features.

- parameter estimation, i.e estimating parameters for the models represented by the above set of dependencies.

Dependency selection: As discussed previously, the space defined by all possible combinations of dependencies with different number of conditioning variables is huge. Searching this space exhaustively is impractical. On the other hand, conventional local search algorithms such as greedy search would explore only a very small portion of this search space. As a result, they are very unlikely to select the optimal dependency model set. Search techniques of this type tend to be localized and suffer from the problem of converging at a local optimum. Hence we choose to apply the highly robust genetic algorithm to this selection problem. The next chapter describes genetic algorithms (GAs) in general and more particularly, applying a GA to model selection in an AF-based LID system.

Parameter estimation: As shown in equation 3.13, the expression for probability of feature value at a particular stream can be further simplified by assuming partial conditional independence between different streams. At the same time, parents from all the different streams need not necessarily be assumed to be independent. For example, parents from streams x and z are assumed to be conditionally independent but not those from streams y and z. Under these assumptions, the probability expression in 3.14 would be approximated to that in 3.15 below.

$$P(f_i^x) = P(f_i^x | f_{i-1}^x, f_{i-2}^x, f_{i-1}^y, f_i^z) \tag{3.14}$$

$$P(f_i^x) \approx P(f_i^x | f_{i-1}^x, f_{i-2}^x) P(f_i^x | f_{i-1}^y, f_i^z) \tag{3.15}$$

It is difficult to predict which assumptions would be most relevant from the point of view of language identification and could be determined empirically.

Once the form of the probability model has been determined, as discussed above, the next step is to train a set of language models encoded by that expression. As discussed in the previous chapter, n-gram models are a collection of probability distributions that predict the probability of occurrence of a sequence of feature values in that language. An

n-gram model assumes that the probability of occurrence of a symbol in a sequence only depends on the previous $n-1$ symbols. In this section, important issues in estimating n-gram parameters and different techniques used for the same will be discussed.

A major problem in n-gram parameter estimation is presented by data sparsity. It has been found experimentally that a vast majority of valid word sequences occur very rarely, even in very large corpora [44]. For example, in a study of the LOB English language corpus of 1.2 million words, it was found that as many as 60% trigrams and 20% bigrams occur only once in the corpus [43]. Another issue is the problem of unseen sequences. Experiments show that though the possibility of encountering new or previously unseen word sequences decreases with an increase in corpus size, the expected chances of finding new n-grams after the entire corpus has been covered are quite high [44]. For example, on the LOB corpus, the expected chances of coming across new trigrams after examining the entire training corpus was as high as 60% [43]. These issues demonstrate the importance of using appropriate estimation techniques for providing accurate probabilities to rarely occurring sequences and unseen sequences. These issues pose less serious problems for phone-based and AF-based n-gram models mainly due to the much smaller cardinality of the phone and feature systems as compared to the vocabulary of a language. Nevertheless, they need to be tackled appropriately.

3.5.3 Basic parameter estimation

An n-gram model assigns a probability value to a word (or symbol) given its context or history, i.e. $P(w|h)$. This distribution is estimated using all occurrences or samples of the context in the training data, given by S, where S is represented as:

$$S = hw_1, ..., hw_m \tag{3.16}$$

In general, this probability distribution is assumed to belong to a known parametric family of distributions. The parameters associated with the distribution (vector θ) are estimated using S. Many different approaches including maximum likelihood estimation

and Bayesian estimation have been applied to this problem. In the maximum likelihood approach, the value of θ that maximizes the likelihood or the probability of observing the samples associated with that context in the training data is selected.

$$\theta_{ML} = argmax_{\theta} P(S;\theta) = argmax_{\theta} \prod_{i=1}^{m} P(w_i, \theta|h) \tag{3.17}$$

In Bayesian estimation, the parameter is itself considered to be a random variable and an *a priori* distribution is assumed for it. Once the LM parameters have been estimated using one of these techniques, the actual probabilities assigned to the various n-grams need to be modified or *smoothed* to compensate for the effects of data sparsity. These issues are discussed in detail in the next section.

3.6 Smoothing

Data sparsity in the training data gives rise to the two following problematic scenarios.

- A majority of valid word sequences occur very rarely, even in very large corpora. As seen above, the probability assigned to a sequence is computed using the count (i.e number of samples) for that sequence in the training data. This implies that the probability assigned to these sequences by the language model would inappropriately be extremely small.

- The expected chances of finding new n-grams (possibly in the test data) after the entire training data has been covered, is quite high. As the counts for such unseen events would be zero, the language model would assign zero probabilities to these unseen sequences though they are valid sequences.

It is obvious that once the LM parameters have been estimated, the actual probabilities need to be adjusted such that no sequences are assigned zero probabilities and those of rarely seen sequences are bumped up. Since this process of adjustments counters the sharp valleys in the probability distributions caused by these events, its is aptly known as *smoothing*.

The smoothing of probability distributions in a LM is effected by the following process:

- Discounting - a certain amount of probability mass is taken away from the more frequently observed n-grams
- Re-distribution - the discounted probability mass is redistributed among the zero frequency and very low frequency n-grams.

In general, smoothing schemes are also sometimes known as discounting schemes. Two techniques that are commonly employed by smoothing schemes are *back-off* and *interpolation*. For zero frequency sequences, lower order n-grams are recursively used in estimating their probabilities. For example, if the count for a trigram is zero, it is backed-off to its bigrams, and if the bigram also doesn't exist, then to the unigram. In interpolation, higher order and lower order n-gram models are combined together linearly to estimate the n-gram probability, irrespective of the existence of the higher order n-gram. Several different smoothing schemes have been proposed, including:

- additive
- absolute
- Good-Turing
- Jelinek-Mercer
- Witten-Bell
- Kaneser-Ney

In additive smoothing, which is one of the simplest smoothing methods, each n-gram is assumed to occur δ times more than its actual count, with the value of δ ranging between 0

and 1. This ensures that no n-grams have zero probability and actual n-gram probabilities are computed as:

$$P_{abs}(w|h) = (\delta + c(hw))/(\delta * V + \sum_{w} c(hw)) \tag{3.18}$$

Where, $c(hw)$ represents the count for n-gram with history h and word w and V is the vocabulary size.

The Good-Turing estimate is based on the assumption that n-grams that occur with the same frequency should be assigned the same probabilities. If an n-gram occurs r times in the training data then its count is modified to r^* such that:

$$r^* = (r+1)n_{r+1}/n_r \tag{3.19}$$

where, n_r is the number of n-grams that occur exactly r times in the training data. Using this equation, count of zero frequency n-grams is adjusted to: n_1/n_0. As the value of r increases, n_r could become zero for some values of r and the above equation cannot be used. Several methods have been suggested to smooth the values of n_r such that none of them are zero [40]

Many other smoothing techniques, including Jelinek-Mercer and Witten-Bell use a linear interpolation between higher order and lower order n-gram models.

$$P_{wb}(w|h) = \lambda P(w|h) + (1-\lambda)P(w|h') \tag{3.20}$$

where, h' is the history for the (n-1)-gram and the parameter λ is computed specifically for each n-gram. Equation 3.20 can be interpreted as: use the n-gram model with a probability of λ and back-off to the (n-1)gram model with a probability of $1-\lambda$. A detailed discussion of the various smoothing schemes and their relative performance under varying conditions of training set size and for different n-gram lengths can be found in [40].

In our n-gram model training, we tried using different smoothing schemes, including liner, Good-Turing and Witten-Bell. Though choice of smoothing scheme did not significantly affect LID performance, Witten-Bell smoothing was found to give the best accuracy overall, across different experiments.

3.7 LM toolkits

Two popular language modeling toolkits that have been widely used by research communities are the CMU-Cambridge toolkit [42] and the SRILM toolkit [41]. Both these toolkits support n-gram language models. The CMU SLM (statistical language modeling) toolkit, which was first introduced in 1994, has been in use in several academic, government and industrial laboratories. Apart from supporting different discounting strategies such as linear, absolute, Good-Turing and Witten-Bell, it also provides special provisions for handling context cues (language structure tags such as sentence and paragraph boundaries), forced back-offs and linear interpolations between different n-gram models [42]. The SRILM toolkit which was first introduced in 1999 is a more powerful and flexible toolkit. Along with the command set, it also provides a convenient API for extending the capabilities of the toolkit. Besides the standard discounted n-gram models, it supports many other types of models including class-based models, cache models, skip language models, dynamically interpolated LMs etc. [41]. It also supports a wide range of discounting schemes. Both toolkits support the ARPA LM format.

In our experiments, we have used both toolkits for n-gram training and scoring test sequences. We have also experimented with different smoothing techniques. Implementation details of our experiments are provided in the following chapters.

All standard n-gram modeling tools follow the chain rule for estimating the probability of a speech sequence. The probability of a symbol at a given time position is computed based on the previous $n-1$ symbols in the sequence. Hence these tools are not suited for modeling dependencies between parallel sequences of symbols. It is especially difficult to visualize dependencies between feature symbols occurring at the same time position, in different streams using these tools. A novel extension of the SRILM toolkit supporting a parallel representation, known as factored language models (FLM), provides the right framework for modeling cross-stream dependencies. In the FLM toolkit, each word is viewed as a collection of *factors*. These factors relate to linguistic features of the word,

such as its morphological class, root etc. One of the objectives of developing this toolkit was improving the statistical estimation in n-gram models for Arabic, which is a highly inflected language [45]. It can be used to train n-gram models for individual factors. Since the factors associated with a word also occur in parallel, the FLM toolkit is especially suitable for training cross-stream dependency models, where each feature group is a factor.

3.8 N-gram modeling for cross-stream dependencies

N-gram modeling for a feature-based system presents a slightly different challenge as compared to n-gram modeling for words. For example, the vocabulary size is an important parameter in word n-grams. Limiting the vocabulary size results in out of vocabulary words, also known as OOVs. How the n-gram model handles OOVs is an important issue and can significantly impact the performance of the n-gram model. Due to the much smaller cardinality of the feature-based and indeed the phone-based system too, the vocabulary size and hence OOVs are not an issue. However, certain issues specific to feature-based systems need to be resolved. Two such issues are feature synchronization and score combination and are discussed in this section. Both issues arise due to the parallel architecture of the feature-based system.

3.8.1 Synchronization

In a feature-based system, parallel streams of articulatory features are processed simultaneously. The speech signal is mapped to alignments by the feature specific acoustic models. Typically, the alignments for each feature group contain a sequence of feature symbols along with frame (or timing) information about the duration of each symbol and possibly the corresponding acoustic scores. For a cross-stream n-gram model, the alignments or feature sequences for the corresponding streams need to be *synchronized*. A one-to-one correspondence between symbols from the two feature streams does not exist. For example, a feature symbol in a stream with comparatively longer duration may span over more than one feature symbol in the other stream. Also, a symbol from one stream may align with a particular symbol for half its frames and a different symbol for the re-

maining frames. The length of the overlap between two symbols in the different streams may vary from a few frames to the entire length of the symbol. Synchronization problems arising due to this can be seen from figure 3.4. The dashed vertical lines show the synchronization mis-match between the different streams. These issues become further

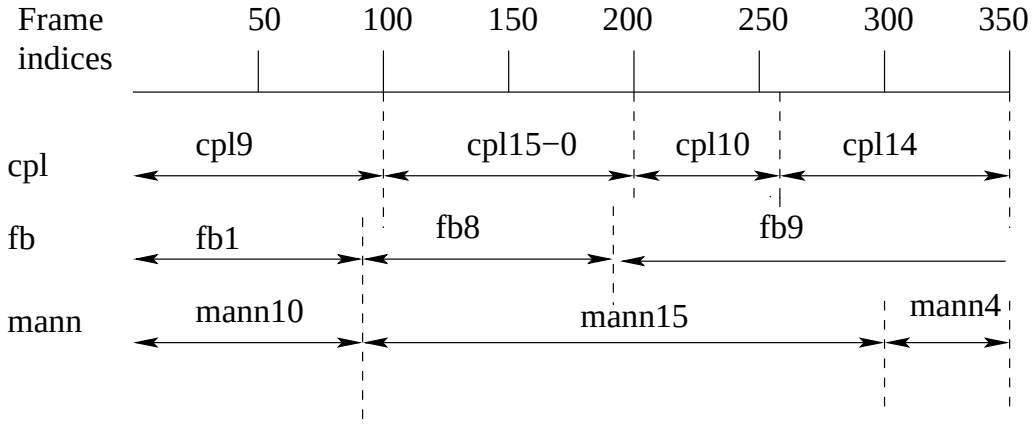

Figure 3.4: Synchronization of streams *cpl*, *fb*, and *mann* with reference to frame indices

aggravated with increase in number of feature streams. The synchronization scheme will have an impact on the cross-stream n-gram models and n-gram scores for test utterances. In general, alignments from the acoustic decoders tend to contain some duplications of symbols i.e. sometimes, the same feature is recognized repeatedly before the recognition process moves on and selects another model, resulting in the duplicate symbols. These repetitions further complicate the synchronization problem.

