Automatic determination of sub-word units for automatic speech recognition

Fiona Couper Kenney

Thesis submitted for the degree of Doctor of Philosophy

University of Edinburgh

(2008)

Abstract

Current automatic speech recognition (ASR) research is focused on recognition of continuous, spontaneous speech. Spontaneous speech contains a lot of variability in the way words are pronounced, and canonical pronunciations of each word are not true to the variation that is seen in real data.

Two of the components of an ASR system are acoustic models and pronunciation models. The variation within spontaneous speech must be accounted for by these components. Phones, or context-dependent phones are typically used as the base sub-word unit, and one acoustic model is trained for each sub-word unit. Pronunciation modelling largely takes place in a dictionary, which relates words to sequences of phones. Acoustic modelling and pronunciation modelling overlap, and the two are not clearly separable in modelling pronunciation variation. Techniques that find pronunciation variants in the data and then reflect these in the dictionary have not provided expected gains in recognition.

An alternative approach to modelling pronunciations in terms of phones is to derive units automatically: using data-driven methods to determine an inventory of sub-word units, their acoustic models, and their relationship to words. This thesis presents a method for the automatic derivation of a sub-word unit inventory, whose main components are

1. automatic and simultaneous generation of a sub-word unit inventory and acoustic model set, using an ergodic hidden Markov model whose complexity is controlled using the Bayesian Information Criterion
2. automatic generation of probabilistic dictionaries using joint multigrams

The prerequisites of this approach are fewer than in previous work on unit derivation; notably, the timings of word boundaries are not required here. The approach is language independent since it is entirely data-driven and no linguistic information is required. The dictionary generation method outperforms a supervised method using phonetic data. The automatically derived units and dictionary perform reasonably on a small spontaneous speech task, although not yet outperforming phones.

Acknowledgements

There are many people whose influence and support have made the completion of this thesis possible. Many friends, family and colleagues come to mind when I look back over the years I have been a PhD student. Thank you to so many who have taken an interest in what I've been doing.

Specifically I would like to thank my supervisors, Simon King and Steve Renals, for their consistency, support, expertise and patience throughout the years this has taken. I have learnt a lot from you both.

I would also like to thank my examiners, Rob Clark and Mark Huckvale, for an interesting and enlightening discussion, which enabled this work to come to completion in a way that has been satisfying.

CSTR has been a great place to work, and I am grateful to all my colleagues there for making that the case. For your approachability, help and friendship, thank you Korin Richmond, Joe Frankel, Rob Clark, Mike Lincoln, Peter Bell, Cassie Mayo, Zhang Le, Avril Heron. In particular I am grateful for many thoughtful and thought-provoking conversations with Mirjam Wester: thank you.

Without my family this thesis would not exist. Thank you for all your support - I know you've all rooted for me. Thank you Mum, Dad, Hanna and Joanna for providing childcare at crucial times. Dad, thank you for helping me think through various maths problems and coding issues along the way.

Finally, thank you Laurence for being so dependable and believing in me. We've got a lot to show for the past few years... what's next?!

Dedication

This thesis is dedicated to my children, through whom and for whom I have learnt so much and will go on learning.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(Fiona Couper Kenney)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Automatically derived units

Pronunciation variation in spontaneous speech is considered to be a major limiting factor on the performance of automatic speech recognition (ASR). In an ASR system, pronunciation variation is largely modelled by the dictionary and the acoustic models, where the dictionary may contain pronunciation variants, and the acoustic models account for variation beyond these. A typical ASR system uses phones[1] (or context-dependent phones) as the sub-word unit: acoustic models are trained for each phone, and the dictionary defines the relationship between words and phones, thus making explicit all allowable phone sequences.

Canonical pronunciations of words are very often extremely different to realised pronunciations in spontaneous speech. There is therefore a good motivation to look for the realised pronunciations and define these in the dictionary, such that the dictionary more closely reflects the data. However, explicitly modelling pronunciations by extending the dictionary in this way has not led to the gains in recognition performance expected (see Strik & Cucchiarini 1999). The reasons for this are unclear. It is possible that assuming that speech is phone-based is limiting advances in recognition. Phones are an abstraction, and since human transcribers cannot always agree on

[1]phones are the acoustic realisations of 'phonemes', where phonemes are phonetic units defined to be discriminative within a language

the boundaries between phones or even their identity, it can be argued that their use makes a poor starting point for acoustic modelling. The fact that the unit inventory is determined independently of the choice of statistical model may also be a cause for the performance ceiling currently experienced in recognition. A typical system has hidden Markov models (HMMs) for each phone (or context dependent (CD) phone), but it is possible that phones are not optimal for modelling by HMMs. Further discussion of these issues is found in Chapter 2.

Typically a phone set is defined and the data is transcribed in terms of these phones. Acoustic models are trained on the basis of this transcription. Pronunciation variation must be accounted for in these acoustic models or in the pronunciation dictionary, which clearly is also in terms of the pre-defined phone set. Alleviating this dependence on phones and instead determining the sub-word unit (SWU) inventory automatically may result in units which more closely reflect the data and so deal with some of the pronunciation variation. Consider a unit inventory which is learned from data automatically using a likelihood objective function, and a dictionary designed in conjunction with these units. In such a system, the mismatch between the actual realisations and the transcriptions (in the dictionary) will be minimised. Descriptions of the pronunciations in the data in terms of the new unit inventory will account for the variation within the data in a way that phones cannot. Two realisations of the same word whose differences are not captured in the phonetic dictionary will have different sequences of automatically derived SWUs. So the variation between the pronunciations can be captured using the SWUs. Theoretically then, pronunciation variation can be dealt with in a better way using automatic SWUs.

A process able to automatically construct a unit inventory and dictionary using only acoustic speech data and word transcriptions could be used to construct ASR systems for languages that have fewer existing resources than, say, English. This is another motivation for research into automatically derived units. Many English language corpora exist which are transcribed at high levels of detail, and there continues to be funding to collect data in English (among other languages). For languages where transcribed acoustic data and pronunciation dictionaries exist, it is possible, even straightforward, to build working ASR systems. However, this is not the case for many of the world's languages. Defining the phone set and writing a pronunciation dictionary requires linguistic knowledge, skill and time, and is therefore expensive, and so ASR is not possible for many languages. Given a way of automatically transcribing, modelling and describing pronunciations of any speech data, speech recognition becomes possible.

1.1.1 Research Aims

This thesis contains an investigation into methods able to derive sub-word units automatically from data. In any research, is is valuable to consider questions at a higher level than the details of the investigation, to learn more than specific processes and results, and compare new systems to what is standard. The following research aims are considered in this thesis.

- **How feasible is it to do ASR in this way?** What are the added computational costs of searching for a unit inventory? How scalable are process which automatically determine unit inventories? What are the data requirements?
- **How well does it work, compared to using phone models?** Can a new method overcome weaknesses of standard methods? Word error rate is the obvious indicator. Other comparisons include computational storage costs, and initialisation requirements.
- Finally, it is useful to evaluate the **strengths and weaknesses of doing ASR this way.**

The investigation into this aspect of ASR requires solutions to a number of specific questions. Using an automatically derived unit inventory necessarily affects the acoustic modelling and the pronunciation modelling of an ASR system. There are a number of questions which must be answered in order to devise the inventory:

- how many SWUs should the unit inventory have?
- how should the acoustic data be segmented to give the data for each SWU, and clustered to allow modelling of each SWU?
- what type of acoustic model should be used, and how can the acoustic models and the unit inventory size be jointly optimised?

- how are the SWUs related to words: how are words pronounced in terms of the SWUs?

1.1.2 Research approach

The above questions are investigated in this thesis, with the goal that any methods developed to derive a unit inventory will be fully automatic, and model- and data-driven:

- Fully automatic: the process should be fully automatic from start (raw data, including a word transcription) to finish (a unit inventory and dictionary relating words to new units). Manual intervention should not be required at any stage.
- Model-driven: the process should take into account, from the outset, the type of statistical models that will be used for acoustic modelling. The models themselves should be involved in segmenting the data. This will avoid the (typical) two stage process in which the data is segmented according to one criterion (automatically or manually), and then the segments are modelled. This two-stage process is unlikely to be optimal. A model-driven method should result in a unit inventory which fits the data and the model set well - if the model type were to change, the product would be a different unit inventory.
- Data-driven: Often a unit inventory is chosen independently of the data set. This is the case when phones are used as the sub-word unit; it is also true when data is segmented automatically according to a linguistic goal. A data-driven approach will take into account the particular characteristics of the given data. As for the model-driven goal above, if a different data set is used, a different unit inventory would be expected to be found.

1.1.3 Contribution of this thesis

This thesis contains methods which provide answers to all four questions in Section 1.1.1 above. The first three are solved in one simple process involving an ergodic HMM to simultaneously and automatically determine the unit set, train acoustic models for each unit, and segment and cluster the data. The number of units is determined using the Bayesian information criterion (BIC). This process differs from that seen in reported methods of inventory derivation in that it is so compact: in previous reported methods, units are often derived by iterating between separate processes of segmentation, clustering and acoustic modelling. In most previous methods, the number of units is pre-defined. The use of BIC here avoids this design limitation. The question of how to relate new SWUs to words is answered in this thesis using joint multigrams, providing a probabilistic dictionary automatically given the unit and word transcriptions. The product of the combination of these methods is a working speech recognition system. The construction of the full system requires minimal initialisation: word boundaries are not required, which differs from almost all existing methods; the number of units in the inventory is determined automatically; transcription of the data is only needed at the word level; and finally, no linguistic interpretation of the acoustic sub-word units is necessary since the whole process is automatic.

1.2 Outline

The remainder of the thesis is ordered as follows. In Chapter 2, the motivations behind searching for automatically derived SWUs are further explored, and a review of the other available methods for deriving SWUs is presented and discussed. In Chapter 3, the problem of automatic data segmentation is explored. Segmentation and clustering techniques are reviewed, and a method involving an ergodic HMM to automatically and compactly generate the data segmentation and unit inventory is presented. Dictionary generation methods are considered in Chapter 4. A semi-automatic, greedy method is used to provide a baseline, and a fully automatic method using joint multigrams is presented. The two types of dictionary are compared using phonetic data. Results and investigations of the combination of the unit derivation and dictionary generation are presented in Chapter 5. Finally, there is a concluding chapter.

1.3 Glossary of abbreviations

Abbreviations that are used in this thesis are listed here for ease of reference.

ASR	automatic speech recognition
BIC	Bayesian information criteria
CD-phone	context dependent phone
HMM	hidden Markov model
HTK	hidden Markov model toolkit
SWU	sub-word unit

Chapter 2

Unit inventory determination within statistical speech recognition

This chapter contains a brief description of statistical speech recognition, as the context for unit inventory determination. A number of recognition systems which use automatically derived sub-word units (SWUs) are reviewed, including a discussion of the challenges involved in SWU generation. The techniques presented in this thesis are introduced as part of the discussion of existing methods.

2.1 Statistical speech recognition

The goal of automatic speech recognition (ASR) is to transcribe speech automatically and accurately: to output the text of what was said. It is possible to compare different ASR systems based on the ability of each one to recognise the same speech, by comparing their output word transcriptions.

In statistical speech recognition, the problem of assigning a word sequence W to an input observation of speech, X, is posed as maximising the likelihood of the word sequence, given the speech data.

$$\hat{W} = \underset{W \in \mathcal{W}}{\operatorname{argmax}} \; P(W|X) \tag{2.1}$$

$\mathcal{W}$ is the set of all possible word sequences. $P(W|X)$, the probability of the word sequence W given the speech X, cannot be directly evaluated. Using Bayes' Rule[1], equation 2.1 can be rewritten as

$$\hat{W} = \underset{W \in \mathcal{W}}{\operatorname{argmax}} \frac{P(X|W)P(W)}{P(X)} \tag{2.2}$$

$P(X)$ is the prior probability of the speech, which is constant over the maximisation and so can be ignored. Thus equation 2.2 is reduced to

$$\hat{W} = \underset{W \in \mathcal{W}}{\operatorname{argmax}} \; P(X|W)P(W) \tag{2.3}$$

$P(X|W)$, the probability of speech X given a word sequence W, is computed by statistical models. The model set able to calculate probabilities for all combinations of X and W is referred to as the **acoustic model** set. The model able to compute probabilities $P(W)$ is the **language model**.

The speech waveforms are not used directly for X, but instead are reduced to a series of **feature vectors**, $X = x_1 ... x_i$, by extracting information from the waveform at regular time intervals. There are many ways to do the feature extraction, for example, linear predictive coding, filter bank coefficients, mel-frequency cepstral coefficients (MFCC). See Rabiner & Juang (1993) for an introduction to each of these. Each type of parameterisation aims to extract information useful to determining speech sounds, while reducing the effect of other information, such as amplitude and fundamental frequency (F0), contained within the waveform, but not interesting for speech recognition tasks. The most common, and the only parameterisation used in experiments reported in this thesis, is the MFCC.

The word error rate (WER) is the score used to rate and compare different speech recognition systems. WER is defined as

[1]Bayes' Rule states that $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

$$\text{WER} = \frac{S+D+I}{N} * 100,$$

where S is the number of word substitutions, D word deletions, I word insertions, and N is the total number of words in the utterance for which the WER is calculated.