A simple solution to these synchronization issues is to align the symbols for all feature streams under consideration that overlap for one or more frames. However, we observed that this frame level synchronization lead to a lot of duplications in the aligned feature sets resulting in degraded language ID performance. The duplications cause an imbalance in the counts for the respective feature n-grams and that can impact both the n-gram parameter estimation and smoothing. They cause a similar imbalance during n-gram scoring too, where the probabilities for the repeated n-grams get emphasized. For these reasons,

we decided to discard all sequential duplications in the aligned feature sequences. This led to a significant improvement in the language ID accuracy. In fact, we observed similar tendencies in the within-stream n-grams too, where discarding all symbol duplications lead to more reliable language models and a marked improvement in language ID accuracy. Other synchronization schemes based on different values of minimum number of overlapping frames are possible and the final choice for the threshold can be made empirically. The feature symbols align differently for different feature combinations and are a function of the frame information for the symbols in each group. For example, alignments for *cpl* and *fb* will synchronize differently as compared to *cpl* and *mann*, and when the duplicate feature sets are discarded, the resulting aligned feature sets will compress differently and will have different sizes (i.e. different number of feature symbols) in each case.

3.8.2 Score combination

As described in the previous sections, the model set of a feature-based LID system consists of within-stream and cross-stream dependency models. Each model assigns a score to the test utterance. Several strategies for combining these scores can be envisioned. A simple score combination technique is to consider the product of the likelihoods for each model as described in equation 3.6. This rule assumes that the feature streams and the corresponding language models are independent of each other. Each feature group has its own recognizer and alignments for a test utterance from each recognizer has different number of feature symbols. The scores for different feature groups thus have different ranges. To make the scores for different models comparable, they need to be normalized by dividing by the number of symbols in the corresponding utterance sequences, before score combination. Phone based LID systems using language dependent phone recognizers also possess a parallel architecture and face a similar score combination issue as discussed in section 2.2.6 in chapter 2. Techniques of combining scores from different models using higher-level classifiers have been reported [28, 12]. In these studies, likelihood scores from the different language models are considered to be elements of a feature vector. This vector is input to a higher-level classifier such as a GMM or ANN based classifier, and the

final language hypothesis is made based on the classifier output.

3.9 Summary

In this chapter we have considered the theoretical framework behind a feature-based LID system. Articulatory features in ASR systems and the motivation behind using AF-based system for LID were considered. The importance of modeling cross-stream dependencies and related issues were also discussed. In the next chapter, the search problem of model set selection in an AF based LID system will be discussed, along with the motivation for using a genetic algorithm for this search problem. A detailed discussion of GAs will also be provided.

Chapter 4

THE GENETIC ALGORITHM

4.1 Introduction

We come across many different types of search problems in almost all fields of science and technology. As a result, there exist a wide range of search algorithms and strategies. The appropriate search algorithm for a problem can be selected based on problem domain and specifications. These factors also determine problem complexity; for example, consider the following search problems in different domains and having varying complexities:

- Search for the smallest integer in an array of integers.
- Given a map, search for the shortest route from city A to city B.
- Search for documents or files on the Internet containing text in a specific language.
- Search for the best set of features for classifying flowers in a bunch containing n different species.

It can be seen that for each problem, a search algorithm known to work for problems in that domain and suitable for the specific problem at hand should be considered. For example, for the first search problem, any sorting algorithm such as selection sort, insertion sort or quick sort, can be used. Depending on size of the array, required speed and availability of memory etc., the appropriate sorting algorithm can be selected. Search problems that involve selecting one or more elements from a given set to be used for a specific purpose, can be identified as a separate sub-class, as *selection problems*. For example, the problem mentioned above, of selecting the best sub-set of features for flower classification, can be

constructed as a selection problem. Selection problems are also known as combinatorial optimization problems, where every valid subset of features represents a solution and the problem lies in identifying the best solution.

In general, a selection problem has two parts, namely, selecting a subset of elements/features and evaluating them for classifying the available data. For such problems, the feature selection can be made either independent of the final classification, known as the *filter approach*, or in conjunction with it, known as the *wrapper approach*. The filter approach filters out all features from the set that presumably would not contribute positively to the classification process, before the actual classification is done. On the other hand, in the wrapper approach, the the feature search process is wrapped around the classification task Though the filter approach tends to be computationally less expensive, the optimal feature set tends to be sensitive to the classification algorithm. This dependence is accounted for in a wrapper approach. For this reason, we plan to use the wrapper approach for our selection problem.

Search algorithms can be classified as either *deterministic* or *heuristic*. Deterministic search strategies are usually exhaustive, covering the entire space of possible solutions and are guaranteed to find the best solution. An example of a deterministic search algorithm is the breadth-first search. However, for many problems, the search space is huge and an exhaustive search is not possible. For such problems, many algorithms based on heuristic methods have been developed i.e. algorithms guided by *speculative formulations*. These algorithms do not cover the entire search space, but only cover the *promising* areas within this space. These algorithms employ an *evaluation function* to determine the merits (also referred to as profitability or fitness) of possible solutions used to direct the search. Examples of heuristic selection strategies include:

- hill climbing

- greedy search

  - forward selection
  - backward elimination

- simulated annealing
- genetic algorithm

Some of these algorithms were first developed for applications in the field of artificial intelligence (AI) and are convenient to consider from the perspective of AI problem formulation where the objective is to move from the *current state* to the *goal state* using a range of possible operations or moves[54]. For example, a hill climbing algorithm is based on the principle that a move is accepted only if the resulting state is closer to the goal than the current state. Closeness to the goal state is determined using the evaluation function. This implies that in a multi-peaked search space, the algorithm will move towards a local optimum, neglecting the global optimum. This is a major drawback with the hill-climbing algorithm. A detailed description of the hill-climbing algorithm can be found in [54].

Simulated annealing is an improvement over the hill climbing algorithm that tries to overcome this problem by tempering the hill climbing procedure with random re-starts. The probability of jumping to a new starting point is initially very high, but reduces exponentially with time, thus allowing the search to break free from local optima.

The greedy search algorithm is based on the principle of selecting the most profitable move at each point. In forward selection, starting with an empty set, elements are successively added at each step based on the criterion of maximum profitability. The algorithm stops when adding any of the remaining elements to the set does not result in further improvements in fitness. Conversely, the backward elimination algorithm starts with the set of all elements and at each step discards one element from the set. The algorithm stops when discarding any further elements does not result in an improvement in performance on the evaluation function. These techniques have been applied to several feature selection problems and are commonly known as the *branch and bound search techniques.* Like hill

climbing, the greedy search algorithm is another example of a local search technique and tends to converge on some local optimum [54].

The genetic algorithms (GA) belongs to the class of *evolutionary algorithms* that have been successfully applied to several search and optimization problems. It applies strategies that mimic events in evolutionary sciences such as reproduction, mutation etc. One of the major drawbacks of heuristic searches, as discussed above, is lack of robustness i.e they tend to converge on locally optimal solutions. GAs on the other hand, tend to be highly robust and hence find application in many varied fields of science and technology. The unique characteristics of a GA that make it universally applicable, simple to apply and highly robust will be discussed in details in the following sections.

4.2 Genetic Algorithms

Genetic algorithms are based on evolutionary processes such as natural genetics and selection. They follow the principles of survival of the fittest and information sharing through reproduction. John Holland and his group at the University of Michigan, were amongst the key researchers in this field. They studied adaptive processes in natural systems and tried to apply them to artificial systems. Biological systems tend to be highly robust, efficient and flexible by incorporating techniques of self-repair, self-guidance, reproduction etc [46]. These are strategies that even the most adaptive artificial systems cannot support. Genetic algorithms mainly derive their robustness from simulating these mechanisms in an artificial environment. Some of the unique features of genetic algorithms are:

- GAs do not deal directly with problem parameters. They work with an encoded *string* that contains an encoded value for each parameter.

- GAs always work with a *population* of potential solutions (in the form of encoded strings) rather than a single solution.

- GAs employ a user defined *evaluation function* (also known as a *fitness function*) to

determine the merit of each potential solution. The fitness function is a measure of profit or utility that the user wishes to maximize.

- GAs use probabilistic transition rules in all their operators. Operators form the back-bone of a GA; they process successive generations of populations, exploring different areas of the search space and guiding the search towards the global optimum. The basic GA operators are: *reproduction* (also known as *selection*), *crossover* and *mutation*.

We will now discuss these special properties of genetic algorithms along with some GA operators in greater detail.

4.2.1 Special GA attributes

The special attributes of GAs mentioned above make them highly flexible, robust and universally applicable. Since the GAs work with encoded versions of the problem parameters, they are shielded from problem-specific parameter properties that might be difficult to account for in a conventional search. In general, conventional search techniques work directly with the problem parameters. This property of the GA makes it very attractive for applying to problems in domains where very little domain knowledge is available. Powerful GA operators can be applied to encoded strings of parameters irrespective of the problem which the parameters represent, with equal ease. The GA allows the user to define the fitness function that is used to evaluate different solutions being considered. The fitness function helps guide the search in the appropriate direction and ensures that the final solution is optimal for the specific problem the user has in mind. Another advantage of employing a fitness function is that it allows for multi-criterion optimization. For example, in selecting an ideal subset of features for classification, along with accuracy, the cost associated with a set of features may also be an important consideration and can be included in the fitness function in the form of a penalty term. Also, since GAs work on encoded parameters and a user defined fitness function, they do not directly use any problem-specific knowledge in their search. All these features contribute to the universal

applicability of genetic algorithms. They also make them especially applicable to search problems in domains where domain knowledge is difficult to obtain or unavailable. The GA works with a population of potential solutions rather than a single solution. This can be visualized as the algorithm traversing different sections of the search space simultaneously. All the solutions under consideration are evaluated and compared. These features make the GA highly robust and unlikely to get caught in local optima. Finally, all GA operators use random choice as a tool to guide the search in regions of the search space with likely improvements, thus reaching out for the global maximum and further contributing to the robustness of the GA. The algorithm continues to process generations of solutions until a pre-defined termination criterion is achieved.

Since GAs are inspired by natural genetics, many terms from the field of natural science have become a part of GA parlance. Some such commonly occurring terms are:

- *chromosome* - an individual member of the population
- *genes* - individual elements of a chromosome are know as genes. They have two attributes
  - *alleles* - different values that a gene can take up. For example, a binary encoding has two alleles, 0 and 1.
  - *locus* - position of a gene within a chromosome is know as its locus.

4.3 GA operators

Some of the GA operators have been mentioned briefly in the previous sections and will now be discussed in greater detail. Almost all GA operators draw inspiration from natural evolutionary processes. They manipulate individual solutions in a population over several generations, improving their fitness in each successive generation. The most basic operators employed by GAs are:

1. Selection

2. Crossover

3. Mutation

4.3.1 Selection

The selection or reproduction operator is based on the principle of natural selection. Chromosome are selected for reproduction based on their evaluation by the fitness function. The mating pool may contain multiple copies of a highly fit chromosome, while a weak chromosome may have no representation at all. Size of the mating pool is usually equal to the population size. It is extremely important for the selection operator to maintain a healthy balance between choosing fit individuals for mating and maintaining genetic diversity within the mating pool. For example, the population may contain some super-individuals (highly fit individuals that represent a local optimum) and the selection operator needs to guard against these individuals causing pre-mature convergence.

Amongst the first schemes to be used for selection was the *roulette wheel selection*. This can be thought of as every member of the population being given a slot on the roulette wheel whose size is proportional to the individual's fitness. The probability of and individual member being selected by spinning the wheel is thus proportional to its fitness. The wheel is spun and the chromosome from the selected slot enters the mating pool. This procedure is repeated till the desired size of the mating pool is reached. The roulette wheel has since been shown to suffer from stochastic errors due to the random nature of each spin and several other selection schemes have been proposed [46]. These include deterministic sampling, expected value selection, stochastic universal sampling etc. In deterministic sampling, the probability of selection for each individual is calculated by dividing its fitness by the average fitness. The expected number of copies of each string in the mating pool is the product of its probability of selection and the population size. Each member is copied into the mating pool with samples equaling the integer part of its expected number mentioned above. The population is then sorted according to the fractional parts of the expected numbers. The remaining members of the mating pool are

drawn from the top of this sorted list. The first step of stochastic universal sampling is identical to deterministic sampling, however, it differs in how the remaining portion of the mating pool is selected. Another interesting selection scheme is *tournament selection*, also known as Wetzel ranking. In tournament selection, a group of individuals is drawn from the population using the roulette wheel and the string with the highest fitness within this group is selected for mating. The size of this group is an important parameter and can be determined empirically. Though the inferiority of the roulette wheel selection compared to other selection schemes has been established, very little difference has been found in the performance of the various other schemes mentioned above [46]

4.3.2 Crossover

Once the mating pool has been selected, the next step is to apply the crossover operator to randomly paired individuals from within the mating pool. In simple single point crossover, two offsprings are produced by swapping characters or genes between the two mating strings, starting from a randomly selected crossover site. Patterns and templates of substrings within a string are known as *schemata* and highly fit, short length schemata are known as building blocks. A short defining length is important for the survival of a schema because the crossover operator is likely to disrupt longer schemata during swapping of bits across mating partners but not those with shorter defining lengths. Reproduction and crossover look for similarities in patterns observed across the population with emphasis on similarities in high performance strings and process different permutations and combinations of these schemata. Building blocks thus identified are propagated through successive generations of the population with exponentially increasing representation. A GA approximately processes N^3 schemata per generation where N is the population size [46]. This highly efficient and effective processing of schemata is called *implicit parallelism*. Apart from the 1-point crossover scheme, other crossover schemes that have been commonly implemented are multiple point crossover and uniform crossover. Multiple point crossover is similar to 1-point crossover except that crossover takes place at more than one crossover sites. In uniform crossover, every locus is a potential swapping site. A crossover

mask is set up and swapping occurs at all sites indicated by the mask. In 1-point and multiple point crossover, the crossover site is randomly selected and in uniform crossover the crossover mask is generated randomly. 1-point crossover is in general known to be inferior to multiple point and uniform crossover. Though multiple point crossover is more disruptive in general, it also has a capacity of combining more schemata than 1-point crossover. Uniform crossover may also cause considerable disruption but is known to be most effective in finding building blocks. In general, as the number of crossover sites in multiple-point increases, it tends to behave like the uniform crossover. The crossover operator is applied to the mating partners with a certain probability, known as crossover probability thus allowing for the possibility of duplicating the parents into the next generation. In general, a crossover probability ranging between 0.8 and 1.0 is used.

4.3.3 Mutation

The mutation operator is required for maintaining the heterogeneity of the population and preventing premature loss of potentially useful solution prototypes, though it is considered to be of secondary importance compared to selection and crossover. During mutation, each gene of every chromosome in the population is considered and altered with a probability equal to the mutation probability. For example, for a binary encoding, altering a gene simply means changing its allele from 0 to 1 or vice versa. In general, a probability of mutation of around 0.001 or smaller is used.

4.3.4 Advanced GA operators

Apart from the basic GA operators discussed above, many other operators and techniques have been developed over the years with the objective of further improving GA performance in one form or another. Two such operators are described below:

Elitism: Elitism is a simple scheme in which special care is taken to preserve the fittest chromosome seen by the GA so far. For example, before processing a generation, the fittest individual in the generation is noted. If the new generation formed after applying the GA

operators does not contain the stored fittest individual from the previous generation, then it is added to the new population. Elitism is shown to significantly improve GA performance in unimodal optimization problems but may not be effective for multi-modal functions [46].

Crowding: A crowding factor can be applied to an overlapping generation model and can prove to be highly effective in multi-modal optimization. In an overlapping generation model, during a generation, the population is only partially regenerated. When a new individual is introduced into the population, one is selected to be removed from it. In simple terms, when applying crowding theory, the individual that is most similar to the newly created individual is selected for elimination.

In the sections above, we have discussed some GA operators as well as the special attributes of GAs that contribute to their robustness. We will now look at the actual application of a GA to a simple optimization problem.

4.4 GA for a simple search problem

Consider the problem of finding the highest value of the function $f(x) = x^2$ for $0 < x < 64$. Though the answer to this problem is more than apparent, the problem will be useful in demonstrating the steps involved in implementing a GA and its functioning. The only parameter that needs to be considered in this problem is x. Since GAs do not work with problem parameters directly, but with some form of encoding, let us consider the binary representation of the decimal value of x as the encoded parameter. In general, parameter encoding in the form of a binary string is very popular in GAs and will be discussed further in the following sections. The GA requires a fitness function that is related to the final goal of the problem.Let us consider the function $g(x) = x$ for the fitness function for this problem. Two important parameters for a GA are the size of the population and crossover probability. For this problem, a population size of 4 and crossover probability of 1 is assumed. In the first step, 4 values for x between 0 and 64 are generated randomly.

The binary representation of these 4 values constitutes the initial population as shown in figure 4.1. Each member of the current generation is evaluated using the fitness function.

A = 001100

B = 011010

C = 100101

D = 001010

Figure 4.1: Initial population

The respective fitness values assigned to each member of the initial population would be:

$$A = 12, B = 26, C = 35, D = 10$$

The average fitness value for this generation is 20.75 and the fittest individual has a fitness value of 35. The next step is to select pairs from the within the population for reproduction using the selection operator. The probability of selecting a member for reproduction is a function of its fitness value and the pairing together of selected individuals for mating is purely random. Using the selection rule, four members (same number as the population size) are selected for reproduction and randomly paired off as (B, C) and (C, D). Note that due to its high fitness, C gets selected twice while the relatively less fit A does not get any representation in the mating pool. The next step is to apply crossover operator to each pair. Figure 4.2 illustrates the 1-point crossover operation. the crossover sites for the parent sets (B, C) and (C, D) are 2 and 5 respectively. $'c'$ indicates the crossover point and E, F, G, H are the corresponding offsprings.

The new generation consisting of E, F, G, H is thus formed. Fitness values of members of the new population are:

$$E = 25, F = 38, G = 5, H = 42$$

The average fitness of for this generation is 27.50 and the fittest individual has fitness value of 42. It is seen that with a single iteration of the GA, the average fitness of the population has increased and so has that of the fittest member. With successive iterations, the search

will move towards better and better solutions and converge on a string with all 1s. It is interesting to note that if the fitness function used were $g(x) = x^2$, then, the selection function might have chosen appropriate parents with higher probability and improvement in fitness between generations could have been greater. In effect, the algorithm could have converged earlier. This illustrates the importance of selecting an appropriate fitness function vis-a-vis the function being optimized.

4.4.1 GA: Important functions and parameters

Some of the important functions and parameters in a GA run that impact its performance and efficiency are listed below.

- *Population size*: This is an important factor that has an impact on the efficiency and resource requirements of the GA. The optimal population size is a function of the search problem at hand and can be determined empirically. If the population size is too small, it may not contain sufficient genetic diversity to arrive at the globally optimum solution. On the other hand, if the population is too large, it results in an overhead of unnecessary storage and computations.

- *Maximum number of generations*: The genetic algorithm, on its own, cannot recognize the optimal solution and hence does not know when to stop. A user defined termination criterion is used to determine the stopping point. However, in several optimization problems, this criterion is determined heuristically and may over-estimate the optimal solution. For this reason, it is essential to set the maximum number of

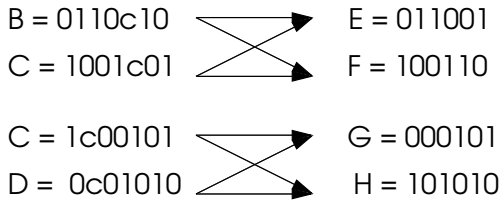

Figure 4.2: Reproduction and crossover operations

generations processed by the GA to a realistic number. In general, setting the maximum number of generations to be processed to be too small hampers the chances of the GA of finding the optimal solution.

- *Crossover and Mutation probabilities*: The crossover and mutation operators have been discussed previously. The probabilities with which these operators are applied within the population can significantly impact their effect and effectiveness.

- *Random number generation functions:* All GA operators encode probabilistic rules to together carry out a structured but random search of the complex search space. The initial population of solutions is also randomly generated. Due to these factors, good random number generation functions play a key role in making a GA effective and efficient.

It has been suggested that one of the major drawbacks of a genetic algorithm is its strong dependence on these parameters, which have to be tuned experimentally for the problem at hand. However, for certain search problems, such as selecting the ideal subset of features for classification, once the optimal subset has been determined, the GA does not need to be rerun. Hence, GAs may still prove to be the most effective solution.

4.5 Previous applications of GAs

Due to the robustness of genetic algorithms, we chose to apply a GA to our selection problem. In this section we will discuss GA applications to similar feature selection tasks as well as to other problems in the same domain as our dependency selection problem. Over the past few decades, genetic algorithms have found application across highly varying fields including biology and bio-chemistry [55], different areas of medicine such as diagnosis [58] and medical imaging [59], a wide spectrum of engineering problems including control systems [57] and VLSI circuit design [56] as well as several simulation problems in the fields of social sciences and economics [60].

Moreover, genetic algorithms have been successfully applied to many problem in the speech and language processing domain [47, 48, 50]. Also, they have been used for feature selection in classification problems in different domains [53, 62, 61] and more specifically, they have been employed in similar model selection problems [51, 52]. These issues are discussed in greater detail below.

4.5.1 GA in speech and language processing

Over the past decade, genetic algorithms have been applied to several problems in speech and language research.

In [50] a GA is used in part of speech (POS) tagging of Chinese sentences. This technique is compared to other techniques of POS tagging including rule-based methods, Hidden Markov Models and recurrent neural networks. The GA technique is found to be the most flexible in terms of integrating different sources of information namely, statistical and rule-based and is found to have the best performance. One of the major drawbacks of the GA system was found to be its computational complexity.

Finite state automata (FSAs) have been commonly used for encoding phonotactic constraints in numerous natural language processing (NLP) tasks, including computational phonology. Normally, these FSAs are constructed manually, which makes it highly expensive and time-consuming. In [48], a GA based approach for learning FSAs from positive data is presented. Results with a toy data-set and a simple set of Russian nouns are discussed. The results indicate that a genetic search can be used successfully for automatic discovery of FSAs for encoding phonotactic constraints for tasks in computational phonology.

Classical parsing techniques conduct an exhaustive search over all possible interpretations of a sentence. However, as the size of the sentence increases, this search space grows exponentially and an exhaustive search may not be possible. In [49], a probabilistic natural language parser based on GAs is described. Results discussed in this work indicate that

GAs provide a robust method for parsing positive examples of natural language. A further improvement in performance is possible by using probabilistic context-free grammars in place of deterministic grammars.

A GA based technique for pronunciation modeling is discussed in [47]. Genetic algorithms are suitable for learning context-sensitive rules for many languages. This work is based on the premise that these rules are useful for pronunciation generation. Different contextual factors are used to predict letter-to-sound mappings, and identifying the ideal subset of contextual factors to consider is an optimization problem. Pronunciation rules are generally expressed in the form of such mappings. In this paper, a GA implementation for generating pronunciation rules is discussed. The rules thus derived are evaluated for the correct probability of letter-to-sound mapping and their coverage of the training set. Experimental results show that it is possible to get a high probability of correct letter-to-sound mapping and a good coverage of the training data using rules derived by the GA. Performance over different values for GA parameters such as population size, number of generations etc. are reported. Though the probability of correct letter-to-sound mapping is relatively insensitive to these parameters, rule coverage is a function of the population size and improves with increasing population size.

4.5.2 Genetic algorithms in model selection

In this section, a couple of example of GA applications to model selection problems in different domains are discussed.

An instance of using genetic algorithm as a search tool for model selection can be found in [51]. A GA is used for model selection for undirected graphical models. A parameter encoding of an undirected graphical model is obtained by representing every existing edge in the model by a 1 and every missing edge by a zero. A novel crossover operator that exploits the graphical representation of the model is proposed. All graph structures within a given population are evaluated with respect to a penalized likelihood

criterion, which combines the likelihood of the data given the model with a term penalizing the complexity of the model. The GA technique is compared to a step-wise backward elimination procedure based on greedy search, where one edge at a time is deleted from the graph structure. Both techniques were tested by an artificial simulation procedure on a graph with 10 vertices, yielding 2^{45} possible models. The results of this simulation study show that GA search generated higher values for the fitness function and selected more parsimonious models in the majority of cases [51].