2.2 Overview of a general ASR system

The acoustic models are the building blocks of the system. In standard ASR systems, the **hidden Markov model (HMM)** is the statistical model used to represent the acoustics. Alternatives to the HMM are regularly investigated; Digalakis et al. (1991), Kannan & Ostendorf (1998), Ostendorf et al. (1996), Ostendorf & Digalakis (1991), Frankel (2003) all present ASR systems where the HMM is not the basic acoustic model. However, in this thesis, only HMMs have been used.

HMMs were introduced in the 1960s by Baum and colleagues. A good introductions to HMMs can be found in Rabiner & Juang (1993, chap. 6).

An HMM is made up of states and transitions between states. The states 'emit' observations with a certain probability. The probabilities are modelled using Gaussians or mixtures of Gaussians. There are many ways an HMM can be organised: given a set of states, it is possible to allow or disallow any transitions between states, including self-transition. A typical set-up of an HMM used for speech is shown in Figure 2.1.

In ASR, the HMM is used to model regions of speech, such as words. In a simple word recognition task, one HMM would be built for each word. It will be trained on tokens of this word spoken by one or more speakers. Then in decoding, these word models compete to describe the observed speech, each one able to describe the observations with a particular probability. The sequence achieving the highest likelihood is the one which generates the word transcription. It is possible to train the system such that certain words or word sequences have a higher likelihood. This is done using a **language model**.

Training whole word models quickly becomes infeasible as the number of words increases. For a model to exist for each word, many models are required, leading to a complex and computationally expensive system. To train a useful model for each word

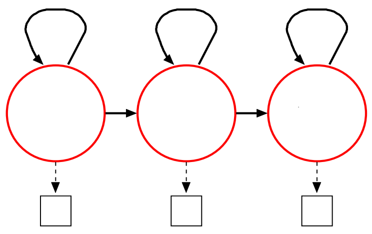
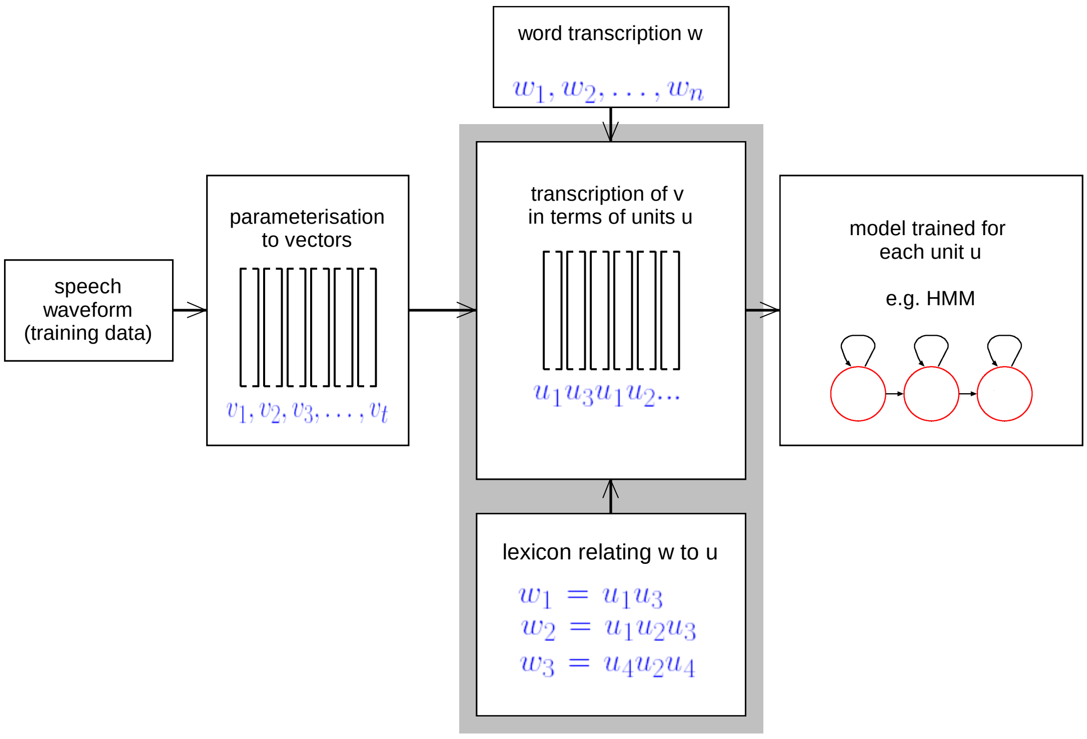


Figure 2.1: A typical HMM used to model a segment of speech for speech recognition. The states are shown by circles, and the observations by squares. The solid lines with arrows represent allowable transitions between states, each has a certain probability in the model. The dotted lines represent observation probabilities. This is referred to as a '3 state left-to-right HMM'.

it is important to have enough examples of that word among the training data. Avoiding data-sparsity becomes increasingly difficult as the vocabulary of the task increases. Words which are not well represented will be poorly modelled, and thus the system will be weak. Instead, smaller 'sub-word' units are typically modelled - units which are repeated across words, and therefore lead to a system with fewer models. In a typical system, phonetic-based units are used, such as phones, tri-phones, syllables. Phonemes are sub-word units which are defined to be contrastive units within a language, and are identifiable by phoneticians. It follows that, since the ear can distinguish different phonemes (and possibly distinct realisations of them: referred to as phones), and these distinctions are significant in differentiating words in language, models of phonemes or phones should provide a good basis for speech recognition. In order to train a set of phone models, we need training data which is transcribed at the level of the phone (which can be created automatically given a word transcription and a dictionary). Then for each phone model, all tokens of that phone are used for training.

The ASR system based on phones rather than words needs to have some way of relating the phones to words in order to output a word transcription. This relationship is described in a dictionary (or lexicon). The dictionary is used in decoding to enable the recogniser to output word sequences rather than phone sequences. The dictionary also defines all allowable phone sequences within the language, thus constraining the decoding search: the dictionary provides a **pronunciation model**.



word transcription w
$w_1, w_2, \ldots, w_n$
speech waveform (training data)
parameterisation to vectors
$v_1, v_2, v_3, \ldots, v_t$
transcription of v in terms of units u
$u_1 u_3 u_1 u_2 \ldots$
model trained for each unit u
e.g. HMM
lexicon relating w to u
$w_1 = u_1 u_3$
$w_2 = u_1 u_2 u_3$
$w_3 = u_4 u_2 u_4$

Figure 2.2: Diagram showing the main steps required in the training of a speech recogniser, based on sub-word units, u. The dictionary is not essential in training, if a full transcription is available. The shaded box indicates the areas of the system which are investigated in the design of a system based on automatically derived sub-word units.

The diagram in Figure 2.2 shows the main inputs and processing steps required to train an automatic speech recogniser. The choice of sub-word unit, u_i, is pre-defined, and the major output of the training process is a model set, where one model is defined for each sub-word unit.

2.2.1 Pronunciation variation

The tasks tackled by the ASR community have increased in complexity from the recognition of words spoken in isolation, to carefully read speech, to spontaneous, conversational speech. A lot of the difficulty in recognising spontaneous speech is derived from the huge amount of variation in the pronunciations of each word. Some of the variation may be characterised by substitution, deletion or insertion of phones, as well as various co-articulatory affects and vowel reductions. Modelling the variation is a major consideration in an ASR system. According to Strik & Cucchiarini (1999), there are three levels of an ASR system at which pronunciation modelling can take place, namely the lexicon (dictionary), the acoustic models, and the language model. (Strik & Cucchiarini (1999) is a review paper of a large number of works in pronunciation modelling to which the reader is encouraged to refer as an overview of the field.)

Pronunciation variation in the dictionary

The most obvious place where pronunciation modelling takes place is the dictionary. Pronunciation variation can be modelled by adding variants of words to the dictionary. The question of how this is done has been tackled in many ways, with variants being determined using linguistic knowledge, and/or variants being determined in a data-driven way. A knowledge-based approach uses information gathered from phonetic and linguistic studies, and incorporates expected variants into the dictionary based on these studied phenomena. For example, Kessens et al. (1999) investigated adding 5 phonological rules involving either phone insertion or phone deletion to the lexicon, and saw an improvement in recognition. Wester (2003) compared this approach to a data-driven approach, and found that the automatic, data-driven method resulted in a larger performance gain. Knowledge-based approaches are not reviewed here since they cannot be used with automatically derived units without some kind of mapping between the units and phones, since the ‘knowledge’ is in terms of phones. Using a mapping is undesirable, since it requires a confidence in phones that we do not have (see Section 2.2.2).

Data-driven methods aim to determine pronunciations which exist in the data and amend the dictionary to reflect these. In order to collect pronunciation variants from the data, some form of transcription is required. When manual phone labelling exists for a corpus, this can be used, but more often the output of a phoneme recogniser provides the transcription, e.g. Heine et al. (1998), Sloboda (1995), Sloboda & Waibel (1996), Westendorf & Jelitto (1996). Gathering the various pronunciations for each word can be done by splitting the phone transcriptions at word boundary locations, e.g. Sloboda (1995), or by aligning the transcription to a transcription obtained by expanding the orthography using the dictionary, e.g. ten Bosch & Cremelie (2002), Wester (2003). An alignment is achieved using dynamic time warping in Heine et al. (1998), and a viterbi search in Westendorf & Jelitto (1996).

The decision about which pronunciation variants to include tends to be based on a threshold, e.g. N-best (ten Bosch & Cremelie 2002, Sloboda & Waibel 1996). A neural network is used in Fukada et al. (1998) to predict alternative pronunciations from canonical pronunciations, and a threshold on probability of the predictions is used. Decision trees are used in Riley (1991), Riley & Ljolje (1995), Fosler-Lussier (1999), Wester (2003) to predict alternative pronunciations by determining the probability of a realised phone given the predicted (canonical) phone and the context. Choosing the optimal number of variants per word is difficult, and no metric yet exists for predicting the value of a dictionary, although some attempts have been made, e.g. Wester (2003). In general, fewer than 2 pronunciations per word is found to be optimal; a greater number of variants increases the confusability of the dictionary and degrades performance.

Pronunciation variation in acoustic models

The acoustic models also model pronunciation variation to some degree. An ASR system which uses a dictionary of canonical pronunciations forces the acoustic models to account for the variation which exists in the data. The inevitable mismatch between the dictionary and acoustic signal will lead to contaminated acoustic models (as put by Wester (2002, p.10-11)), and sub-optimal recognition performance. Adding variants to the dictionary which are present in the data should remove some of this contamination, as the match between the 'actual' phone sequences and the pronunciations in the dictionary increases. Any amendments to the dictionary directly affect the acoustic models, since it is standard to generate a transcription of the data using forced alignment according to the dictionary and data. This transcription is used to retrain the models. This method is noted to improve recognition scores by, for example, Bacchiani & Ostendorf (1999), Sloboda & Waibel (1996), Wester et al. (1998). Yang et al. (2002) discovered that in certain cases context independent acoustic models used along with a dictionary containing variants could compete with context dependent acoustic models, demonstrating the interaction between the dictionary and the acoustic models, and pointing to the importance of a data-specific dictionary. Hain (2002) shows that implicit pronunciation modelling using a carefully constructed single pronunciation dictionary performs at least as well as explicit modelling using multiple pronunciations per word.

2.2.2 Using different sub-word units

Since phonemes are defined to be discriminative within a language, it is an obvious choice to use their (perceived) realisations - phones - as the sub-word unit in ASR. The inventory of phones is a closed set: for different data in the same language, the same unit inventory will be used. Phones are useful since their relationship to words is known (it is possible to manually or semi-manually write dictionaries, and off-the-shelf dictionaries in terms of phones exist), and so they provide a convenient way of segmenting speech into units smaller than words (avoiding the data-sparsity problem of whole word modelling). There are weaknesses in using phones for ASR, however. Co-articulation effects in spontaneous speech are not well modelled by phones, and context-dependent models are used to overcome this weakness. The phone sets that are used are not necessarily valid for real speech, since they make assumptions about the realisation of speech. The ‘dubious nature of phone transcriptions’ is cited as a reason for the limited success of pronunciation variation modelling in Wester (2002, p.20), and the suggestion that other units are devised is made. Another limitation suggested by Wester, echoing Ostendorf (1999), is the assumption typically made that speech consists of sequences of phones strung together, as ‘beads-on-a-string’, whereas speech and language consist of many layers of information, whose sequences are asynchronous. In investigating automatically derived units, the work in this thesis moves away from phones, but the ‘beads-on-a-string’ paradigm is still used: the phone beads are simply replaced by different beads.

Using a pre-defined unit inventory does not allow for a match between the units and the models. The model type and unit set are chosen independently, and the models are forced to represent the units. This can lead to contaminated models, as mentioned in Section 2.2.1. This is particularly true for modelling spontaneous speech. Also using a pre-defined inventory does not allow for variation (of the unit inventory) in different types of data - the same phoneset is used for single speaker recognition, and for spontaneous speech with multiple speakers. There will be more variation in the second dataset, and it is likely acoustic modelling will be better (more accurate) if a different unit inventory is used. Some of this variation is captured in phone-based systems using triphones and parameter tying. However, it is reasonable to believe that a unit inventory which is devised in conjunction with the acoustic models will lead to better modelling of the acoustic space for given data.