As mentioned in section 3.5.1 of chapter 3, Bayesian networks are in wide use as a technique for performing probabilistic inference, mainly for learning dependence relations between random variables. A GA-based search for identifying the best Bayesian network structure for a given database of cases is described in [52]. Empirical results for simulations of the ASIA and ALARM networks are discussed. While the ASIA network is based on a small piece of artificial qualitative medical knowledge, the ALARM network was developed for simulation of potential anesthesia problems in the operating room [52]. Initially, an ordering between the network nodes is assumed to maintain the validity of the randomly generated Bayesian structures. In the following GA implementation, the assumed ordering is discarded and a *repair operator* is applied to the newly generated structures as and when required, to convert them into legal Bayesian networks. A performance analysis of GA parameters for simulations of the ASIA and ALARM networks are reported. The results show that a large population size, a relatively low mutation rate and elitist reduction are important for GA based model selection of Bayesian networks.

4.6 GAs for model-set selection in feature-based LID

In this section we will discuss the specific issues of applying a genetic algorithm to our problem of selecting dependencies in a feature-based LID system. The final objective is to build a feature-based language identification system that incorporates the ideal subset of within-stream and cross-stream dependency models.

4.6.1 Why GAs?

As discussed in the chapter 3, along with within stream dependencies, modeling dependencies between features in different streams is effective in exploiting the parallel architecture in a feature-based system and contributes to improving language identification accuracy of the system. Visualization of the causal relationship enforced by a dependency between features at different time positions within a stream as well as across streams is easily possible using a Bayesian network. As discussed in chapter 3, a feature system with 5 groups and a context length of 3 has 16K possible dependency subsets. A dependency subset is made up of such causal relationships associated with each group. Each subset can potentially form the model set of the feature-based LID system and has an LID accuracy associated with it. It is obvious that an exhaustive search of over all such subsets is not feasible and a heuristic search is required. Comparing LID accuracies of individual models and different combinations, of model sets indicates that complex and nonlinear interactions occur between feature groups. The ideal dependency set would be a complex function of these interactions, based on the data corpus, the languages under consideration, the amount of data available etc. Due to these factors, a knowledge-based approach to dependency selection would be impractical. At the same time, due to the multi-peaked nature of the search space, a heuristic algorithm based on local search is also unlikely to be adequate. For these reasons, we chose to implement a GA for selecting the dependency model set for our feature-based LID system.

In summary, we choose to apply a genetic algorithm for model-set selection because:

- the search space defined by all possible combinations of dependencies is huge and can not be searched exhaustively;

- complex, non-linear interactions between feature streams make it impossible to predict which combination of dependency models would be good for language ID; and

- due to the multi-peaked nature of the search space, heuristic algorithms based on

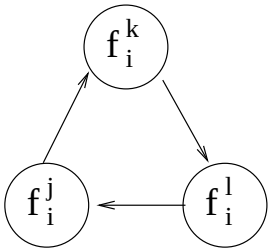

Figure 4.3: Circular dependencies

local searches are likely to converge on some local optimum, but a genetic search would be more robust, in this case.

4.6.2 GA implementation issues

A simple implementation of a GA was discussed in the previous section. Before applying a GA to dependency selection, the search problem need to be clearly defined and various issues specific to the nature of this problem need to be addressed and will be discussed in this section.

4.6.3 Defining the search

Consider a list containing all possible within-stream and cross-stream dependencies. Every subset of this list represents a potential model set for a feature-based LID system and the selection problem is defined as identifying the ideal model set from amongst all these possibilities. However, every subset of the list does not necessarily represent a valid model set. All subsets that contain circular dependencies can be eliminated. Figure 4.3 illustrates a Bayesian network with a circular dependency where f_i^k is conditioned on f_i^j which in turn is conditioned on f_i^l which in turn depends on f_i^k.

It is essential to avoid circular dependencies in a model set to ensure that it represents a valid probability distribution. One scheme of avoiding circular dependencies would be to restrict the parent set of successive feature groups, such that circular dependencies are not possible. For example, the feature groups are organized in an ordered list such

as: $(f^{cpl}, f^{fb}, f^{mann}, f^{rd}, f^{vpl})$. Assuming a context length of interest of 3, this can be enlarged to an ordered list of feature variables as:

$$(f_{i-2}^{cpl}, f_{i-1}^{cpl}, f_i^{cpl}, f_{i-2}^{fb}, f_{i-1}^{fb}, f_i^{fb} f_{i-2}^{mann} f_{i-1}^{mann} f_i^{mann} f_{i-2}^{rd} f_{i-1}^{rd} f_i^{rd} f_{i-2}^{vpl} f_{i-1}^{vpl} f_i^{vpl})$$

A variable at any position in this ordered list, can have as it parents, any subset of feature variables preceding it, but no variable that comes after it in the list. This will ensure that the selected model set does not contain any circular dependencies. For example, parents for the f_i^{mann} can be drawn only from the list:

$$(f_{i-2}^{cpl} f_{i-1}^{cpl} f_i^{cpl} f_{i-2}^{fb} f_{i-1}^{fb} f_i^{fb} f_{i-2}^{mann} f_{i-1}^{mann})$$

Feature variable f_i^{cpl} can be conditioned only on variables at previous time positions within the *cpl* group, whereas, f_i^{vpl} can be conditioned on any feature variable in the list (excluding itself, of course). The drawback of this scheme is that by not allowing certain dependencies, certain sections of the search space have been eliminated from the search and hence, the model set identified using this scheme may not be optimal. Another scheme that would circumvent this issue would be to allow all possible model sets and then filter out all subsets that encode circular dependencies. However, the ordering scheme has a much simpler implementation as compared to the filtering scheme. Furthermore, the disadvantage mentioned above can be countered to some extent by repeating the search with different feature group orderings. For these reasons, we choose to apply the ordering approach for avoiding circular dependencies. The selection problem can now be rephrased as follows: given an ordered set of feature variables, identify the ideal subset of models for the given language identification task using a feature-based system.

4.6.4 GA implementation

In this section, we will discuss GA implementation issues related to the model set selection problem presented above.

Parameter encoding Once the problem has been defined, the first step in applying a GA search to the problem is parameter encoding. For the model set selection problem, each dependency can be considered as a parameter. Hence, every parameter is binary

valued, defining whether the dependency is included in or excluded from the model set. Thus a binary encoding of the parameters seems to be an obvious choice. This choice is also in keeping with one of the fundamental principles of GA coding (Goldberg, 1989, p.80):

"The user should select the smallest alphabet that permits a natural expression of the problem."

This principle suggests that a binary encoding, if applicable, would be the best option. The maximum number of schemata per bit of information of a coding is an inverse function of the cardinality of the coding scheme. The essence of a GA search is based on processing schemata, hence, the greater the number of schemata available for processing, the better. Hence, a binary encoding continues to be a very popular choice for parameter encoding.

Consider the ordered list of feature variables with context length of two:

$(f_{i-1}^{cpl}, f_i^{cpl}, f_{i-1}^{fb}, f_i^{fb}, f_{i-1}^{mann}, f_i^{mann}, f_{i-1}^{rd}, f_i^{rd}, f_{i-1}^{vpl}, f_i^{vpl})$

Cross-stream bigram dependencies for this ordered list are:

$(cpl \rightarrow fb, cpl \rightarrow mann, cpl \rightarrow rd, cpl \rightarrow vpl, fb \rightarrow mann, fb \rightarrow rd, fb \rightarrow vpl, mann \rightarrow rd, mann \rightarrow vpl, rd \rightarrow vpl)$

The preceding variable is the conditioning variable and the following variable is the dependent variable. Two types of cross stream bigrams are possible for this variable set, namely, dependency with conditioning variable from another stream at the previous time position ($f_{i-1}^{cpl} \rightarrow f_i^{fb}$) and dependency with conditioning variable from another stream at the same time position ($f_i^{cpl} \rightarrow f_i^{fb}$). In the model set list above, the former type of bigrams are not considered for simplicity. This results in 10 dependency models and hence, 10 parameters. A binary encoding for these parameters would have a string length of 10. Each unique binary string with 10 bits would result in a unique subset of the list above and would represent an LID system with the set of models included in the string. For example, consider any random 10 bit string:

1011010001

This string represents a set of the 1st, 3rd, 4th, 6th, and 10th models from the list, i.e.

$$(cpl \rightarrow fb, cpl \rightarrow rd, cpl \rightarrow vpl, fb \rightarrow rd, rd \rightarrow vpl)$$

The GA population would consist of such 10 bit strings, each representing a set of models. The next step in GA implementation would be to determine a fitness or evaluation function that assigns an appropriate fitness value to a string.

4.6.5 Evaluation function

The evaluation function or fitness function plays a key role in the evolution of the population in a GA search. It is based on the objective of the search or optimization and assigns a fitness value to each chromosome based on its closeness to the final objective. For example, a chromosome that represents parameters that are very close to the parameters of the final desired output will be declared highly fit by the fitness function. Thus, the fitness function plays a role in enforcing the survival of the fittest principle. An incorrect evaluation would mislead the GA search away from the optimal solution and a function that is loosely bound to the desired output could result in an under-achieving GA. For these reasons, it is extremely important to use a correctly formulated fitness function that is directly and closely related to the desired output.

LID accuracy-based fitness The chief objective of the model set selection problem is to maximize the LID accuracy. Hence, a fitness function that is based on the LID accuracy of the feature-based system with a model set represented by the chromosome would be ideal. However, this function is computationally rather expensive. Every chromosome evaluation requires an LID run with the corresponding model set. A simpler fitness function, such as one based on a maximum-likelihood criterion could be considered. However, we have observed in the past that the n-gram likelihood scores obtained on the test data are not necessarily related to the final LID accuracy. For these reasons, we have decided to use an LID accuracy-based fitness function. One technique of making this fitness function more efficient would be to store the computed fitness values in a table. Every time a chromosome is re-encountered, its fitness can be determined by a simple look-up in the

table.

4.6.6 GA Operators

Once the parameter encoding and fitness function have been defined, starting with a randomly generated initial population, the GA can start processing successive generations of solutions using its operators. In our GA implementation, we considered the following different selection and crossover techniques:

- Selection:
  - roulette
  - tournament
  - stochastic universal sampling
- Crossover:
  - 1-point
  - 2-point
  - uniform

Apart from the selection, crossover and mutation operators, the elitist operator was also implemented. The GA parameters such as population size, maximum number of generations etc. as well as the type of selection and crossover operator were empirically tuned for the dependency selection problem.

Before considering further details of the GA implementation, a discussion of the baseline LID system will be given.

Chapter 5

BASELINE FEATURE-BASED LID SYSTEM

5.1 Introduction

In chapter 3, the theoretical framework behind an articulatory feature (AF) based LID system, was discussed. We will now discuss our baseline AF-based LID system. The baseline system has been built around the OGI-TS multi-lingual speech corpus. It contains spontaneous as well as fixed vocabulary utterances of varying duration in 10 languages, recorded over telephone lines. A detailed description of the corpus can be found in section 2.3.1 of chapter 2.

The first step in language ID is pre-processing of the acoustic signal. This is followed by acoustic decoding and finally, scoring of the decoded signal. The two essential components of an AF-based LID system are feature acoustic models and language models. In the following sections, a detailed description of both these components in out baseline system is presented. Further enhancements in the form of duration relabeling of alignments and speech/non-speech segmentation are also discussed. Finally, baseline results for our LID system are presented.

Along with the feature-based system, a comparable phone based system was also developed in parallel. The motivating factors behind developing a phone-based system were:

- It would provide a useful benchmark for comparison for the feature-based system
- Since the phone and feature systems model complementary information sources, combining the two systems might be advantageous. The phone system can be seamlessly integrated into the feature system as an additional feature stream.

The phone based system was developed along very similar lines as the AF system in terms of acoustic model training and n-gram model training.

5.2 Signal pre-processing

In most corpora, especially in conversational speech, it is likely that the training and test utterances, along with speech, also contain pauses and other non-speech events, such as laughter, breathing etc. Such non-speech events do not contain any language discriminatory information and are not only not helpful, but are likely to cause confusions in the language ID process. Hence, we segmented all the training and test utterances into speech non-speech units using a speech detection algorithm, and all non-speech segments were thrown out. The segmentation was done by a neural-network which was trained on a hand-labeled sub-set of the training data and was followed by a temporal smoothing of the network outputs.

A comparison of language ID performance with and without this type of speech non-speech segmentation showed that discarding the non-speech segments contributed towards significantly improving the language ID accuracy of the system. A discussion of the significance measure used and the method of computing LID accuracy are discussed in Chapter 6 section 6.3.

Several representations of the spectral information in a speech signal, some of which are discussed in chapter 2. In our system, we have used 13 Mel-frequency cepstral coefficients and 13 delta cepstral coefficients, sampled every 10 msec, yielding 26-dimensional feature vectors. Furthermore, techniques for noise robustness (for example, variance normalization), removing channel effects (for example, RASTA pre-processing (see chapter2, section 2.2.1) and speaker normalization (for example, VTLN (see chapter2, section 2.2.8.2) have been applied in the past, to the speech signal. However, in our baseline system, we have not implemented any of these techniques.

5.3 Acoustic models for AF based system

In this section, we will discuss various issues related to training feature-based acoustic models, including, the choice between language-independent and language-dependent models, HMM based acoustic models, choice of model topology, the HTK toolkit, model training and finally, model refinement techniques.

5.3.1 Language-independent vs. language-dependent acoustic models

The two major strategies for acoustic decoding involve implementing either language-independent (LI) recognizers or language- dependent (LD) recognizers. Language-dependent recognizers have the ability to model differences in the acoustic realization of features across different languages. In addition, since a test utterance is mapped onto more than one sequence, one per recognizer, they provide a richer representation of the utterance. Language model scores are available for each target language per recognizer providing a much wider choice of scores during score combination as compared to an LI system. However they also require a significantly larger number of parameters. On the other hand, in an LI system, training data from all target languages is pooled together for training the acoustic models, resulting in more robustly trained models. A large number of features are shared across languages and an LI system removes the redundancy of training separate models for each of them. A study of the available literature shows that LD systems significantly out-perform LI systems [11]. The tradeoff between LID accuracy and system complexity needs to weighed and the final decision is often driven by available resources and more importantly, available training data. The OGI-TS 10 language corpus contains, on average, roughly just over 2 hours of speech for each of the 10 languages, which is divided over training, development and test sets. Since this does not seem to be enough material to train individual acoustic models for each language, we have opted to use the LI framework of acoustic decoders for our AF-based LID system for the OGI-TS corpus.

5.3.2 HMM based recognizers

Different approaches to acoustic decoding include using GMMs, ANNs, and HMMs [29, 19, 10]. Each technique has its advantages and pitfalls. GMMs are computationally less expensive compared to ANNs and HMMS. ANNs are powerful classifiers, while HMMs can effectively model the temporal nature of the speech utterance. In all our experiments, we have used HMM-based acoustic decoders trained using the hidden Markov model (HTK) toolkit [65]. HMMs incorporate the principle of dynamic programming and are a powerful statistical tools for classifying patterns of time-varying sequences [63]. The HTK toolkit provides a set of library modules and tool for building, training, editing and decoding with HMMs. Though the toolkit has been used for applications such as speech synthesis and character recognition, its primary application has been in speech recognition research. We will now look at different aspects of HMM based feature recognizers including model topology, model training, decoding and model refinement.

An HMM is associated with a set of "hidden states" and a corresponding set of observable vectors for each state. In the context of speech recognizers, the hidden states can be assumed to model the articulatory configurations related to the linguistic speech unit (feature or phone or word) that the model represents and the corresponding observable vectors are the spectral features extracted from the speech utterance. The variations in acoustic realization of the speech-unit caused due to speaker variability, environmental variability etc. are modeled by the Gaussian that represents the observation vectors. In general, to better model the temporal nature of speech, a left-to-right state topology is followed. To further account for acoustic variability and co-articulation effects, skips between HMM state may be allowed. Our baseline system uses a three emitting state model with a skip from state one to state three as illustrated in figure 5.1.

A separate HMM is trained for each distinct feature variable in each feature group. Table 5.1 lists the articulatory feature groups in our system along with the number of models or symbols in each group. Each group contains separate models for silences and other non-speech events such as breathing, laughter etc. In all, there are 8 such non-speech

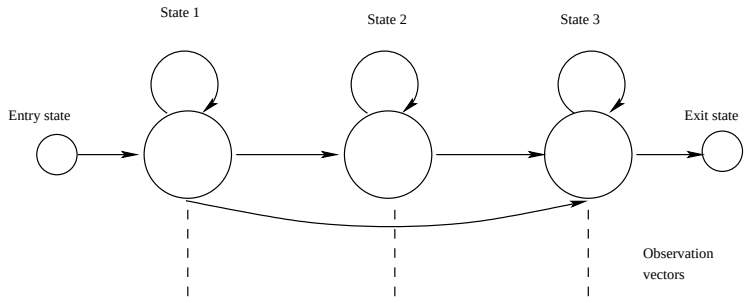

Figure 5.1: 3-state left-right topology for baseline HMMs

Table 5.1: List of AF streams in our baseline system along with the number of models in each stream

Feature	*cpl*	*fb*	*mann*	*nas*	*rd*	*voi*	*vpl*	*phone*
# models	18	12	16	10	10	10	12	133

models in each group. Though most of the non-speech events are eliminated from the utterance during speech/non-speech segmentation, they cannot be eliminated completely, and hence, separate acoustic models are trained for them.

In the first step, HMMs are built and trained using the available training data and in the next step, the HMMs are used to decode (or recognize) the test utterances. To begin with, an initial estimate needs to be made for the HMM parameters. The initialization program in HTK divides the observation vectors equally among all model states for making an initial estimate. It iteratively finds the maximum likelihood state sequence, reassigns the observation vectors and updates the HMM parameters accordingly. In Baum-Welch re-estimation, rather than assigning the observation vector to any particular state, it is assigned to every state with a probability proportional to the state occupation probability for that state at that time frame. The state occupation probabilities are computed using the *forward-backward* algorithm [65]. The main model training phase also uses the same

Baum-Welch algorithm, but all the models are trained in parallel. This training phase is also iterative and in general, the model is trained through 2-5 iterations of this type of embedded training.

The parameter set of an HMM includes the Gaussian mixture distributions associated with the observation vectors for each state and the transition probabilities between the states. Each state is associated with a single Gaussian component, however, it is possible to increment the number of mixture components as discussed in the following section. Transition probabilities between different states are stored in the transition matrix. A model with n states has a transition matrix with dimensions $n \times n$ where the element in the i^{th} row and j^{th} column represents the probability of transition from the i^{th} state to the j^{th} state.

Decoding is done using the *Viterbi* algorithm. A recognition network is used to represent multiple hypotheses in a recognition output. Nodes in this network represent speech units, such as words, phones, or as in our case, articulatory feature variables. This network is converted to a network of HMMs where each node represents an HMM, which in turn is a network of states connected by arcs, where each arc is associated with the corresponding transition probability. During decoding, the Viterbi search finds a path through this network that has the highest probability, given the sequence of observation vectors for the test utterance.

5.3.3 Acoustic model refinement

One of the shortcomings of the model set described above is that it assumes a uniform three state topology for all models thus ignoring the variations in duration that characterize the features that the models represent. Some examples of duration statistics for non-noise feature variables in streams *cpl* and *fb* are summarized in table 3.2 below.

A simple technique of modeling the duration characteristics of a feature is to update the number of emitting states in its model based on the *average* duration of its samples

Table 5.2: Duration statistics for some feature variables: average, minimum and maximum duration in milliseconds

Feature model	avg. duration	min. duration	max. duration
cpl8	85.3	1.0	1310.0
cpl9	56.7	4.0	646.0
fb8	76.7	1.0	1350.0
fb9	69.1	2.5	770.0

in the training data. Another variation of this technique would be to update the number of emitting states according to the *minimum* duration of its samples in the training data. We looked at both these techniques for improving out baseline acoustic models. Though duration modeling based on minimum duration did not make much difference, a significant improvement in recognition accuracy was observed with the average duration models.

Other techniques for model refinement include [65]:

- context-dependent models
- parameter tying
- clustering
- mixture component incrementing

In context-dependent models, several models are trained per feature (or speech unit) for different combinations of left and right neighbors. This technique helps to better model co-articulation effects which are predominant in conversational speech utterances. Parameter tying allows for sharing of parameter values at the state or model level. The ideal groups of parameters to be tied can be identified using data-driven clustering or decision tree clustering techniques. Another method of enhancing the HMM set is to increment

the number of mixture components in the Gaussian mixture distributions associated with the model states. All these techniques are supported by HTK tools. The actual degree of refinement of acoustic models achieved from these techniques depends upon the specifications of the system under consideration, such as size of the model set, available training data etc. We tried to implement some of these techniques, namely parameter tying and clustering, but did not record any improvement in performance with the same. Mixture component incrementing did result in an improvement in LID accuracy, though the improvement was not uniform across all feature groups. Finally, we applied mixture component incrementing only to those groups that showed a tendency to improve performance, such as the *mann* feature. The number of mixture components were increased iteratively and the final number of mixture components was determined empirically and set to 4 mixture components per state for *mann* and 2 mixture components per state for *cpl* and 1 mixture component for the remaining feature groups.

5.4 Language models for baseline AF based system

In our feature-based system, we use n-gram models to estimate the probability of a test utterance occurring in that language. Analogous to acoustic modeling, the two phases in language modeling are:

- training phase - parameter estimation and smoothing
- test phase - scoring the test sequence.

A description of parameter estimation in n-gram models and smoothing techniques can be found in chapter 3 sections 3.5.3 and 3.6 respectively. The important design criteria for n-gram models are the order of the n-gram and smoothing techniques. Our baseline system consisted of only within-stream n-gram models. Experiments with cross-stream models and model selection are described in the following chapter. Furthermore, rather than considering any subset of features at previous time positions within the stream, only

conventional n-gram models for different values of n were considered. Though experimenting with different subsets of the within stream features can lead to improved LID performance [17], we did not include these here, since the primary goal of these experiments was to investigate cross-stream dependency modeling and model-selection from the point of view of language ID.

At different points of time, we have used both the CMU and the SRILM toolkits for training and testing with our within-stream as well as cross stream dependency models. A description of these toolkits can be found in chapter 3, section 3.7 From these experiments, we found that there was no change in the LID accuracy across systems with LMs trained with either of the two toolkits. Finally, we settled on the SRILM toolkit, specifically, the factored SRILM toolkit [45] because of its extensible nature and more importantly, ability to model cross-stream dependencies. We experimented with different context lengths (i.e different values of n) for each of the feature groups and studied language identification accuracy for different combinations of feature groups with different context lengths. Though the effects of varying the context length were not uniform across feature groups, the system with best LID accuracy combined five feature groups (*cpl*, *fb*, *mann*, *rd*, *vpl*) using trigrams for modeling within-stream dependencies for each group. Scores from different feature streams were combined using the product rule (equation 3.6 from chapter 3). From these experiments we discovered that the *nas* and *voi* feature groups were not particularly useful for language ID. Hence, in the following experiments, we have not included these two streams. We also experimented with different discounting schemes available in the two toolkits and observed that though the different discounting schemes did not result in significantly different LID accuracies, in general, Witten-Bell discounting gave the best LID results across all feature groups.

5.5 System enhancement

With a front-end of acoustic models and a set of n-gram feature models described above, the basic language ID system was constructed. In section 2.2.7 chapter 2, techniques

to enhance LID systems in general, such as using gender-dependent acoustic models and using higher level classifiers for score combination etc. were studied. We have successfully applied some of these techniques to our baseline LID system. These include:

- duration relabeling
- speech/non-speech segmentation

We have already discussed speech non-speech segmentation in section 5.2.

There is considerable variation in the duration of a feature or phone unit across its instances in the corpus. This is reflected by the frame information as well as the number of copies of the feature or phone at the frame-level in the alignments output by the acoustic decoders. However, as explained in section 3.8.1, due to synchronization and smoothing issues, we keep only a single representative instance and discard all its repetitions. Besides, a sequence is stripped of its time frame information before n-gram training and scoring. Due to this, information about the variation in duration across instances of a feature are lost. A simple method of incorporating this duration information is by tagging each occurrence of a feature as either long or short based on its comparison to the average duration for that feature. A variation of this technique is to split the feature instances as long or short at a threshold other than the average duration and also at more than one threshold, resulting in a 3-way (or higher) split. We found that applying such duration relabeling techniques results in significant improvements in language ID accuracy. The best results were observed with a 2-way split with the threshold set at the average duration..

We also experimented with using higher-level classifiers such as GMMs and ANNs for score combination rather than using the simple product rule (equation 3.6) The classifier parameters were trained on the development set. Though this resulted in huge improvements in the LID accuracy of the development set, the improvements did not carry over to the evaluation set. This was probably because the parameters were over-training on the

development set. However, re-running the experiments with different parameter settings did not yield any improvements in performance on the evaluation set.