2.3 Deriving units automatically

The focus of this thesis is the derivation of a unit inventory. In a system where the unit inventory is not pre-defined, the transcription of the data and the dictionary must both be generated. They are interdependent and should be jointly determined. The shaded box of Figure 2.2 highlights these investigated system components.

There are different ways to approach the problem of jointly determining the unit inventory and the lexicon. An intuitive approach is to consider the problem as three steps, or three areas which require solutions:

1. The speech signal must be segmented into units in some way, based on some criteria
2. The segments of speech found must be clustered into units such that the segments can be labelled, with the segments having the same label being similar in some way. The result of this step with the output of step 1 is a transcription of the data in terms of sub-word units.
3. Relationship between each word and sequences of units must be established. These relationships form the pronunciation dictionary.

Two other significant factors in the design of a unit inventory are

- the size of the unit inventory
- the acoustic models representing each unit

This deconstruction of the problem into three steps is essentially the way in which the research in this thesis has been approached, and the structure of the thesis reflects this.

This formulation of the problem into three steps implies that a good choice of solutions to three independent problems will lead to a good unit inventory. However, while it is possible to devise such a sequential approach, there will be no guarantee of joint optimality of the units and lexicon if they are not in some way jointly determined, since they are so interrelated. All methods of unit inventory derivation published to date involve iterative processes, in order to achieve this joint determination.

There are many facts that are taken for granted in a phone-based system which are unknown at the outset in training a sub-word unit (SWU) based ASR system. Most fundamentally, the number of SWUs in the system is not known. Clearly this is never the case in a phone-based system. The number of units per word is not known, since there is no lexicon, and indeed the number of units in each token of the same word may not be the same. Similarly, the length of each unit is not known, and may not be similar across different occurrences of the same unit. Word boundaries and unit boundaries are also unknown. When the output of one process determines one of these variables, its value will inevitably effect another, and so should be re-computed. In order to make the search for units and pronunciations tractable, some of these factors can be fixed. As we will see in the literature review that follows, most systems based on SWUs fix the number of units in a system, and constrain their search for units by doing so. The method presented in this thesis requires no such specification.

The type of acoustic modelling is a very important system choice, but is not implied in the way the problem is broken down into 3 steps, above. It is possible to generate the segmentation and unit inventory through any of various segmentation and clustering methods, and then use this segmentation to train acoustic models. This is seen in most of the systems reviewed in the following section. However, decoupling the acoustic modelling from the acoustic segmentation is unlikely to be as effective as an integrated method.

2.3.1 Literature review

In this section, various ASR systems using automatically derived units are described. Aspects of each system are compared in Table 2.1.

In the late 1980s, IBM researchers developed the concept of 'fenones' as sub-word units to investigate alternatives to phones. Fenones are frame-sized units labelled by a vector quantizer. These are introduced in Bahl, Brown, do Souza, Mercer & Picheny (1993) for isolated word, speaker-dependent recognition. An inventory of 200 prototype fenones was first generated by vector quantization (VQ) of 5 minutes of speech. Then for each word in the vocabulary, VQ vectors are extracted at regular intervals from one example of the word. The vectors are compared to the 200 in the inventory, and each is given the label of the closest (in terms of Euclidean distance) prototype fenone. This label sequence is the 'fenonic baseform' for the word: its pronunciation in terms of fenones. The fenones are then modelled by Markov models, and trained using 9 utterances of the same word. This method of sub-word unit generation and modelling achieved improved word error rates on a small vocabulary isolated word task, but not on a task with larger vocabulary. It is not surprising that pronunciations ("fenonic baseforms") based on a single utterance of a word will not perform well on a larger task. Improving the fenonic baseform using multiple training examples was then implemented: baseforms were created for a number of examples of each word, and then a single label sequence was chosen using maximum likelihood search across all baseforms given a model set. The Markov model set was trained using singleton fenonic baseforms from the first experiment. Using this process to generate baseforms improved system performance for all isolated word, speaker-dependent tasks attempted, compared to a phonetic baseline. A further experiment is reported using speech from multiple speakers, where again the word error rates generally improve on those using a phone-based system.

The system is extended in Bahl, Bellegarda, do Souza, Gopalakrishnan, Nahamoo & Picheny (1993) (this paper was published before Bahl, Brown, do Souza, Mercer & Picheny (1993), but was actually written after) to model pronunciation variation with more flexibility by modelling using more Markov states.

These methods by Bahl et al. do not require a lexicon generation step since each word is treated independently to determine the pronunciations. This is only possible in data where the location of word boundaries is known. There is no justification for the choice of 200 units in the system. These two pre-processing requirements (word boundaries and the number of SWUs sought) are typical of the methods that exist for generating SWUs.

Around the same time, Paliwal and Svendsen devised sub-word units, also for an isolated word single speaker task. In Svendsen et al. (1989) the acoustic segmentation is achieved using the maximum likelihood (ML) segmentation of Svendsen & Soong (1987). This method, whose purpose is to group frames with acoustic similarities together, is discussed in the following chapter, Section 3.1. The clustering of the ML acoustic segments is then carried out using 'segment quantization', which is analogous to vector quantization. The number of clusters is a pre-determined value, N. Each of these N clusters of segments is then modelled by an HMM, and as such, a unit inventory with corresponding model set is defined and trained. Three simple lexicon generation methods were then applied, the simplest being choosing a pronunciation at random for each word from the various training utterances of that word. Two further methods involving clustering of the different pronunciations for each word were also used. The final system achieved recognition scores comparable to whole-word HMM models, which were state-of-the-art at the time.

In Paliwal (1990), the lexicon generation is extended to a probabilistic form. The system uses the same sub-word unit derivation as Svendsen et al. (1989). For each word in the vocabulary, a statistical model of pronunciation is trained on all pronunciation variations of that word in the training data. The model is an ergodic HMM whose states are the sub-word units of the system. Recognition using this probabilistic lexicon improves on deterministic lexicons.

Again, we see that both Paliwal and Svendsen et al. require the number of SWUs to be predefined, and require data with word boundaries.

Maximum likelihood acoustic segmentation is also used in the SWU recognition systems of Fukada et al. (1996) and Bacchiani (1999). The emphasis of the Fukada et al. system is lexicon generation which incorporates phonetic knowledge to deal with unseen words. In order to do this, the inventory of SWUs is used to devise phoneme models, where a phoneme model is a sequence of SWUs. This is done by aligning the SWUs with a phoneme transcription, and for each phoneme merging the SWUs used to represent it in the data (merging the SWU means and variances), to give a sequence of merged SWUs to represent each phone. Presumably (this is not explicitly stated in the text, but is essential for the task carried out: a word recognition experiment) a dictionary of words in terms of phonemes is needed then to make the word models, which are defined as the concatenation of the phoneme models that make up the word's pronunciation. Thus, the acoustic segments are automatically defined, and individually modelled, and the lexicon jointly uses these and phoneme transcriptions. The recognition results on a speaker-dependent, spontaneous speech task show slight improvement on an HMM phoneme-based system. This result shows that the higher level of detail in the acoustic modelling (SWUs are shorter than phonemes, and in this system there are 120 SWUs, rather than typically 50-60 phonemes) can benefit recognition. The lexicon generation method requires an initial phone-based dictionary, and a good phonetic transcription of the data, so in fact the system is still dependent on phonetic knowledge and expensive initialisation (if the transcription is done by hand).

The recognition system of Bacchiani (1999) based on automatically derived units uses maximum likelihood acoustic segmentation, as above, to initialise the acoustic segmentation of the training data. Rather than clustering at this stage, pronunciation constraints are introduced, which alter the segment boundaries, and then clustering of the revised segments leads to the sub-word units. The pronunciation constraints enable the dictionary to be written directly: the segmentation of all tokens of a particular word will have the same number of segments (defined to be the median number of segments for that word following the unconstrained initial segmentation), and the same unit sequence. Forcing each token to have the same unit sequence allows the pronunciation of each word in terms of sub-word units to be immediately known: a separate lexicon generation step is not required. These constraints enable a pre-clustering of the segments, within tokens of a word, and further clustering must then be carried out across words so that sub-word units are re-used across words. This is done using a divisive method, with maximum likelihood as an objective function. The system showed recognition results which improved on phone systems for a low complexity system, and comparable results for higher complexity system on a RM task (which is read speech). Again, word boundaries are required to enable the system to be trained, and the number of SWUs pre-determined (the affect of this variable is investigated, but its value is not discovered as part of the algorithm). No initial lexicon is required, and no phone transcription.

Holter & Svendsen (1997) also uses ML segmentation to derive initial acoustic segments, which are then clustered into S 'codebook' clusters. The focus of the method here is the labelling of each acoustic segment (from the set of S clusters) in order to constrain the number of pronunciations for each word, and to use external linguistic knowledge. Two labelling schemes are proposed, the first to assign similar label-sequences to tokens of the same word, and the second to assign identical label-sequences to tokens of the same word. Dynamic time warping is used in the first scheme to search for a reference token among all examples of a particular word. The reference token is the "utterance that has the smallest average DTW-distance to all other utterances of that word", where utterances are represented as a series of acoustic centroids. The centroids of all tokens of the word are then aligned to the reference token (using DTW again), and all segments which end up aligned with the same segment in the reference token form a cluster. Each cluster is labelled according to the nearest codebook cluster (from the set of S constructed earlier). In this way all utterances of the same word receive similar label-sequences, but they do not have to be exactly identical. In the second labelling scheme, identical labelling for each utterance of a word is sought, by jointly searching for the segmentation and labelling such that a criterion is minimized. The same codebook of labels is used, and a search through a trellis of codewords is carried out, searching for the path with minimal distance for all utterances of each word. The recognition task these systems were tested on was simple: a multiple-speaker corpus containing 20 words, each spoken in isolation. Significant improvement in recognition rates are seen using both labelling schemes compared to an unsupervised labelling. Both labelling schemes perform comparably to systems with whole-word or phoneme-based models - all close to 100%. This paper demonstrates the care needed in constructing the pronunciations of each word, and shows that linguistic knowledge is not necessary to achieve segmentation. However, the method is tested on a small task of simple data, so its value is not known for continuous speech recognition. The requirements here are again the pre-definition of S, the unit set size, and word boundaries.

The sub-word unit derivation of Hersch (2003) takes a different approach to the 3 steps (segment, cluster, write a dictionary) that we have seen in work so far. The method used for this system instead begins with clustering of the data using k-means (it is not made clear what form 'the data' is in - frames of feature vectors, perhaps?) followed by acoustic modelling of the clusters. The trained acoustic models are then used to generate a segmentation of the data in terms of the clusters. This is unusual, to derive initial segments after acoustic modelling (although often segment boundaries are adjusted by trained acoustic models). The number of clusters is then adjusted by merging clusters if doing so raises the likelihood of the data. The generation of a dictionary is achieved by representing each word in the dataset by an HMM whose states emit sub-word units. The word-HMM models are trained on the sub-word unit transcription generated by the trained acoustic models of clusters. While this dictionary generation method results in a relatively flexible, probabilistic dictionary, it is restricted by the initial topology of the word-HMMs since the number of states per word-HMM has to be predefined. The experimental results for this system are very poor, around 40% WER on the OGI numbers task. This is possibly due to the initialisation of the sub-word units: just clustering the data frame-wise (if this is in fact what was done) without looking for segments according to some criteria is unlikely to result in sequences of frames in the same cluster, resulting in very short segments. Alternatively, the poor results could be due to the dictionary (word-HMM models) being too flexible for the task. The dictionary and units are not jointly determined: the SWUs are the product of one process with no regard for the subsequent constraint on the number of SWUs per word.

The approach taken by Singh et al. (2002) (also in Singh et al. (2000)) to derive SWUs and a dictionary automatically also does not require word boundaries or a specified unit set size. Instead, the system design is formulated as a maximum likelihood problem to jointly find the unit set (segments and acoustic models) and dictionary. Its formulation implies a huge search over all possible word segmentations, unit set sizes, acoustic model parameters, where the training data likelihood is being maximised. The initial equation is broken down into a set of equations which are solved using a divide-and-conquer strategy. The solutions of these equations are used in an iterative process to estimate models, derive a dictionary in terms of models, and increase unit set size. The experiment reported on a read speech task (RM) shows the error rates decreasing as the number of units increase, when tested on training data. Testing on unseen data, the best error rate is seen using 34 sub-word units. Using both context independent models and context dependent models, the automatic system could not outperform a phone-based system. However, the work shows that it is possible to define a unit inventory and a dictionary with little external data, since no linguistic information is necessary, including no word boundaries.

2.3.2 Discussion

In this section, comparisons between the approaches reviewed above are made, including comparing aspects of the systems to the approach of this thesis. In this way, the approach of this thesis is introduced in the context of research which precedes it.

Fenones (Bahl et al.), among the earliest automatically derived SWUs, are a simple, data driven, unit inventory and have been seen to improve word recognition. As described above, prototype fenones are output by a vector quantizer, and the training data is then segmented by each frame's distance from these prototypes. Fenones are in the same class when they are closer to a particular prototype than to any other prototype, using some distance measure. Markov models of each fenone are trained subsequently for use in recognition. This is typical of SWU generation: segmentation by some means and then acoustic modelling of groups of the segments. This simple data driven method can achieve gains in recognition. Theoretically more improvement is possible if the sub-word unit derivation allows the statistical model to play a part in determining the units. This is the approach taken in this thesis: jointly, segment boundaries and the acoustic models for each unit are determined. Instead of segmentation and clustering providing the decision about which frames of speech to model using the same statistical model, the models themselves define the boundaries. Segments of speech are in the same class if they are modelled by the same acoustic model, instead of being modelled by the same acoustic model because they are in the same class as determined by a separate objective function.