Acoustic properties of male and female speech are highly distinct. Gender based models have been effectively used in speaker identification, speech recognition as well as in language ID [11]. However, preliminary experiments with gender based n-gram models in our baseline system proved to be futile. There is a huge imbalance in the number of utterances spoken by male and female speakers in most languages in the OGI-TS corpus (for example in Tamil, there are 396 and 62 utterances respectively for male and female speakers). This could be one of the reasons why the gender dependent n-gram models did not help in our system.

5.6 Baseline language ID accuracy

In this section, we will discuss the baseline LID results with the AF based system described above. In the baseline system, only within-stream feature models were considered. LID results for the individual feature streams as well as the combined feature system are presented in Table 5.3. Results for the phone based system are also reported. These results are for the development set and evaluation set respectively, of the OGI-TS corpus which consists of 1898 utterances ranging from less than 3 sec. to up to 45 sec. in duration.

5.6.1 LID accuracy: very short, short and long utterances

The duration of an utterance is very important from the LID perspective. In almost all LID applications, a quick response time with a short test utterance is a key requirement. On the other hand, it has been observed in most LID systems that accuracy degrades significantly for shorter utterances [11]. It is thus essential to monitor the LID performance of a system over different utterance length durations. Table 1.2 reports the LID results for the development set with the combined feature-based system for utterances categorized according to their duration as very short (smaller than 3 sec.), short (between 3 to 15 sec.) and long (longer than 15 sec.). As expected, the LID results show improvements

Table 5.3: LID accuracy for baseline feature and phone systems

	% LID accuracy	
System	dev. set	eval. set
cpl	47.52	47.02
fb	38.88	39.61
mann	45.94	43.45
rd	38.72	36.69
vpl	35.67	39.07
cpl, *fb*, *mann*, *rd*, *vpl*	58.95	57.30
phone	49.84	47.99

with increasing utterance length.

Table 5.4: LID accuracy for baseline feature system for different utterance durations

	% LID accuracy		
Utterance duration	AF system	phone system	# utterances
very short ($\leq$ 3 sec.)	45.59	33.61	601
short ($\geq$ 3 sec. and $\leq$ 15 sec.)	62.18	56.73	1120
long ($\geq$ 15 sec.)	74.16	65.17	177

Results in table 5.4 show that the AF system significantly out-performs the phone system for all categories of utterances, namely very short, short and long utterances. Since our primary goal was to work on developing the feature-based system, efforts were concentrated on the AF system rather than the phone-based system. Enhancement techniques,

such as, speech/non-speech segmentation and duration relabeling were applied only to the feature-based system. For this reason, the baseline feature system has a much better performance than the baseline phone system. Results from comparable feature and phone systems are reported in one of our previous works [16]. These results showed that, though the phone system did better on the longer utterances, the feature-based system did significantly better on the shorter utterances, confirming our hypothesis that the richer AF representation could be significant for short utterances.

5.6.2 LID with utterance length modeling

The OGI-TS corpus consists of spontaneous and non-spontaneous utterances ranging from a couple of seconds to a maximum of about one minute. While the shorter utterances tend to be fixed vocabulary utterances, such as an enumeration of the days of the week or numbers from one to ten, the longer ones contain spontaneous speech. Thus, the length of the utterance also defines the speaking style and vocabulary effects. Explicitly modeling these effects with duration-specific n-gram models could lead to further improvements in language ID performance. To test this hypothesis, we grouped utterances into different sets based on their duration and trained n-gram models only on utterances from the specific duration based categories. Based on LID performance, for each duration category, either the duration-specific n-grams or the standard n-grams were selected for scoring. The duration categories considered for these experiments were: below 1 sec., 1-3 sec., 3-5 sec., 5-7 sec., 7-20 sec. and above 20 sec. These categories were selected such that the distribution of utterances across the different categories was balanced. LID results from these experiments are compared with the baseline LID results in Table 5.5.

These results indicate that duration specific n-gram models can contribute to further improving language ID performance of the system. However, as mentioned above, for the OGI-TS corpus, the length of the utterance also encodes information about speaking styles and vocabulary effects. Hence, improvements due to duration specific n-grams could also be due to these characteristics of the OGI-TS corpus rather than just utterance length modeling.

Table 5.5: Comparison of % LID accuracy for baseline system and for system with utterance duration modeling

data set	baseline system	utterance duration modeling
dev. set	58.95	62.85
eval. set	57.30	59.87

The parallel architecture of the feature-based system provides a rich representation of the speech utterance. By modeling cross-stream dependencies, we can further exploit the inherent advantages of a feature representation. We hypothesize that these modeling strategies may be especially advantageous for very short length utterances where a very small context is available. We will test this hypothesis by evaluating the LID accuracies across different durations on completion of our experiments with cross-stream models based on a genetic algorithm search. These results are presented in the following chapter along with a detailed description of our experiments in cross-stream dependency modeling and model set selection using genetic algorithm.

Chapter 6

EXPERIMENTS AND RESULTS

6.1 Introduction

A novel approach to language identification based on articulatory-phonetic features was presented in Chapter 3. The probabilistic framework for this approach was described and the motivation for modeling dependencies across feature streams was presented. Finally, the search problem of identifying the model set of within-stream and cross-stream dependencies from all possible combinations of dependencies was discussed. A detailed description of genetic algorithms was provided in Chapter 4, along with the reasoning behind the decision of using a genetic algorithm for model set selection. Issues related to applying a genetic algorithm for dependency selection in an articulatory feature-based language ID system were discussed. In Chapter 5, our baseline AF-based LID system was described.

In this chapter, we describe our experiments in model set selection along with a discussion of the experimental results. Apart from searching for a good set of models for the entire development set, we also conducted separate GA searches over different sets of development data, grouped according to the utterance lengths. Results from these experiments are also presented here. This is followed by a discussion of the drawbacks of the genetic algorithm, and the methods we employed to counter the same. Finally, we conclude with a comparison of the GA based search with the greedy search for the model set selection problem.

6.2 GAs for model set selection: experimental settings and parameters

Some important GA parameters that significantly impact the GA run are population size, maximum number of generations, etc. In our GA implementation, some of these parameters are set in a separate text file, while some parameters, whose value is less likely to change across GA runs, such as the termination criterion, are set in the initialization script. Following is a list of GA parameters along with their typical values. The typical values represent the values that were most commonly used in our various GA runs.

- population size (e.g. 200) - The population size is a function of the size of the search space and hence, depends on the next parameter, the string length. For example, a long string, represents a huge model set and entails a large population size for a thorough, balanced search.

- string length (e.g. 15) - The string length defines the total number of models in the search and thus determines the size of the search space. Each element of the string represents a dependency model.

- maximum number of generations (e.g 75) - Regardless of the status of the search, the GA stops when the number of generations processed equals this number

- crossover probability (e.g 0.8) - The probability with which pairs of individuals from the mating pool undergo crossover.

- mutation probability (e.g 0.001) - The probability with which any gene of any individual in the population is likely to be mutated.

- crossover operator (e.g. uniform) - three different types of crossover operators have been implemented, namely, 1-point, 2-point and uniform.

- selection operator (e.g tournament) - three different types of selection operators have been implemented, namely, roulette wheel, tournament and stochastic universal sampling (SUS).

- termination criterion (e.g. LID accuracy of best individual=90%) - when the termination criterion is satisfied, the GA stops, regardless of the number of generations processed.

- convergence criterion (e.g. 0.1) - when the difference in the fitness (in our case the % LID accuracy) of the best individual and the average fitness is less then the convergence criterion, then the GA stops, regardless of the maximum number of generations and the termination criterion

In our model-set selection searches, we experimented with different values of population size, crossover and mutation probabilities as well as the different selection and crossover operators.

Before applying the GA to our model-set selection problem, we implemented a GA to a simple string matching problem, with the chief objective of getting familiarized with the implementation and functioning of genetic algorithms. On these pilot experiments, we observed that, among the selection operators, the tournament and SUS operators were superior to the roulette wheel operator in terms of achieving the goal in fewer number of generations. The goal was to generate a user-defined string from a random population of strings. A distance measure was defined to evaluate how closely a string matches with the "goal" string. Similarly, the 2-point and uniform crossover operators were found to out-perform the 1-point operator. The GA performance was further improved (i.e. the GA required fewer number of generations to generate the goal string) when the elitist operator was implemented.

For the model selection problems, the values for population size, crossover probability, mutation probability and maximum number of generations were tuned empirically. The range of values for the population size was determined based on the number of models over which the search was conducted. The minimum population size was set to 100 while the maximum was around 600. With small population sizes, the GA did not always converge

on the best subset of models in terms of LID accuracy. Increasing the population size improved language ID performance, but beyond a certain point, increasing the population further did not lead to improvements in LID accuracy. The crossover probability was varied between 0.8 to 1.0 and did not seem to affect GA performance. Multiple runs of the GA with the same parameter settings did not necessarily select the same set of models. This is because of the randomness associated with a GA (for example, the initial population is generated randomly), as well as, the probabilistic components in its operators. However, though there was a difference in the models sets selected, there was very little or almost no difference in the language ID performances of these model sets.

As explained in section 4.6.1 of Chapter 4, we used ordered lists of feature groups to generate the list of cross-stream dependency models to be included in the search. The ordered list was used to ensure that the resulting model sets did not encode any circular dependencies. However, this strategy confines the search to a section of the search space and the entire space of possible combinations of model sets is not explored. To counter this disadvantage, we repeated the GA search with different orderings of the feature groups. Results from these experiments showed that the best model sets were discovered with the following ordering: $(f^{cpl}, f^{rd}, f^{vpl}, f^{fb}, f^{mann})$.

6.3 GAs for model set selection: initial experiments

All our experiments were implemented over four distinct groups of model sets:

- group A: within-stream feature trigram models + cross-stream feature bigram models

- group B: group A + cross-stream bigrams between feature groups and the phone stream

- group C: group B + phone trigrams

- group D: within-stream phone trigrams + cross-stream feature bigrams + cross-stream bigrams between feature groups and the phone stream

Composition of the different categories of model sets is summarized in Table 6.1.

Table 6.1: Composition of groups A, B, C and D

System	fea. trigrams	phone trigrams	x-stream fea. bigrams	x-stream fea.-phone bigrams
Group A	√		√	
Group B	√		√	√
Group C	√	√	√	√
Group D		√	√	√

There were multiple objectives behind this categorization. Group A was created to study the effect of cross-stream dependency modeling on the AF based system. Group B investigated the effect of adding some phone information (in the form of cross stream modeling between feature groups and the phone stream) to the AF system. The phone and feature sequences represent complementary sources of information and combining the two in a probabilistic framework could lead to significant improvements in language ID performance. This hypothesis is tested by group C which includes within-stream and cross-stream n-grams (trigrams and bigrams respectively) from both the feature and phone groups. Groups B and C seem to be very close, with the only difference between them being the phone trigrams. However, this difference becomes very significant from the point of view of number of parameters in the system. As explained in Chapter 3, the total number of phone models (133) is much larger than the number of models in the feature system, summed over all feature groups (89) and as a result, the total number of parameters for all the n-gram models in group B (33K) is only a small fraction of the total number of parameters in group C (2.38M). Hence, it would be interesting to study the two groups

separately and compare their LID performances. Group D has been created to study the effect of cross-stream dependency modeling on the phone system. It should be noted that the cross-stream bigrams included in these experiments represented dependency models with the conditioning and dependent variables from two different feature streams at the same time positions. Incorporating more complex cross-stream dependency models for language ID is discussed later in this section.

Language ID accuracies for the four groups are presented in Table 6.2. The language ID accuracy is computed as:

$$\%\, LID\ accuracy = \frac{\#\ correctly\ hypothesized\ utterances}{total\ \#\ utterances} 100 \tag{6.1}$$

An improvement of around 2.5% is considered to be significant for the development and evaluations sets. It translates to a significance at the 0.005 level using a difference of proportions significance test. Other relevant information, such as the total number of models in the search, number of models selected by the GA, total number of parameters in the system etc. are also reported.

Table 6.2: % LID accuracies with models selected by a GA search for groups A, B, C and D

	% LID accuracy				
System	dev. set	eval. set	total # models	# models selected	# parameters
Group A	60.22	58.44	15	6	31K
Group B	61.85	60.06	20	7	33K
Group C	64.54	62.17	21	6	2.38M
Group D	54.21	51.35	16	6	2.35M

Some interesting observations and conclusions from these experiments were:

- Significant improvements in language ID accuracy were observed for the development set in each group. These results should be compared to the baseline results in table 5.3.