It is possible that the work using fenones did not continue due to the challenges of generating a pronunciation dictionary, as is implied in Holter & Svendsen (1997). As discussed above, fenone generation requires word boundaries, and pronunciations are sequences of fenones from a single example of each word. This does not allow for pronunciation variation, and clearly relies heavily on the example of each word being representative of many occurances of that word. The work using fenones indicates that using sub-word units that are automatically derived can improve upon the modelling and recognition power of using phones. Without methods to search for a good prototype pronunciation, the process is limited, however. The work in this thesis looks at these problems and presents a method of dictionary generation.

The method presented in this thesis combines segmentation and clustering into a single step by using an ergodic HMM, followed by dictionary generation using joint multigrams. Each state of the ergodic HMM is designated to be a sub-word unit. The segmentation and clustering step jointly determines segment boundaries, clusters, and HMM model variables. The Bayesian information criterion is used as an objective function to determine the number of sub-word units in the system, and the complexity of the models (in terms of number of Gaussian mixture components per HMM state). The dictionary generation process is automatic, aligning word transcriptions with sub-word unit transcriptions, but not requiring word boundary information. The result of this process is a multiple pronunciation, probabilistic dictionary.

The sub-word unit generation process in this thesis differs from a standard process. Typically, acoustic models of the sub-word unit set are trained following some kind of segmentation of the data. For example, the fenone SWU (Bahl et al.) is the output of a vector quantizer. Fenones are in the same class when they are closer to each other than to fenones in another class, using some distance measure. Markov models of each fenone are trained subsequently for use in recognition. Fenones are a simple, data driven, unit inventory, and have been seen to improve word recognition. If the sub-word unit derivation allows the statistical model to play a part in determining the units, theoretically more improvement is possible.

The task of jointly determining a unit inventory and dictionary is complex, as we have seen. In order to achieve a match between the unit design and the dictionary, some iteration between the two tends to be used. Bacchiani iterates between data segmentation given unit inventory, and unit parameters given segmentation. Singh et al. iterates between fixing the dictionary and using it to find acoustic models, and then using the acoustic models to update the dictionary. A further layer of iteration is also required by Singh, in the search for the number of sub-word units, N. N is gradually increased, and the units and dictionaries retrained accordingly, while the increase in N increases the recognition rates of some held out test data. In the BIC-multigrams approach presented in this thesis, re-estimation of segment boundaries and model parameters is done using a trained dictionary in order to ensure that the units and dictionary are matched. This is the only iterative aspect to this process.

Jointly optimising a unit inventory and a dictionary for a particular data set is a highly unconstrained problem, with many parameters. Typically, to make the problem more tractable, a number of constraints are employed. In most of the systems reviewed above, with the exception of Singh et al. and Hersch, the search for sub-word units is constrained at the outset by pre-defining the number of sub-word units the system will have. The methods which require this figure to be predefined are focusing on different aspects of the overall unit inventory derivation, and need the simplification of this constraint in order to explore other system parameters, for example word pronunciations, or the clustering and modelling of segments. However, without allowing the number of units to be different, it is impossible to tell whether the final trained systems are optimal for any optimisation criterion. In Hersch, this constraint is not required; instead the size of the unit set is adjusted by merging clusters according to some criteria. However, the technique requires the number of units per word to be predefined, which is another way of simplifying the search. As noted above, in Singh et al. (2002) the size of the unit inventory is a variable in the system, and the criterion to be met in increasing the inventory size is the increase in recognition scores for some held out data. In the BIC-multigrams approach of this thesis, the number of units is not pre-defined, but is determined on the basis of the BIC value each model set size scores during training.

Another input required by many of the systems reviewed here to constrain the search for a unit inventory is word boundaries. The early systems (Bahl et al., Paliwal, Svendsen et al.) were developed on isolated word data, and hence any decisions about searching for word boundaries was completely avoided. Bacchiani, Fukada et al., Paliwal, and Holter & Svendsen all require word boundary locations as inputs to their methods, removing the need to search for possible word boundaries as part of their algorithms. This constraint is exploited by Paliwal in his deterministic dictionary generation process: pronunciations for each word can be collected by splitting the sub-word unit transcriptions by the word boundaries. Bacchiani requires word boundaries in order to introduce two pronunciation constraints, which limit another potentially large search space, for the number of units per word and each word's pronunciation. The methods presented in this thesis do not require this knowledge of word boundary locations.

The next chapter fully presents the method of joint unit and model design. Chapter 4 presents the dictionary generation method.

Author	Task	Acoustic Modelling	Requirements / Constraints
Bacchiani	read speech (RM) & spontaneous speech	HMM	word bounds, N
Bahl	isolated word	Markov model	word bounds, N
Fukada	isolated word	trajectory model	word bounds, N, phone transcription, phone lexicon
Hersch	spontaneous speech (OGI Numbers)	HMM	number of units per word
Holter	isolated word	HMM	word bounds, N
Paliwal	isolated word	HMM	word bounds, N
Singh	read speech (RM)	HMM	none necessary
Svendsen	isolated word	HMM	word bounds, N
Couper Kenney	spontaneous speech (OGI Numbers)	HMM	none necessary

Table 2.1: A comparison of the systems reviewed in this chapter. All systems require acoustic training data, word transcriptions, and a prescribed acoustic model topology. Requirements beyond these are listed in the final column, where N indicates that the number of units in the system is pre-determined.

Chapter 3

Segmentation and Clustering

As outlined in Section 2.3, the problem of designing a unit inventory can be broken down into the steps of (1) segmentation, (2) clustering, and (3) lexicon generation. This chapter is concerned with steps (1) and (2). A literature review of methods available for segmentation and clustering is below in Section 3.1, followed by details of two experiments investigating different methods of segmentation, in Sections 3.2 and 3.3.

3.1 Methods in the literature

Methods for the segmentation and clustering of speech are required for various tasks, as well as sub-word unit determination. Such tasks include:

- Speaker segmentation, a task requiring data containing portions of speech spoken by different speakers to be segmented into sections containing speech of one speaker only, and these sections labelled such that the speech of the same speaker has the same label. This labelling is achieved through clustering of the segments found. This task, sometimes called diarization, is used in automatically indexing data such as news broadcasts or meetings.
- Automatic segmentation into sections containing different background noise conditions, useful for more specific noise modelling.

- Speech segmentation for language recognition, attempting to group the speech of each language together from audio data containing multiple languages.
- Automatic discovery of phonetic boundary locations, as a first pass in a hand-transcription task to make the process faster and cheaper.

There are various methods used to achieve the segmentation and clustering for these different tasks. As we see in the review that follows, some methods use relatively separate methods for the two steps, often with an iterative process connecting them. Others use a more integrated approach, combining the goals of both segmentation and clustering into one procedure. Table 3.1 contains a summary of the methods of segmentation and clustering reviewed.

3.1.1 Segmentation

Broadly, the methods of segmentation considered here fall into two categories,

1. Acoustic measure based: some form of measure is defined relating to the acoustic parameterisation of the signal. This acoustic measure criterion defines where boundary locations should be placed.
2. Model based: a model type is chosen, and a search is carried out to find the best segmentation such that, for example, the likelihood of these models is maximised.

There are many ways to design criteria for segmenting speech based on the acoustics alone. Firstly, the speech must be parameterised into feature vectors. As we briefly saw in the previous chapter, there are many choices of feature vector types, and for each type, decisions about dimensionality and frame sizes must be made. Then a method must be chosen or devised to place boundaries between vectors such that a segmentation of the data is found. A large number of such algorithms are available.

The methods which can be classed as acoustic measure-based all rely on the generation of some function of the acoustic vectors (or their derivatives) which is then segmented according to its shape.

One way to define boundary locations is to look at a distance between adjacent frames. This distance as a function of time will exhibit peaks where there is a large distance, and at these points it is reasonable to place segment boundaries, such that there is not a lot of variation within a segment. The distance measure must be defined. In Sharma & Mammone (1996), a Euclidean distance is used during the search for boundaries. A form of this style of segmentation is also seen in Couvreur & Boite (1999), who apply a speech segmentation algorithm for a broadcast news speaker segmentation task. In their work, segmentation is based on acoustic similarity between neighbouring frames: for a candidate segment boundary, the frames to the right and to the left of the boundary are modelled by Gaussians, and the distance between the Gaussians computed. This is carried out across the data. Boundaries are then placed where the distance measure is at maxima.

Another way of describing this method of segmentation is to define distortion measure, also based on distance between frames. This is seen in Holter & Svendsen (1997), who achieve segmentation by a search for segment boundaries which minimize a (within segment) distortion measure. As with Sharma & Mammone (1996), the measure of distance is Euclidean.

An alternative to looking directly at the feature vectors, is to consider the shape of functions of the vectors. For example, in Adami & Hermansky (2003), the trajectories of f0 and energy are computed, and segment breaks are placed where the trajectory exhibits peaks and troughs (where rates of change are zero). Segment breaks are also inserted at starts and ends of voicing. In this particular method, the segments can be labelled directly from 10 possible classes, each defined by a particular trajectory shape. Typically this is not possible in methods of segmentation: clustering of some sort has to be performed. Also for most tasks which require speech segmentation, 10 classes are not sufficient to capture information relevant to the task.

Another function that can be used in segmentation is the spectral change. One of the methods proposed in Svendsen & Soong (1987) is to place boundaries at peaks in the function of first order cepstral coefficients. These peaks relate to positions of rapid spectral change, and boundaries at these places should lead to stationary segments.

Svendsen & Soong (1987) also introduce a maximum likelihood (ML) objective function for finding segment boundaries where a specific distortion measure is minimised such that the speech within a segment is quasi-stationary. A dynamic programming (DP) algorithm is used to carry out the search for these boundaries. This is an acoustic metric based method. This method is used in the segmentation step of Paliwal's work (see Paliwal 1990), and also in Fukada et al. (1996). Bacchiani also uses a variation of this method in Bacchiani (1999), changing the objective function from the distortion metric, to a Gaussian model likelihood. In this way, Bacchiani converts the algorithm to a model-based approach to segmentation.

Model-based segmentation methods have been used for some tasks also. By model-based, I mean that the statistical models that will be used in acoustic modelling of segments are involved in the search for segment boundaries. In Chen & Gopalakrishnan (1998*a*, 1998*b*) and Ben et al. (2004), the segments sought are assumed to be modelled by Gaussians. At an hypothesised segment boundary, Gaussians are trained for segments either side of the boundary, and the Bayesian Information Criterion is calculated for each Gaussian. A boundary is placed between two segments a and b if the change in BIC is negative. In this way, the likelihood of the models are used to inform the choice of boundary locations. These authors use this method for a speaker segmentation task. A model-based segmentation method involving BIC is the main process used in this thesis to derive sub-word units. The use of BIC is different to these papers, however. This is presented later in this chapter.

It is possible to use other methods for segmentation when the sequence of segments is known. For example, if we have a phonetic transcription which is not time aligned, we might use a form of HMM realignment to find and refine the phone boundaries. This is seen in Ljolje & Riley (1991) and Pellom & Hansen (1998). This method, implicit in HMM training algorithms used today, is often used in systems which have used an automatic segmentation technique, but only once initial segmentation and modelling have been carried out.

Other acoustic segmentation methods include spectral variation function (SVF), as seen in Petek et al. (1996), temporal decomposition of spectral vectors (Cernocky 1998), and vector quantization (Bahl, Brown, do Souza, Mercer & Picheny 1993). Descriptions of these methods are not relevant to this chapter.

3.1.2 Clustering

In general, the output of the segmentation step is a set of segments for the given data with no way of relating them to each other. What is needed is an inventory of units, where a unit represents of a number of segments which are similar in some way. The clustering should result in segments in one class being more similar to each other than to those of any other class.

A popular machine-learning approach to this problem is the k-means algorithm. k-means is a simple algorithm in which the input data is clustered into k (which is a pre-defined value) clusters. This is done using a distance metric to compare data points. More detail about k-means appears in Section 3.2, where it is employed as part of an experiment.

Paliwal (1990) used k-means to cluster his acoustic segments, by first finding the centroid of each segment. Bacchiani (1999) and Fukada et al. (1996) use k-means to cluster segments by first representing each cluster by the parameters of its Gaussian distribution.

It is possible to classify many clustering algorithms as 'bottom-up' or 'top-down', where the set of clusters is decreased by combining clusters, or increased by dividing clusters, respectively. Couvreur & Boite (1999) use an agglomerative (bottom-up) clustering procedure for the speaker segments, combining the contents of the two nearest clusters. The number of speakers, N, is assumed to be known in their task, and so clustering is stopped once N is reached. Bottom-up clustering is also seen in Hersch (2003), where clusters are merged if the overall likelihood is increased by that action. Ben et al. (2004) and Chen & Gopalakrishnan (1998*a*, 1998*b*) use BIC (again - the segmentation method also involved BIC in these methods) for bottom-up clustering - each cluster (initially each cluster contains a single segment) is modelled by a Gaussian, and the two nearest clusters are combined if the combination causes the overall change in BIC score to be negative.