- For the evaluation set, significant gains were obtained for groups B and D, compared to the baseline systems.

- For all the different runs of the GA with various settings, the GA always selected the within-stream trigram models for all the streams (features and/or phone) in that group, along with some cross-stream dependencies.

- Cross-stream dependencies are selected for groups A and B, both of which do not include the phone trigrams. Also, cross-stream dependencies are selected for group D, which does not include the feature trigrams. However, for group C, which includes both the feature and phone trigrams, no cross-stream dependencies are selected. This seems to indicate that the combination of within-stream trigrams from both systems and one set of within-stream trigrams with cross-stream bigrams models, are two different methods of modeling similar information. While including the phone trigrams gives the best LID accuracy, it comes at a significant cost in terms of number of parameters (column 4 of Table 6.2).

- It is impossible to predict which dependencies will be selected by the genetic algorithm, in the different GA runs. This points to strong non-linear interactions between the different conditioning variables.

For group A, along with the 5 within-stream trigrams, the cross-stream bigram for $cpl \rightarrow fb$ was selected. In group B, along with the feature trigrams, the GA selected 2 cross-stream bigrams, namely, $cpl \rightarrow phone$ and $rd \rightarrow fb$. Group B includes dependencies between feature and phone streams. There are two different varieties of such dependencies:

- $feature \rightarrow phone$, i.e. with a feature as the conditioning variable and the phone as the dependent variable.

- $phone \rightarrow feature$, i.e. with the phone as the conditioning variable and a feature as the dependent variable.

Both these dependencies for a particular feature-phone pair cannot be included simultaneously as that would encode a circular dependency. We carried out experiments with different combinations of both types of dependencies, always ensuring that there were no circular dependencies in the model set. Results from these experiments showed that the $feature \rightarrow phone$ type of dependencies were selected by the GA over the other type and model sets with these dependencies had better language ID accuracies. Hence, in all the following experiments, we have only incorporated dependencies of the type $feature \rightarrow phone$.

In the definition of the model selection problem, it was stated that a feature variable can have as its parents, any combinations of features preceding it in the feature list at any time positions within the context length under consideration. However, in the experiments described above, we have only used bigram cross-stream dependencies with the conditional variable from a different stream and at the same time position as the dependent variable. Since these models only cover a small group of the possible cross-stream dependency models, we next looked at other formulations of cross-stream models, namely:

- trigram models with parents belonging to different streams and at either the same time position as the dependent variable or at a previous time position. Example: $(f_i^{cpl}, f_{i-1}^{rd}) \rightarrow f_i^{fb}$ and $(f_i^{rd}, f_i^{vpl}) \rightarrow f_i^{phone}$.

- trigram models with one parent from a different stream and one parent from the same stream as the dependent variable. Example: $(f_i^{cpl}, f_{i-1}^{fb}) \rightarrow f_i^{fb}$ and $(f_{i-1}^{rd}, f_{i-1}^{phone}) \rightarrow f_i^{phone}$.

The exhaustive set of all such cross-stream trigrams is huge and we included only a few them in our GA search, with the objective of gauging their potential for language ID and verifying whether they were selected by the GA. Though the individual LID performance

of many of these models was superior to the cross-stream bigram models from the initial experiments, the bigram models were selected over any of these trigram models by the GA, along with the within-stream trigrams. As explained above, complex and non-linear relationships exist between the different feature groups and hence the corresponding n-gram scores which could be one of the reasons why the trigram cross-stream models do not get selected. However, as mentioned above, the experiments with such cross-stream dependency models were not exhaustive, and further experimentation could lead to lead to the discovery of more useful cross-stream models.

6.4 Model selection based on utterance length

The importance of n-gram modeling based of utterance length has been discussed in Chapter 5, section 5.6.2. The ideal set of dependency models for language ID could also be a function of utterance length. To test this hypothesis, we grouped utterances into different sets based on their duration. Separate GA based searches were carried out for the individual groups. To avoid language specific biases in the GA search, we sub-sampled the duration-specific groups such that there was roughly the same amount of data for each language within a group. Furthermore, to better model the utterance length effects, the n-gram models were trained only on utterances from the specific duration based categories and these were incorporated in the GA search for all duration groups for which they out-performed the regular n-gram models. Results from these experiments confirmed our hypothesis and there was considerable variation in the model-sets selected by the GA for the individual utterance length categories. The utterance length was characterized using two different parameters, namely, the actual utterance duration in seconds and the number of symbols in the utterance. Experimental results with both the criteria yielded very similar language ID results. Results for groups A, B and C for different duration specific categories are summarized in table 6.3.

Some interesting observations from these experiments were:

- The model-sets selected for the utterance duration-specific categories did not always

Table 6.3: % LID accuracies for duration-specific GA searches

		Group A		Group B		Group C	
Dur.	# utt.	% LID	# models	% LID	# models	% LID	# models
$\leq$ 1 sec.	301	71.1	9	71.1	9	74.4	10
1-3 sec.	239	44.4	11	54.0	7	55.6	9
3-5 sec.	490	63.1	7	65.4	7	70.9	8
5-7 sec.	321	60.3	7	68.3	6	70.5	7
7-20 sec.	124	65.6	8	72.0	9	73.6	10
$\geq$ 20 sec.	175	69.3	5	73.2	5	73.3	5

include all the within-stream trigram models, as in the case of the initial experiments.

- As can be seen from columns 4, 6 and 8 of table 6.3 in most cases the number of models selected by the GA was higher compared to the model sets from the initial experiments (column 4 of table 6.2).

- More cross-stream dependency models were selected in the duration-specific searches compared to the initial experiments.

The GA tries to select a model-set that maximizes the language ID accuracy of the development set and the selection process is hence a complex function of the n-gram model characteristics, the development set and the interactions between the different conditioning variables. As a result, it is difficult to interpret the composition of the GA selected model set based on linguistic knowledge. This further justifies our decision of using genetic algorithms rather than a heuristic or other conventional search algorithm for the model selection problem. We did see some trends in model set selection, certain cross-stream dependencies that get selected regularly by the GA across the different groups and with different experimental settings, such as $cpl \rightarrow fb$, $cpl \rightarrow phone$ and $rd \rightarrow fb$.

To gauge the contribution of duration-specific GA searches, LID results with duration specific n-grams with and without duration specific GA searches are summarized in table 6.4.

Table 6.4: Comparison of % LID accuracies with duration specific n-gram models for duration-specific GA searches vs. standard GA search

	duration-based GA		standard GA	
System	dev. set	eval. set	dev. set	eval. set
Group A	64.74	59.81	63.35	61.95
Group B	67.15	63.35	67.44	63.25
Group C	69.69	63.35	67.23	65.02
Group D	54.21	52.16	54.21	51.35

Results from these experiments indicate that applying duration-specific GA searches does not give significant improvements in language ID accuracies in most cases (except for group C) as compared to results from model sets selected by a standard GA. Moreover, the improvements on the development set are less likely to carry over to the evaluation set (see columns 3 and 5 of Table 6.4).

6.5 Genetic algorithms vs. greedy search for dependency selection

In the first set of pilot experiments to determine the potential of cross-stream dependency modeling for language ID, we used a greedy search for model selection with an earlier system [16]. However, the greedy search is a local search technique and hence, may not be suited for the dependency selection problem and for this reason we chose to apply the genetic algorithm to this problem. To confirm these suppositions, we repeated the dependency selection experiments with a greedy search (GS) algorithm with our current system. Dependency models were selected by the GS algorithms as follows:

- Make a list of all possible dependencies.
- Consider one dependency at a time and evaluate the LID accuracy in each case. Add the dependency giving the best LID performance into the system and remove it from the list of dependencies.
- Successively repeat the step above till the point where adding further dependencies does not improve the LID accuracy of the system.

GS-based search was applied over the entire development set to the four different categorizations of model sets, namely, groups A, B, C and D. Comparison of language ID accuracy for these groups with the GA and GS algorithms respectively are summarized in Table 6.5.

Table 6.5: Comparison of % LID accuracies with models selected by GA and GS for groups A, B, C and D

System	GA (% LID Acc.)	GS (% LID Acc.)
Group A	60.22	58.48
Group B	61.85	61.59
Group C	64.54	64.54
Group D	54.21	52.16

Results from these experiments show that the GA out-performs the GS in most cases and confirm that the GA is better suited for the dependency selection problem. Differences in LID results for the two algorithms are statistically significant for groups A and D.

6.6 GA based selection: drawbacks and counter measures

The language ID results on the development set for duration specific GA searches are summarized for the different duration categories in Table 6.3. The effect of duration-

specific GA searches on the development set as a whole, along with a comparison with standard GA search are summarized in table 6.4.

A comparison of the evaluation set results with duration modeling (columns 3 and 5 of table 1.3) to those of the baseline system (table 5.2, row 2) show that the improvements in LID accuracy for the evaluation set are modest compared to those for the development set. Furthermore, improvements in language ID with duration specific GA searches are less likely to carry over to the evaluation set.

This is not completely unexpected, as the fitness function of the GA is the language ID accuracy of the development set. The GA tries to identify the model-set that maximizes this accuracy as much as possible, and in the process, is likely to over-train. We experimented with two different approaches to counter GA over-training.

When selecting dependency models, it can be hypothesized that the GA is likely to select two types of dependencies:

- *generalized* dependencies - that are good for language ID and would contribute to improving LID performance for any data-set.

- *specialized* dependencies - that are not necessarily good for LID in general but are suited to the characteristics of the development set and hence improve LID performance.

The specialized dependencies cause GA over-training, and one technique to counter over-training would be to eliminate these dependencies from the model-set. Also, since the generalized dependencies are assumed to improve LID performance across data-sets, they would be selected by the GA irrespective of the composition of the development set. Based on these hypotheses, we ran the GA with different, randomly selected subsets of the development set. The number of utterances per language in the development set was maintained to be the same to avoid language-specific biases. From the model-sets selected for each development sub-set, we selected only those dependencies that were common to all of them. This experiment was further extended to consider all dependencies that

were common to at least k sub-sets, for different values of k. Finally, this model set of generalized dependencies was applied to the evaluation set. The experiment was performed with model sets for groups A, B and D. Though there were slight improvements in accuracy of the evaluation set for groups A (+0.52%) and D (+1.0%), there was a slight decline in accuracy for group B (-1.0%). Though these results were not significant, it was interesting to note that there was considerable variation in dependencies selected for the different development sub-sets. Some of the reasons why this technique was not very effective could be:

- The data set over which the GA is applied should be sufficiently large for the GA to select generalized dependencies. However, the OGI-TS corpus contains only a limited number of utterances per language. Moreover, the development data was further sub-divided by splitting into sub-sets for this experiment.

- Due to the complex, non-linear interactions between the different streams, it may be impossible to separate out generalized and specialized dependencies from the model set selected by the GA.

An alternative technique of countering over-training is to use the minimum description length (MDL) criteria during model selection. The MDL criterion advocates selecting the model that minimizes the description length of data encoded by the model plus the cost of the model. This is equivalent to maximizing the probability of data given the model minus the cost of the model [66] which is the Bayesian information criterion (BIC).

$$BIC(M, Y, X) = log(Y|M, X) - \frac{k}{2}log(n) \tag{6.2}$$

Where, M is the model-set, Y is the classification output, X is the data input, k is the number of parameters in the model set M and n is the size of the data-set. While the first term accounts for how well the model fits the data, the second term accounts for model complexity. So far, we have been using the % language ID accuracy as the performance measure for a model-set. For using the MDL criterion, the LID accuracy needs to be

converted into a probability distribution which can be done using the model set and the corresponding error function [66]. The second term in equation 6.2 is the cost of the model set, also known as the model complexity. The k in equation 6.2 stands for the number of parameters in the model set and can be interpreted in different ways.

In preliminary model complexity penalization experiments, we interpreted k as the number of dependency models in the model set. Multiple runs of the GA with this modified fitness criterion for different values of λ were carried out. Results from these experiments are summarized in table 6.6. These experiments included duration-specific n-gram modeling and the LID accuracies can be compared to those in table 5.5, column 3. No GA-based experiments were run for group C, since in this group only within-stream

Table 6.6: GA-based dependency selection with complexity penalization, where k = *# dependency models*

	without penalization			with penalization		
System	dev. set	eval. set	# models	dev. set	eval. set	# models
A	63.2	59.1	5	60.6	57.0	3
B	65.0	62.6	7	64.9	61.6	5
D	58.2	54.8	5	56.7	53.2	3

feature and phone models had been selected. These results show that, though the GA now selected fewer number of models, it did not show improvements in the evaluation set LID performance. We also looked at other interpretations of k, such as the number of n-grams seen by dependency models, summed over all models in the model set. This criterion was more complicated and required several GA runs with different values of λ to identify a meaningful value for λ that struck the right balance between the LID accuracy-based term and the model complexity term. In results from preliminary experiments with this criterion, we saw a similar trend as seen in results in table 6.6; though the GA now selected

model sets with fewer parameters, there were no improvements in the evaluation set LID accuracy as when compared to its accuracy without model complexity penalization.