Sharma & Mammone (1996) report a different approach, in which initially a search for the optimal number of segments for the given data is carried out, followed by a search for segment boundaries for this number of segments. The optimal number of segments was obtained through a search between a calculated minimum and maximum number of segments: for each possible number of segments a 'cluster optimality criterion' score is calculated. The optimal number of segments is defined to be where this value is highest. This optimality criterion requires segment boundaries to be found, which is done via a dynamic programming search over the acoustic parameter space using a metric involving the distance between frames to determine the best boundary locations. These segments are then modelled by a Normal distribution before the optimality criterion can be calculated. This approach of finding the number of segments first and then the boundaries is unusual. It was only applied to isolated word recognition.

Vector quantization is used in Cernocky (1998) and Holter & Svendsen (1997).

Paper	Segmentation		Clustering	Task / notes
Adami et al. (2003)	inflection of F0 + energy trajectories	A	directly from shape of trajectory	speaker and language recognition
Bahl et al. (1993)	vector quantization	A	vector quantization	word models with non-phonetic subword units
Ben et al. (2004)	model-based, using BIC	M	agglomerative, using BIC	speaker diarization
Cernocky (1998)	temporal decomposition of spectral vectors	A	vector quantization	low bit-rate coding of speech
Chen et al. (1998*b*)	model-based, using BIC	M	agglomerative, using BIC	speaker diarization
Couvreur & Boite (1999)	acoustic distance between frames	A	agglomerative	speaker diarization
Holter & Svendsen (1997)	minimize acoustic distortion measure	A	vector quantization	automatic unit derivation
Ljolje & Riley (1991)	HMM realignment	M	n/a	phonetic labelling
Pellom & Hansen (1998)	HMM realignment as Ljolje (with adaptation for noise)	M	n/a	phonetic labelling (in noise)
Petek et al. (1996)	spectral variation function	A	n/a	segmentation of spontaneous speech
Sharma et al. (1996)	acoustic distance between frames	A	n/a	speech segmentation
Svensen et al. (1987)	ML objective function, minimising a distortion measure	A	n/a	automatic speech segmentation

Table 3.1: A comparison of the methods used to achieve segmentation and clustering by various researchers. 'A' or 'M' under segmentation refer to broad segmentation category: acoustic measure and model-based, respectively

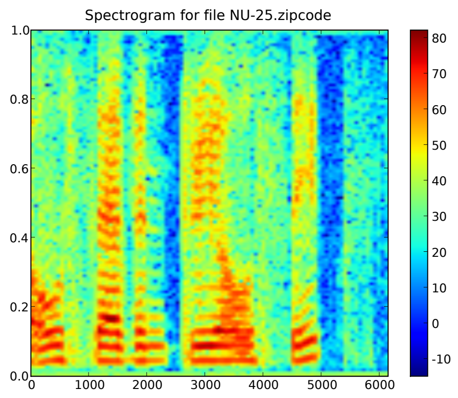


Figure 3.1: Spectrogram of utterance 25.zipcode, text: "oh seven three oh six"

3.2 Spectral Segmentation

In order to become better acquainted with the task of speech segmentation, some experiments are presented here involving a simple acoustic segmentation algorithm. Segmenting the speech just by looking at the acoustic signal is not a new method, as seen in the literature review in Section 3.1 above.

The aim in finding sub-word units in a speech signal is to place boundaries such that between boundaries the signal does not exhibit much variation, while across boundaries change occurs. The purpose of this thesis is not to look for phonetic boundaries or any other linguistically motivated unit, but to derive units from the acoustic signal.

One of the segmentation methods introduced by Svendsen & Soong (1987) was spectral segmentation. This idea is used here. The spectral features of a speech signal are found using Fourier Analysis. The spectrum is a representation of the signal in the frequency domain, and provide a characterisation of speech at a particular time point. A spectrogram is an image of spectra over time. At times when the vocal tract changes rapidly, the spectrogram image exhibits a change in pattern. The spectrogram in Figure 3.1 shows how clearly some of these boundaries can be seen.

A common parameterisation of speech is Mel Frequency Cepstral Coefficients (MFCCs). Usually in modelling applications, MFCC derivatives (deltas) are included, and often double derivatives also. As noted in Svendsen & Soong (1987), points of spectral change

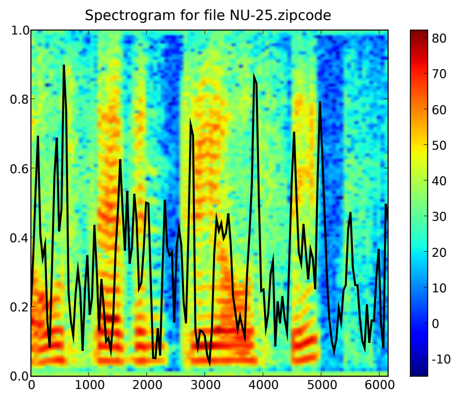


Figure 3.2: Example plot of the function of the sum of MFCC deltas for OGI numbers utterance NU-25.zipcode, text:"oh seven three oh six"

can be located where the the sum of 1st order derivatives as a function of time changes direction. Boundaries for sub-word units can then be placed wherever these changes in direction occur. The problem is thus reduced to peak picking.

An example plot of the function of the sum of MFCC deltas is in Figure 3.2, for a file from the OGI Numbers Corpus. As can be seen there, there are many local peaks in the function. It is useful to have ways of controlling the number of boundaries placed, or the minimum duration of a segment. Here this has been achieved using a threshold: boundaries are placed where the function crosses a particular value. Another way to control the number of boundaries is to smooth the function before peak-picking. However, this has not been investigated here. The graphs in Figure 3.3 show the effect of four threshold (t) values on the segmentation of one utterance: as the value of t increases, the length of segments increases.

Once an acoustic measure-based segmentation has been achieved, we have speech data broken into many small segments, but with no relationship between them. Some method of clustering the segments of speech found must be applied in order for any modelling to take place. A common form of clustering, as seen in the literature review above, is k-means clustering. The input to the k-means clustering algorithm is a value of k (the number of clusters to divide the data into), and a representation of each data point to be clustered.

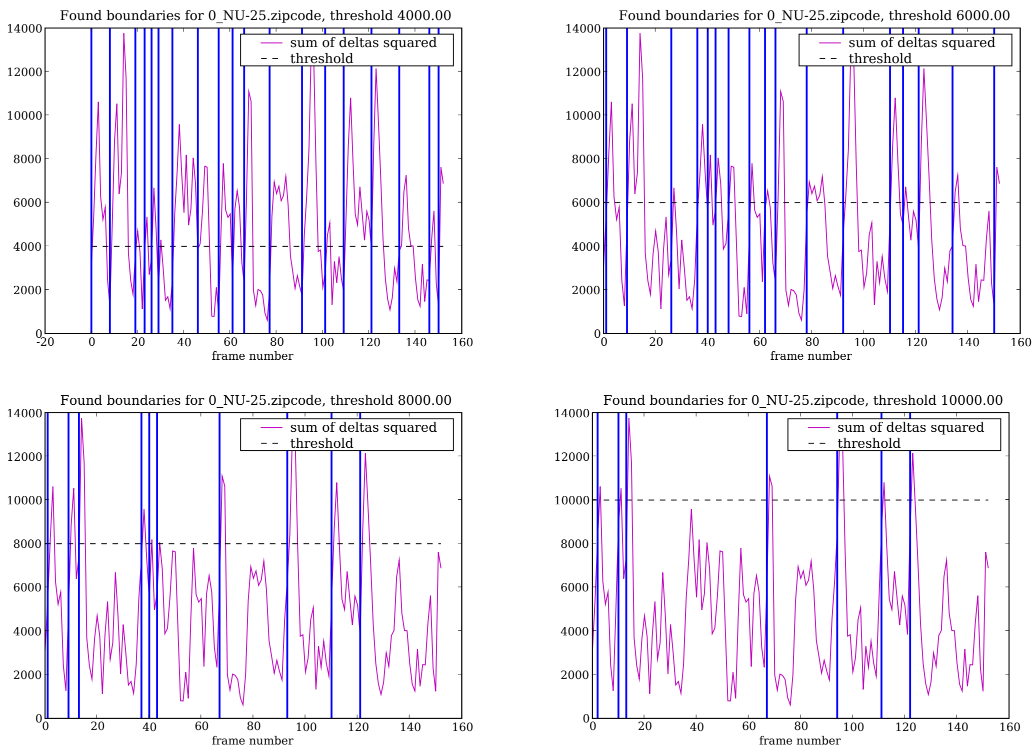


Figure 3.3: Boundary locations of segments found by spectral segmentation given different threshold values. Utterance text: “oh seven three oh six”.

The k-means algorithm, reproduced from Manning & Schutze (2001, chapter 14) is below:

```
1   Given: a set $\mathcal{X} : \vec{x}_1, \ldots, \vec{x}_n \subseteq \Re^m$
2           a distance measure $d : \Re^m x \Re^m \longrightarrow \Re$
3           a function for computing the mean $\mu : \mathcal{P}(\Re) \longrightarrow \Re^m$
4   Select $k$ initial centres $\vec{f}_1, ..., \vec{f}_k$
5   while stopping criterion is not true do
6           for all clusters $c_j$ do
7                   $c_j = \{\vec{x}_i | \forall \vec{f}_l \quad d(\vec{x}_i, \vec{f}_j) \leq d(\vec{x}_i, \vec{f}_i)\}$
8           end
9           for all means $\vec{f}_j$ do
10                  $\vec{f}_j = \mu(c_j)$
11          end
12  end
```

Clearly the value of k is not known a priori in this task, and so clustering with various values is carried out. The data to be clustered in this case are variable length speech segments, parameterised as MFCC vectors. The k-means algorithm requires a measure of the distance between each item, and clusters near items. We need to define a distance, then, between sequences of vectors, where the lengths of sequences may be different. Since we assume the spectral segmentation algorithm has found segments of speech with little within-segment variation, the simplest way of reducing the vector sequence into a single vector is by computing a mean MFCC vector. Then the distance measure used can be the Euclidean distance between mean vectors. So for a sequence of 39-dimensional vectors $\vec{v}_1, \vec{v}_2, \vec{v}_3 ... \vec{v}_z$, one mean MFCC vector $\vec{v}_R$ is created, where each coefficient i of $\vec{v}_R$ is the mean of coefficient i in the z original vectors.

There are other choices for how a sequence of vectors can be represented as a single point. For example, a trajectory of the MFCC vectors of each segment could be used, and normalised (stretched or compressed) to force each segment to have a trajectory of the same length. Alternatively each segment could be modelled by a Gaussian, in which case the Gaussian's mean and variance would represent the segment, and these two values used in k-means clustering. The Gaussian mean representing the segment is the same as the simple mean computed as above for each segment.

There are two parameters, then, in the segmentation and clustering as described here: the threshold, t, controlling the number of segments, and the number of clusters, k, which is the number of units. Spectral segmentation and k-means clustering using various values of t and k was carried out to segment and cluster the training data of the OGI Numbers Corpus. (Details about this corpus are below, in Section 3.5.3.) The graph in Figure 3.4 shows how parameters t and k affect the average number of units per word in this data set.

The outcome of this segmentation and clustering is a unit inventory (with k units, for various k-values), and a number of segments for each unit (dependent on the value of t). HMMs can be built for each unit inventory: here, for each unit, a 3-state, left-to-right HMM was trained on all the segments clustered as part of that unit. The graphs in Figure 3.5 show that some inventories yield a higher training likelihood, possibly indicating inventories which fit the data well. The inventory whose trained model set achieves the highest training likelihood has 60 units (k=60), and the segmentation was obtained with a threshold t = 6000.

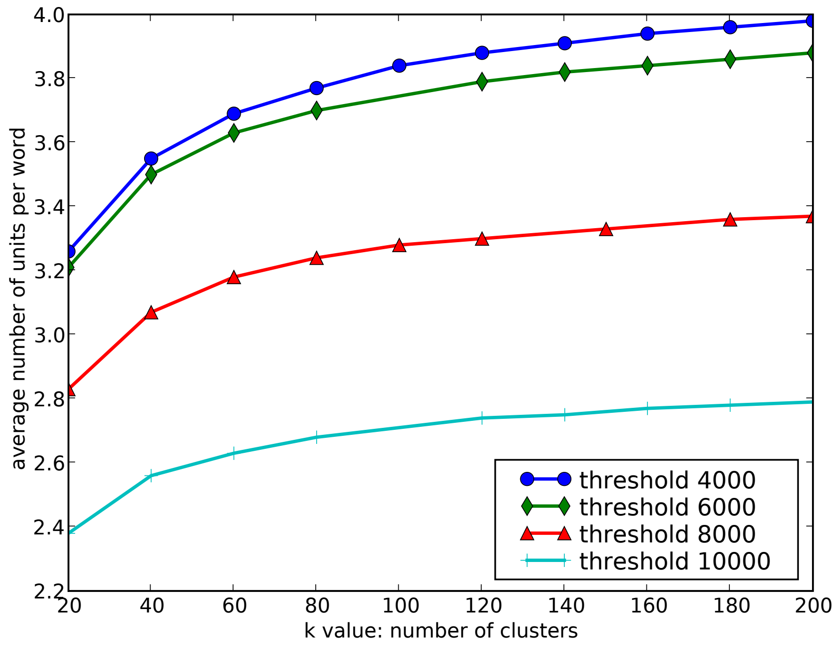


Figure 3.4: Average number of units per word found by spectral segmentation as parameters t (threshold) and k (number of clusters) change. Average calculated for each utterance and values normalised as: $\frac{1}{N}\sum_{i=1}^{N}\frac{|U_i|}{|W_i|}$ where N is the number of utterances, $|U_i|$ the number of units in utterance i, and $|W_i|$ the number of words in utterance i.

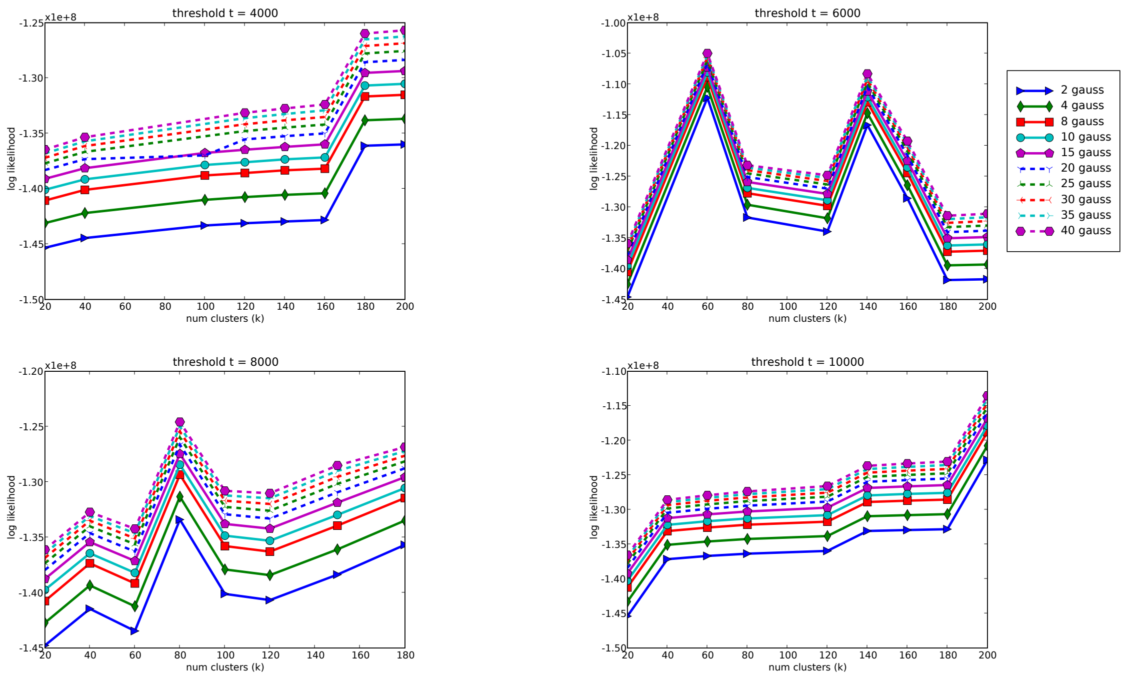


Figure 3.5: Plots of likelihoods for various values of t as the number of clusters increases, across different model topologies (number of Gaussian mixture components in legend)

3.3 Model-based segmentation and clustering

The acoustic segmentation carried out in the section above does not attempt to take into account the type of statistical model that will be used in the speech recognition system. The acoustic models representing each sub-word unit are of fundamental importance in a system, since poor modelling cannot lead to accurate recognition. Therefore it would be better to allow the use the modelling structure as an input to the segmentation algorithm as well as the acoustic data, in order that the units found

1. are modelled well by the given model type
2. occur frequently in the given data

So, simply put, the goal of the segmentation step is to find a segmentation of the acoustic data that best fits the type of statistical model used and the given data.

Having chosen a statistical model type, we want to train a set of these models, allowing each model to choose which pieces of data to model. The locations where transitions between models occur will be considered to be segment boundaries. Each of the trained models will be considered to represent a sub-word unit, automatically derived. This process enables segmentation and clustering to be achieved simultaneously. Figure 3.6 illustrates this basic model topology, for a generic statistical model. In order to use this method for generating sub-word units, we need (a) a way of training the connected set without a transcription, and (b) a way of determining the location of transitions between models.

There are training and decoding algorithms for the hidden Markov model (HMM) which meet these requirements ((a) and (b)), providing a way of training the connected model set, and a way of locating the transitions between models. If (as in Bacchiani (1999) and Fukada et al. (1996)) each sub-word unit is to be modelled by a single HMM state, the search for segment boundaries and training of sub-word unit models can be achieved by embedded training of an ergodic HMM[1]. This is made possible using existing training algorithms, and is the main technique used for segmentation and clustering in this thesis. In training an ergodic HMM, the number of states and initial parameter set

[1]in an ergodic HMM, all states can follow all other states, i.e. the transition matrix is full

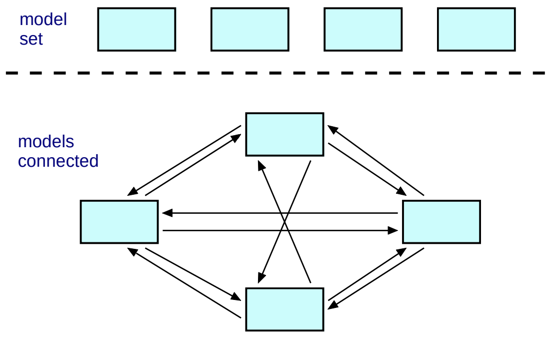


Figure 3.6: Illustrating allowing the models to determine segments and clusters: take a model set (a set of a particular type of statistical model), and connect the models in the set, such that each model can follow any other model. Train the large, connected model on acoustic data, and interpret each sub model as the model of an individual sub-word unit. See text.

must be specified. As we will see in Section 3.4 below, a criterion must be devised which enables the number of states used to be chosen in a motivated way, and in the experiments here, the Bayesian Information Criterion is used. First, though, we will look at the process for generating the sub-word unit set using an ergodic HMM in more detail.

3.3.1 Process

The process used to find the set of sub-word units is:

1. Take n HMM states, and connect them as an ergodic HMM.
2. Initialise the mean and variance of each state to be close to the data mean and variance, but distinct from all other states. The method used to achieve this is described fully in Section 3.5.3 below.
3. Train the ergodic HMM on all training data, providing the training algorithm with no transcription information, at either word or sub-word level.
4. The most likely state sequence of the ergodic HMM for each utterance is the sub-word unit transcription. The training of a single ergodic n state HMM leads to

a set of n automatically derived sub-word unit models and a transcription of the training data in terms of these. There is no need for a 'clustering' step. Standard decoding techniques are used to find the most likely state sequence; details are below.

5. The sub-word unit models are extracted from the ergodic HMM: each state representing one sub-word unit. Now we have a set of models and a transcription of the training data in terms of the model names. With a pronunciation dictionary in terms of these models, we would be able to carry out a standard recognition task, i.e. generate a word transcription for acoustic data. The generation of a dictionary is covered in Chapter 4

Embedded training of ergodic HMM

The outcome of training an HMM are the set of statistics

- the transition probabilities
- the mean and variance of each Gaussian
- the observation probabilities : the probability that observations were generated by the HMM

HMMs are typically trained using the forward-backward (or Baum-Welch) algorithm, which (iteratively) maximises the likelihood of observations given models. Details of the forward-backward algorithm can be found in Rabiner & Juang (1993, chapter 6) or Jurafsky & Martin (2000, Appendix D) or the HTK[2] manual.

This algorithm is used without alteration (as implemented by HTK) in the experiments of this thesis. Instead of a set of HMMs being trained, as in a standard speech recognizer, here just one (large) ergodic HMM is trained.

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

Decoding to find the most likely state sequence

Clearly a necessary outcome of any sub-word unit generation process is a transcription of the data in terms of the new unit inventory - without this, it is not possible to train a dictionary. For these experiments, where the units are determined using an ergodic HMM, the transcription of each utterance directly corresponds to the state sequence through the HMM for that utterance. However, since the state sequence is hidden, it is only possible to discover the likelihood of any state sequence for an observation sequence. In many decoding tasks, it is only necessary to find the most likely state sequence. This is true here; we do this using the Viterbi algorithm.

Details of this algorithm can be found in Rabiner & Juang (1993, page 339), Jurafsky & Martin (2000, sections 5.9 and 7.3) and the HTK manual.

3.4 Information criteria

As described above, the training of an n state ergodic HMM leads neatly to a set of n sub-word unit models. How, though, is n determined?

In training statistical models, the likelihood of a model over the training data will always increase as the number of parameters within the model increases. So an ergodic HMM with n_a states will have a higher likelihood than an ergodic HMM with n_b states if $n_a > n_b$. Maximising this likelihood, then, is not a useful criteria to enable a good choice of n. In general, as more parameters are used, models become more fitted to the particular training data. This is referred to as ‘overfitting’; the effect of overfitting is models that are unable to generalise to different data. Instead of maximising likelihood, a different kind of criterion must be found to avoid this and make models with more general use.

Information criteria provide a way to limit the complexity of models by penalising for large numbers of parameters. Usually the score given to a particular model by an information criterion is a function of the likelihood of that model minus a penalty term involving the number of parameters in the model. Thus using such a criterion as the objective function to be maximised, rather than likelihood, should result in a model able to generalise to new data. In our case, the score given to an ergodic HMM with n states can help us determine the optimal value for the number of sub-word units.

The Bayesian Information Criterion (BIC) has been used for various tasks in speech processing. One such task is speaker segmentation (or diarization), which involves splitting input speech into regions by speaker, and clustering these regions such that each speaker's speech segments are in the same cluster. (see Ben et al. 2004, Chen & Gopalakrishnan 1998*b*). As seen in the literature review above, the approach in these works is to segment and cluster separately, using BIC in each stage as the criterion for placing segment boundaries, and merging clusters, respectively.

The task here, of finding acoustic units in speech data, is analogous to speaker segmentation. The search is for regions of the same speech sound in this task, rather than regions of the same speaker in the diarization task. While there is analogy in the task, the use of BIC here is different to the uses of BIC for diarization in the literature, where two processes are employed. Here, segmentation and clustering are jointly performed by training an ergodic HMM, and BIC is used to choose the number of states in the ergodic HMM.

The size of the unit inventory is defined by the size of the ergodic HMM. Training the ergodic HMM provides the parameters for each unit model, and the segment boundaries for each occurrence of each unit. The use of BIC is not essential to derive units in this way, however some criterion is needed to determine the optimal number of units: ergodic HMMs with different numbers of states must be trained and one inventory chosen in some way. The test of a speech recognition system is in its ability to recognise words, quantified in the word error rate (WER). The best criteria, then, is the WER of each inventory; the inventory achieving the lowest WER is considered to be the best representation of the data. It is costly, in terms of time and computation, to determine the WER for each ergodic HMM, as a dictionary must be generated for each one, and recognition tests performed. If some criteria could be employed at an earlier stage of the process, this would be beneficial. It turns out that the ergodic model providing the inventory which achieves the lowest WER also achieves a high BIC score. This is seen in the experiments in chapter 5. This correlation was hypothesised, since information criteria enable a compact representation of the data to be determined, and such a model which does not over-fit the data should achieve better recognition rates than one which has a large number of parameters tuned to the specific training data. The correlation is useful since dictionary generation and recognition need only be carried out for models achieving high BIC scores. In this way, the training requriements of the full system are reduced.

3.5 Bayesian Information Criterion

3.5.1 BIC formulation

The Bayesian Information Criterion (BIC) was proposed by Schwarz (1978). BIC is defined as follows:

$$BIC(M) = \log L(X, M) - \lambda \tfrac{1}{2} \#M \log(N)$$

Where X are data to be modelled, M is a candidate model, N the data set size, and $\#M$ the number of parameters in M. The penalty weight, $\lambda = 1$. $L(X, M)$ is the joint likelihood of the data and the model.

3.5.2 Pilot Experiment

A pilot experiment is presented which demonstrates that BIC is able to determine the most appropriate model complexity, for artificial data.

Artificial data was generated by sampling from one hidden Markov model (HMM) with n states, generating data of dimension d. The output distributions of each state were specified such that the means were far apart. 1000 frames of data were generated.

The purpose of this pilot experiment is to show that maximisation of the BIC criterion can be used to discover the number of states in the generating HMM.

In order to do this, the artificial data generated was used as training data for a number of candidate ergodic HMMs. For each HMM, the BIC score was calculated.