6.7 Summary and conclusions

We have developed a novel approach to language identification based on articulatory-phonetic features. LID performance of such a system is comparable to that of the standard phonotactic approach. An AF based system has a much smaller parameter set, which could be especially important for LID systems with limited power, memory and computational resources.

We have shown that significant improvements in language ID performance are possible by explicitly modeling dependencies across different feature streams. Identifying the ideal model-set of within-stream and cross-stream dependencies for language ID is an NP-hard search problem. Due to the huge search space defined by all possible combinations of dependencies, an exhaustive search is not possible. We have shown that a good set of dependency models for language ID can be discovered by employing a genetic algorithm based search. A GA-based search out performs other heuristic search methods. In general, the ideal model set for the given data is a complex function of several factors such as languages in the set, n-gram models, size of the data set and its characteristics, utterance lengths, combination function etc., and it is not possible to make a knowledge based estimate of this model set.

Applying separate GAs to duration specific clusters of the development data does result in improvements in its language ID performance. However, such a GA-based search is more likely to over-train on the development data cluster and these improvements do not carry over to the evaluation set.

We have developed a generalized framework in which parallel streams of information about a speech utterance are combined together to perform language ID. The system is

scalable to further incorporating any other sources of information that can be expressed as sequences of discrete symbols, such as prosodic symbols.

BIBLIOGRAPHY

[1] Y. Muthusamy, N. Jain, R. Cole, "Perceptual benchmarks for automatic language identification," in Proc. ICASSP '94, vol. 1, pp. 333-336, 1994.

[2] I. Maddieson and I. Vasilescu "Factors in human language identification." in Proc. ICSLP '02, vol. 1, pp. 85-88, 2002.

[3] www.mc-mlmhs.org/interpreter/intserv/att.htm

[4] R. G. Leonard, "Automatic language identification," RADC/Texas Instruments, Inc., Dallas, Tech. Rep. RADC-TR-74-200/TI-347650, 1974.

[5] D. Cimarust and R. Ives, "Development of an automatic identification system of spoken languages: Phase I," in Proc. ICASSP '82, pp. 1661-1663, 1982.

[6] M. Sugiyama, "Automatic language recognition using acoustic features," in Proc. ICASSP '91, vol. 2, pp. 813-816, 1991.

[7] F. Goodman, A. Martin, R. Wohlford, "Improved automatic language identification in noisy speech," in Proc. ICASSP '89, vol. 1, pp. 528-531, 1989.

[8] M. Zissman, "Automatic language identification using Gaussian Mixture and Hidden Markov models," in Proc. ICASSP '93, pp. 399-402, 1993.

[9] A. House and E. Neuburg, "Toward automatic identification of the language of an utterance. I. Preliminary methodological considerations," in Journal of Acoustic Society of America, vol. 62, no. 3, pp. 708-713, 1977.

[10] M.Zissman, "Comparison of four approaches to automatic language identification of telephone speech," in Proc. ICASSP '96, vol. 4, no. 1, pp. 31-44, 1996.

[11] M Zissman, "Predicting, diagnosing and improving automatic language identification performance," in Eurospeech '97, vol. 1, pp. 51-54, 1997.

[12] T. Gleason and M. Zissman, "Composite background models and score standardization for language identification systems," in Proc. of ICASSP '01, pp. 529-532, 2001.

[13] D. Matrouf, M. Adda-Decker, J. Gauvain and L. Lamel, "Comparing different model configurations for language identification using a phonotactic approach," in Proc. Eurospeech '99, vol. 1, pp. 383-386, 1999.

[14] K. Berkling and E. Barnard "Language identification of six languages based on a common set of broad phonemes," in Proc. ICSLP '94, vol. 4, pp. 1891-1895, 1994.

[15] K.Berkling, T. Arai and E. Bernard "Analysis of phoneme-based features for language identification," in Proc. ICASSP '94, vol. 1, pp. 289-294, 1994.

[16] K. Kirchhoff and S. Parandekar, "Multi-stream statistical n-gram modeling with application to automatic language identification," in Proc. of Eurospeech '01, vol. 2, pp. 803-806, 2001.

[17] K. Kirchhoff, S. Parandekar and J. Bilmes, "Mixed-Memory Markov models for automatic language identification," in Proc. ICASSP 2002, vol. 1, pp. 761-764, 2002.

[18] T. Hazen and V. Zue, "Automatic language identification using a segment-based approach," in Proc. Eurospeech '93, vol. 2,pp. 1303-1306, 1993.

[19] Y. Muthusamy, R. Cole, M. Gopalkrishnan, "A segment-based approach to automatic language identification," in Proc. ICASSP '91, vol. 1, pp. 353-356, 1991.

[20] T. Hazen, V. Zue, "Segment-based automatic language identification," in Journal of Acoustical Society of America, vol. 4, pp. 2323-2332, 1997.

[21] Y. Muthusamy, "A segmental approach to automatic language identification," PhD thesis, Oregon Graduate Institute, 1993.

[22] A. Thyme-Gobbel and S. Hutchins, "On using prosodic cues in automatic language identification," in Proc. International Workshop of Parsing Technologies, 1995.

[23] F. Cummins, F. Gers and J. Schmidhuber, "Language identification from prosody without explicit features," in Proc. Eurospeech '99, 1999.

[24] T. Schultz, I. Rogina and A. Waibel, "LVCSR-Based Language Identification," in Proc. ICASSP '96, vol. 1, pp. 781-784, 1996.

[25] S. Mendoza, L. Gillick, Y. Ito, S. Lowe and M. Newman "Automatic language identification using large vocabulary continuous speech recognition," in Proc. ICASSP '96, vol. 2, pp. 785-788, 1996.

[26] D. Matrouf, M. Adda-Decker, L. Lamel and J. Gauvain, "Language identification incorporating lexical information," in Proc. ICSLP '98, 1998.

[27] K. Li, "Automatic language identification using syllabic features," in ICASSP '94, vol. 1, pp. 297-300, 1994.

[28] Y. Yan and E. Barnard, "Experiments for an approach to language identification with conversational telephone speech," in Proc. ICASSP '96, 1996.

[29] P. Torres-Carrasquillo, D. Reynolds and J. Deller Jr., "Language identification using Gaussian Mixture Model tokenization," in Proc. ICASSP '02, vol. 1, pp. 757-760, 2002.

[30] E. Wong and S. Sridharan "Methods to improve Gaussian Mixture Model based language identification system," in Proc. of ICSLP 02, vol. 1, pp. 93-96, 2002.

[31] y. Muthusamy, R. Cole and B. Oshika, "The OGI Multi-Language telephone speech corpus," in Proc. ICSLP '92, 1992.

[32] http://www.ldc.upenn.edu

[33] K. Kirchhoff, "Combining articulatory and acoustic information for speech recognition in noisy and reverberent environments," in ICSLP '98, 1998.

[34] K. Kirchhoff, G. Fink and G. Sagerer, "Conversational speech recognition using acoustic and articulatory input," in Proc. of ICASSP '00, 2000.

[35] K. Kirchhoff, G. Fink and G. Sagerer, "Combining acoustic and articulatory feature information for robust speech recognition," in Speech Communication 37, pp. 303-319, 2002.

[36] P. Ladefoged, "A Course in Phonetics," HBJ Publishers, Second Edition, 1982.

[37] K. Kirchhoff, "Robust Speech recognition using articulatory information," PhD thesis, University of Bielefeld, Germany, July 1999.

[38] F. Metze and A. Waibel, "A flexible stream architecture for ASR using articulatory features," in Proc. ICSLP '02, pp. 2133-2136, 2002.

[39] E. Eide, "Distinctive features for use in an automatic speech recognition system," in Proc. Eurospeech '01, 2001.

[40] S. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, 1998.

[41] A. Stolcke, "SRILM - An extensible language modeling toolkit," in Proc. of ICSLP, 2002.

[42] P. Clarkson and R. Rosenfeld, "Statistical language modeling using the CMU-Cambridge toolkit," in Proc. of Eurospeech '97, pp.2707-2710, 1997.

[43] R. D. Mori, "Spoken dialogues with computers," Academic Press, London, 1998.

[44] R. Rosenfeld, "Two decades of Statistical Language Modeling: Where Do We Go From Here?," in Proc. of the IEEE, 88(8), pp. 1270-1278, 2000.

[45] K. Kirchhoff and J. Bilmes, "Novel speech recognition models for Arabic," Johns-Hopkins University summer research workshop 2002, Final Report.

[46] D. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley Pub. Co., c1989.

[47] F. Korkmazskiy, C. Shih and C. Lee, "Pronunciation modeling by genetic algorithm," in Proc. of IEEE workshop on Automatic Speech Recognition and Understanding, pp. 173-176, 1999.

[48] A,. Belz, "Discovering phonotactic finite-state automata by genetic search," in Proc. of ACL, pp. 1472-1474, 1998.

[49] L. Araujo, "Evolutionary Parsing for a probabilistic context free grammar," in Proc. of RSCTC, 2000, pp. 590-597, 2000.

[50] K. Lua, "Part of speech tagging of Chinese sentences using genetic algorithm," in Conference of Chinese Computing, pp. 45-49, 1996.

[51] I. Poli and A. Roverato, "A Genetic Algorithm for Graphical Model Selection," in XXXIX Scientific Meeting of The Italian Statistical Society, '98, pp. 197-208, 1998.

[52] P. Larranaga, M. Poza, Y. Yurramendi, R. Murga and C. Kuijpers, "Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters," in IEEE Journal on Pattern Analysis and Machine Intelligence, 18(9): pp. 912-926, 1996.

[53] J. Yang and V. Honvar, "Feature subset selection using a genetic algorithm," in Proc. of IEEE Intelligent Systems, vol. 13, pp. 44-49, 1998.

[54] S. Russell and p. Norvig, "Artificial Intelligence: A Modern Approach," Prentice Hall Series in Artificial Intelligence.

[55] T.Dandekar and P. Argos, "Folding the Main Chain of Small Proteins with the Genetic Algorithm," in Journal of Molecular Biology, vol. 236, pp. 844-861, 1994.

[56] J. Lienig, "Physical design of VLSI circuits and the application of genetic algorithms," in Evolutionary Algorithms in Engineering Applications '97, pp. 277-292, 1997.

[57] C. Mira da Fonseca, "Mulitobjective genetic algorithms with application to control engineering problems," PHd thesis, Dept. of Automatic Control and System Engineering, University of Sheffield, 1995.

[58] A. Bickel and R. Bickel, "Determination of near optimum use of diagnostic resources using the GENES genetic algorithm shell," in Proc. of Computers in Biology and Medicine, 1990.

[59] K. Delibasis, P. Undrill and G. Cameron, "Design of texture filters with genetic algorithms: an application to medical images," in Proc. of Signal Processing, pp. 57:19-33, 1997.

[60] S. Novkovic "A genetic algorithm simulation of a transition economy: an application to insider-privatization in Croatia," in Computational Economics, pp. 11:221-243, 1998.

[61] Z. Sun, X. Yuan, G. Babia and S. Louis, "Neural network based gender classification using genetic search for Eigen feature selection," in Proc. of IEEE International Joint Conference on Neural Networks '02, 2002.

[62] M. Raymer, P. Sanschagrin, W. Punch, S. Venkataraman, E. Goodman, and L. Kuhn "Simultaneous Feature Scaling and Selection Using a Genetic Algorithm," in Proc. of the 7th International Conference on Genetic Algorithms '97, pp. 561-567, 1997.

[63] L. Rabiner and B. Juang, "An introduction to hidden Markov models," in IEEE ASSP Magazine, pp. 4-16, 1986.

[64] Y. Bengio, R. De Mori, G. Flammia and R. Kompe, "A comparative study of hybrid acoustic phonetic decoders based on artificial neural networks," in Proc. Eurospeech '91, 1991.

[65] http://htk.eng.cam.ac.uk

[66] P. Grnwald, "The Minimum Description Length Principle and Reasoning under Uncertainty," in ILLC Dissertation series 1998-03, University of Amsterdam.