The toolkit HTK was then used to train each model on the artificial data. BIC scores were calculated for each trained model using the joint likelihoods of the data and each

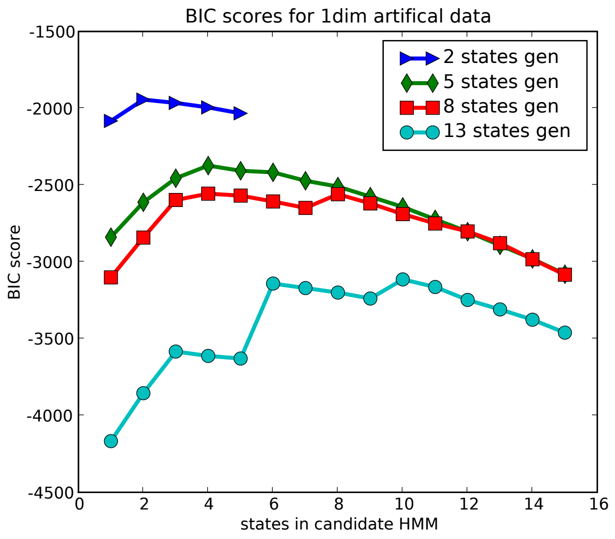


Figure 3.7: BIC scores for various candidate models trained on 1 dimensional artificial data. The number of states in each generating HMM is shown in the legend.

model at training. The model achieving the highest BIC score is considered to be the one that best represents the data. The graph in Figure 3.7 shows the BIC scores for various candidate HMMs for data generated with n=2,5,8,13 and d=1. Each curve increases to a maximum and decreases again, which is the expected behaviour of a BIC curve as the number of model parameters increases. The peak of each curve is close to the number of states in the respective generating model.

Table 3.2 shows the results of all pilot experiments, and demonstrates that, in these simple cases at least, BIC is a reliable way of determining the underlying model complexity.

3.5.3 Sub-word units for OGI numbers

An experiment was conducted using speech data from the OGI Numbers Corpus. Ergodic HMMs with various numbers of states were trained without any labelling information, on the training data. The BIC score of each model was calculated, and results are seen in Section 3.5.4 below.

In standard ASR systems, 3-state HMMs are used to model phonetic units. In order

data dimension, d	num of states, n, in generating model	num of states achieving highest BIC score
1	2	2
1	5	4
1	8	7
1	13	10
2	2	2
2	5	6
13	1	1

Table 3.2: Number of states chosen using BIC, for each of the generating models in the pilot experiment.

to generate SWUs which also have 3 states, the base topology of the erogdic HMM must be slightly different, with fewer transitions (for the same number of states), as shown in Figure 3.8. The same training methods are used for this model. For decoding, the model must be split into constituent 3state units, rather than single state units. Following this step, the same decoding procedure is used as for single state SWUs.

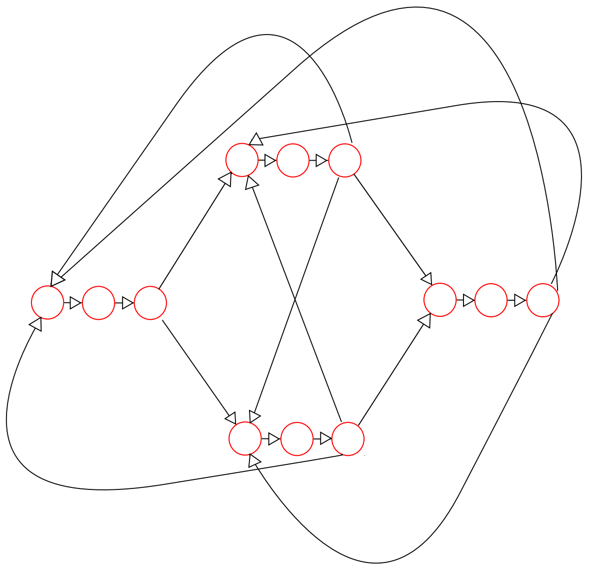


Figure 3.8: The HMM topology used to train 3state sub-word units.

Data

These experiments were conducted using the OGI Numbers Corpus, release 1.3[3]. The utterances in OGI Numbers were taken from other telephone speech data collections, and include isolated digit strings, continuous digit strings, and ordinal/cardinal numbers.

A division of the corpus suggested in Mariethoz & Bengio (2004) is used, where utterances containing truncated words are removed, and then only sentences containing the 30 most frequent words are retained. Then using the modulo-5 rule (3/5 training data, 1/5 validation, 1/5 test), the data is partitioned into 10441 sentences for training, 3582 for validation and 3621 for testing. Our training set was reduced to 10251 files, removing files containing corrupt data.

Initialisation

The HMM model structure used has n states, with g Gaussian mixture components per state. Each Gaussian has a 39-dimensional mean and a diagonal covariance matrix. The number of parameters in each ergodic HMM, $\#M$, is required as part of the BIC penalty term. It is calculated as follows.

$$\begin{aligned} \#M &= \#means + \#variances + \#transitions + \#mixture\ weights && (3.1) \\ &= 39ng + 39ng + n^2 + (ng - 1) && (3.2) \\ &= n(79g + n) - 1 && (3.3) \end{aligned}$$

The means of the states within each HMM must be different initially, to allow training to happen. If all states are initialised with the same mean, they all end up being identical after training, since they all attempt to model all of the training data in the same way. Moving all the dimensions of the mean of each state away from the overall data mean leads to the means of some of the states not moving at all, due to their distance from any data. Moving one or two dimensions of each mean allowed all states to train, so this was used. The distance moved was $\frac{1}{5}\sigma$ where σ is the overall variance of the data.

[3]http://www.cslu.ogi.edu/corpora/numbers/index.html

We need S unique 39 dimensional vectors close to the global mean $\vec{\mu}$. Six processes are used in turn to generate vectors until the total is reached:

1. Generate a new mean by adding $\frac{\sigma_i}{5}$ to *one coefficient*, i of $\vec{\mu}$. Repeat for all 39 coefficients of $\vec{\mu}$, generating 39 new mean vectors.
2. As above, but subtract $\frac{\sigma_i}{5}$ from one coefficient of $\vec{\mu}$
3. Generate a new mean by adding $\frac{\sigma_i}{5}$ to coefficient i, and $\frac{\sigma_j}{5}$ to coefficient j of $\vec{\mu}$ $(i < j)$
4. As (3), but subtract $\frac{\sigma_j}{5}$ from coefficient j
5. As (3), but reversing '+' and '-', i.e. subtracting $\frac{\sigma_i}{5}$ from coefficient i, and $\frac{\sigma_j}{5}$ from coefficient j
6. As (4), but reversing '+' and '-', i.e.subtracting $\frac{\sigma_i}{5}$ from coefficient i, and adding $\frac{\sigma_j}{5}$ to coefficient j

Other ways to initialise the means include (1) directly using S randomly selected data points and using these points as the means, or (2) using k-means to divide the data into S clusters and training an HMM state on each cluster. These have not been implemented here, however, since the perturbation of the means as described led to the mean of every state moving, i.e. finding data to train on. If the data dimension was larger, it would possibly be necessary to use an alternative initialisation.

3.5.4 Results

Single state sub-word units

The graph in Figure 3.9 shows the BIC scores for the models with single state units. The model achieving the maximum BIC score over the OGI Numbers data with an ergodic model made up of single HMM states has 150 states, and 30 Gaussian mixture components per state. This is interpreted to mean that in order to model this data set with single state HMM models for each sub-word unit, 150 such models should be used.

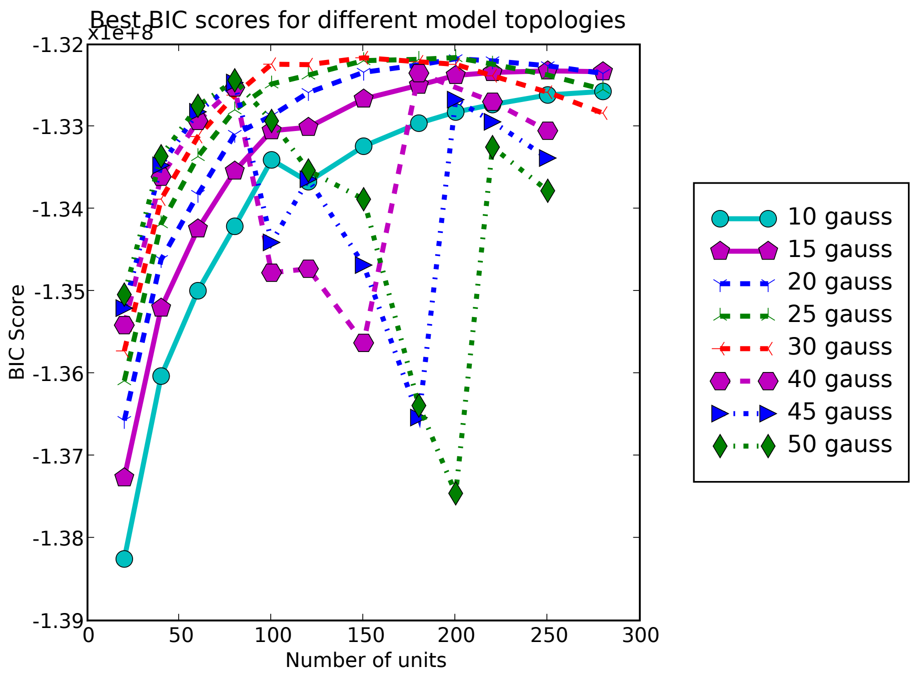


Figure 3.9: Best BIC score across unit inventory size (for single state units) for various numbers of mixture components per state ('gauss' in legend).

The noticeable dips in the graph, on curves representing 40, 45 and 50 components per state, in the range 100 - 200 units are unexpected. The models with lower complexity (lower numbers of mixture components per state) follow the neat pattern of gradual increase and decrease of score. The training likelihood of these higher complexity models is lower than for the lower complexity models with the same number of states, as seen in the graphs of Figure 3.10. This directly affects the BIC score. The reason for the decrease in training likelihood may be overtraining, or may be to do with the way mixtures are split, perhaps leading to some components which are not useful for training any data. In the training process all mixture weights are floored, so no mixtures vanish. No errors were noticed during training, nor was any training data ignored. Further analysis of these phenomena has not been carried out. The purpose of these trained models within this thesis is to derive SWUs, using BIC to control the model complexity. Since the focus is on models achieving high BIC scores, these models with notably lower scores are not further considered.

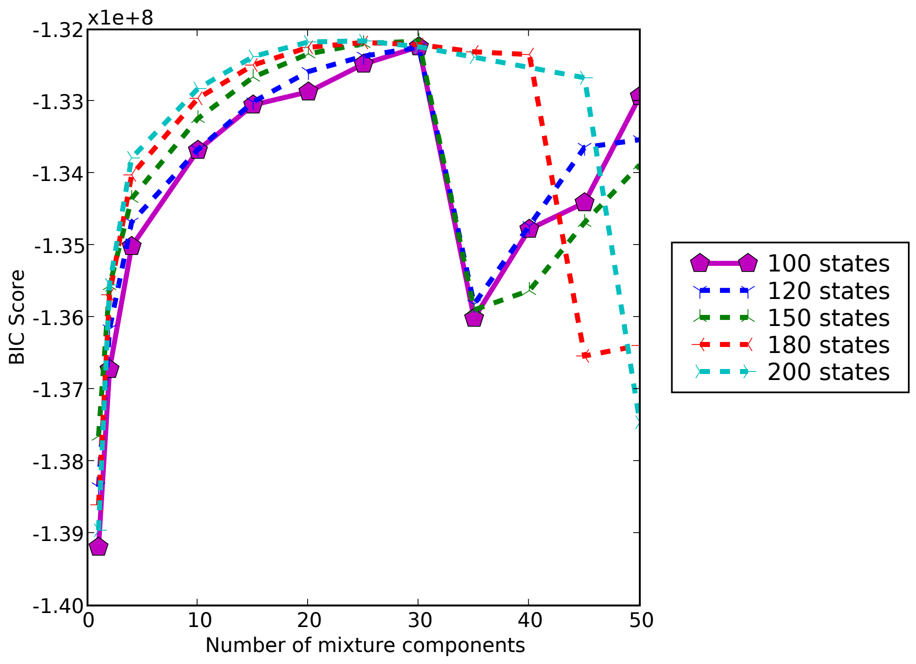


Figure 3.10: Training data likelihood for a subset of the ergodic models built for 1state SWUs, showing dips in likelihood for larger numbers of mixture components.

Three-state sub-word units

The various curves of BIC scores in the graph in Figure 3.11 exhibit the expected shape of gradually increasing and decreasing, each with a peak. Clearly the model with the highest BIC score in this set of models has 200 units (600 states), and 20 Gaussian mixture components per state.

Comparing units

The effect of the different model types on the number of units used to model each word can be seen in the plots of Figure 3.12. Modelling the data with single state HMMs leads to a system that has between 10 and 20 units per word. Modelling the data with 3state HMMs instead leads to between 4 and 9 sub-word units per word. For both model types, the number of mixture components appears to have little effect on the average number of units per word. However, and again this effect is seen for both model types, as the number of units in the system is increased, so the number of units per

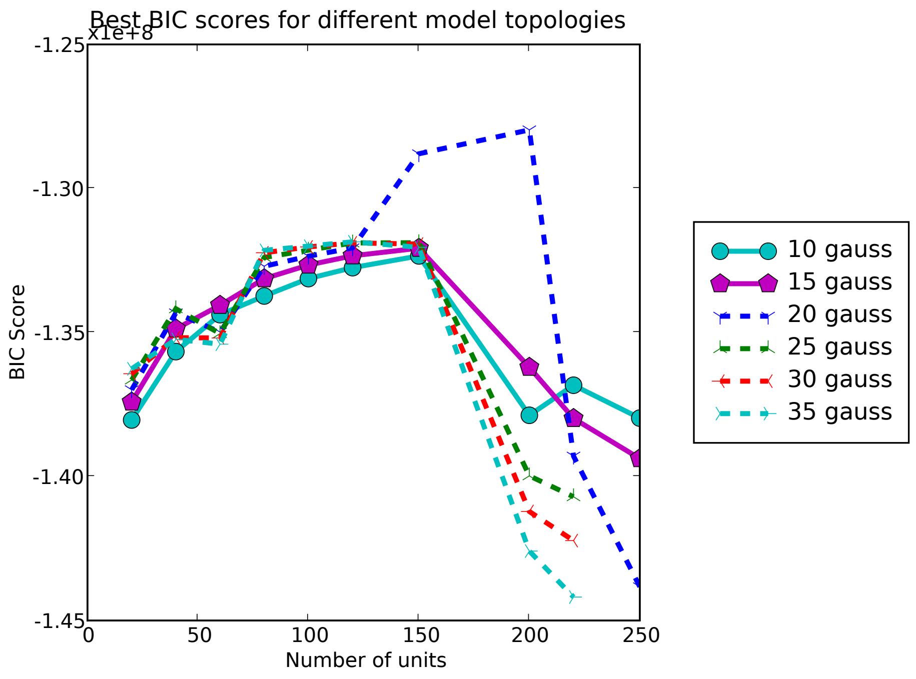


Figure 3.11: Best BIC score across unit inventory size (number of 3state units) for various numbers of mixture components per state ('gauss' in legend).

word increases.

The histograms in Figures 3.13 and 3.14 show the frequency of units across the training data for the models achieving the highest BIC scores for each model type. It can be seen that, for both the 1state and the 3state units, there is a range of frequencies, but that each unit is used to model some of the data.

3.6 Summary

This chapter presented two methods for automatic speech segmentation and clustering to determine a set of sub-word units. The best test of how useful these unit are for speech recognition is by comparison of word error rates. This is not possible directly: the recognition system needs a way of relating the units to words. This is covered in the next chapter: dictionary generation.

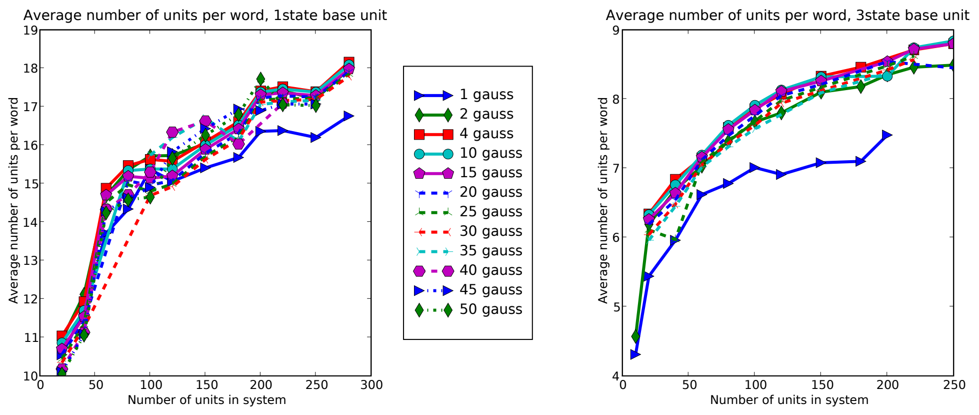


Figure 3.12: Average number of BIC units per word, as number of states increases. Average is total number of BIC units in training data divided by total number of words.

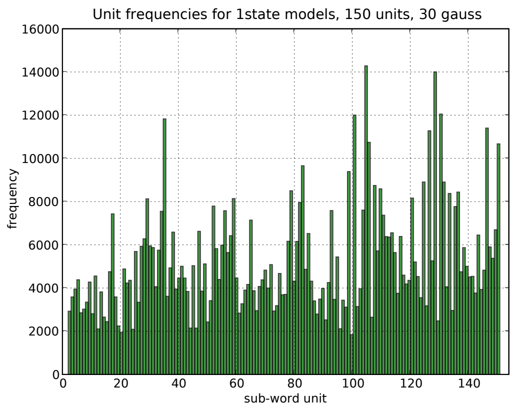


Figure 3.13: Histogram of the frequency of each sub-word unit in training data for the unit inventory achieving the highest BIC score for single state units (150 units, 30 Gaussian components per state)

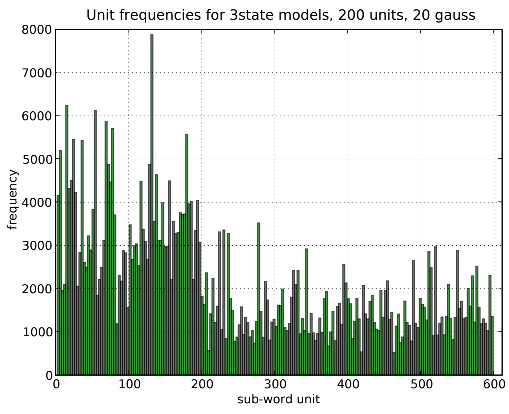


Figure 3.14: Histogram of the frequency of each sub-word unit in training data for the unit inventory achieving the highest BIC score for 3state units (200 units, 20 Gaussian components per state). (The x-axis is labelled to 600 due to the fact that there are 3 states per unit, and units were labelled with every third integer)

Chapter 4

Dictionary generation

In any automatic speech recognition system, a component part is a lexicon describing the relationship of orthographic words to the system's underlying sub-word unit. When the sub-word units used are phonemes or syllables, often an 'off-the-shelf' dictionary can be used since the relationship between words and these linguistic units is understood. However, when the sub-word unit is automatically derived, the dictionary must also be derived in some way. In order for the process of unit derivation to be fully automatic, the dictionary must also be automatically generated from the data. The input data that is available for use in dictionary generation is two strings of labels: word transcriptions, and sub-word unit transcriptions. There is no timing information available, so simply aligning the unit sequences according to word boundary locations is not possible. The fact that timing information is not required means that data preparation prior to using this automatic process is relatively inexpensive: the word sequences can be transcribed in almost real time by a transcriber with good typing skills. However if transcriptions are required to also have word boundary information, the time taken to achieve the transcriptions increases substantially.

This chapter addresses this problem of automatic dictionary generation, finding a solution in joint multigrams. The chapter is organised as follows: Section 4.1 presents methods reported in the literature for dictionary generation and pronunciation variation modelling. In Section 4.2, an intuitive, supervised method is described, the output of which is used as a baseline for comparison to the automatic method using multigrams, introduced in Section 4.3. Following these theoretical sections, a recognition experiment using phoneme-based sub-word units is reported in Section 4.4, in which dictionaries generated by both methods are used as part of speech recognition systems. The word error rates reported following this phoneme-based experiment demonstrate that the automatic method is able to generate dictionaries that are useful for speech recognition. In fact, the results show that dictionaries generated in this automatic way outperform those generated in a supervised way. Following this positive result, it is with confidence that the automatic algorithm can be used as part of a sub-word unit based system as will be seen Chapter 5.

4.1 Methods in the literature

The section on pronunciation variation modelling (section 2.2.1) reviewed a number of ways of generating or amending dictionaries automatically to reflect variants in the data. The purpose of these methods is to extend or amend a phone-based dictionary by analysing data. Many of the methods require an initial dictionary, used to generate initial acoustic models and / or phone transcriptions, before variants are found, e.g. ten Bosch & Cremelie (2002), Yang et al. (2002), Hain (2002), Fukada et al. (1998). In Wester et al. (1998) the initial lexicon was generated automatically using a text-to-speech system. In Hain (2002), the quality of the initial dictionary was found to be important. Most of the methods require word boundaries so that variants can be collected. Word boundary information can be inferred from phonetic transcriptions in conjunction with the dictionary if time aligned transcriptions do not exist. Similarly, Paliwal (1990) relies on exact word boundaries, and Fukada et al. (1996) requires a phone lexicon and transcription (see Section 2.3.1).

The requirement of one or both of an initial dictionary and word boundaries means that the dictionary generation methods in the pronunciation variation literature cannot be applied here.

4.2 A semi-automatic algorithm for dictionary generation

In order to get a feel for the difficulty of the task of automatic dictionary learning, and to get to know the data, an intuitive supervised method was used to generate a dictionary reflecting the data. The output of this algorithm enables a comparison with any dictionaries produced fully automatically.

A simple algorithm is introduced, described in pseudo-code Figure 4.1, to be used to try to capture all the pronunciation variation existent in the data. Each utterance has a word and phone transcription. For each utterance, the word transcription is expanded word-by-word using the dictionary; these pronunciation options from the dictionary are aligned one at a time with the phone transcription, searching for a pronunciation which exists exactly in the phone transcription. Silences and noise are ignored. If this cannot be done, the utterance is flagged, and a new pronunciation is manually added to the dictionary for whichever word could not be matched in this way. This process is continued until there are no utterances which cannot be described by the dictionary.

This is a greedy algorithm, which means that all possible phonemes are assigned to words 'earlier' in the transcription of each utterance. If two consecutive words share a phoneme, it will be associated with the first word. For example, the phoneme 's' is shared in the word sequence 'six six'. The phoneme stream[1] 's I kc kh s I kc kh s' would result in two different pronunciations of 'six', 's I kc kh s' and 'I kc kh s' with the segmentation 's I kc kh s — I kc kh s'. This results in some short pronunciations, which are the 'leftover' phonemes for the second word.

Section 4.4.2 contains details of the dictionary produced using this algorithm for OGI Numbers data.

4.3 Multigrams, Joint Multigrams

Multigrams are probabilistic models which provide a way of determining repeated subsequences in a string of symbols. The multigram model was introduced in 1995 in

[1]OGI Numbers is phonetically labelled using Worldbet, which is an ASCII encoding of the International Phonetic Alphabet, see Hieronymous (1993)

```
Inputs:
        - lexicon containing at least one pronunciation for each word
        - speech utterances, where each utterance has a word transcription
          and a phone transcription.

1. for each utterance, s:
       for each word, d, in the word transcription for s:
           lex(d) = the set of pronunciations of d in the lexicon
            (ordered by pronunciation length, with the longest first)
           for each pronunciation, p, of lex(d):
               if p exactly matches the first n phones of the phone transcription
                (where n is the length of p):
                   break
                   increment count for pronunciation p
           if not pronunciation in lex(d) matches phone transcription:
               skip any remaining words and flag utterance

2. for all flagged utterances:
       manually add pronunciations to lexicon

3. Rerun steps 1 and 2 until there are no flagged utterances,
   i.e. all pronunciations are in the lexicon.
```

Figure 4.1: Pseudo code for the semi-automatic algorithm used for dictionary generation

two papers, Bimbot et al. (1995), Deligne & Bimbot (1995) to model sequences of variable length within symbolic data. An input stream of symbols is broken down into parts by multigrams to express dependencies and repetitions existing within the stream. Under the multigram framework, the input stream is seen to be composed of a series of multigrams, each multigram being a sub-sequence of the input. In this way, redundancy is removed, and patterns are seen easily: the stream `ababab` would be more simply expressed $z_1 z_1 z_1$ where z_1 is the multigram [`ab`]. The stream `abadeab` reduces to the multigram sequence $z_1 z_2 z_1$ where z_2 is [`ade`]. Thus, multigrams compress data by encoding it using fewer bits than the original string required.

A formal definition of multigrams is given by Deligne & Bimbot (1995):

> Let $D = s_1; \cdots ; s_m$ denote a dictionary that contains all the sequences which can be formed by combinations of 1, 2, $\cdots$ up to n symbols of the language vocabulary. A n-multigram model is fully defined by a set of parameters Θ consisting of the probability of each sequence $s_i \in D$:
>
> $$\Theta = (\theta_i)_{i=1}^{m} \tag{4.1}$$
>
> where $\theta_i = p(s_i)$ and $\sum_{i=1}^{m} \theta_i = 1$

A set of multigrams $\{z_i\}$ is derived by jointly maximising the likelihood of the data and of the set $\{z_i\}$:

$$\{z_i\} = \arg\max_{\{z_i\}} \mathcal{L}(O|\{z_i\})\mathcal{L}(\{z_i\}) \tag{4.2}$$

For example, given a sequence
$\mathcal{A} = 1234345634$,
in order to formulate multigrams, we have to find all possible sequences of $\mathcal{A}$ and collect them in a dictionary D. If we only allow sequences of maximal length 3, our dictionary contains the following:
$D = \{1, 2, 3, 4, 5, 6, 12, 23, 34, 45, 56, 63, 123, 234, 343, 434, 345, 456, 563, 634\}$
These sub-sequences of digits gain an initial probability by counting their occurrences. These counts are normalised so the sum of all probabilities is equal to 1, and then we have the basis of a set of multigram models.

The fact that multigrams model variable length sequences means that multigrams are flexible, and able to expose dependencies of variable length in the data. This flexibility contrasts with the n-gram model where dependencies between symbols are only seen if they are within n symbols of each other, and n must be externally specified.

Multigrams were used by de Marcken (1996) to segment a phoneme stream as part of unsupervised language acquisition research, to demonstrate that segmentation of language, as part of language acquisition, is possible simply through enough exposure to the language. The multigrams he trained on the phoneme stream output by a trained HMM-based phoneme recogniser related very well to word boundaries, i.e. to the segmentation an adult speaker would naturally know. Cernocky (1998), investigating automatically derived units, used multigrams to find frequently occurring sequences of his units. He then trained acoustic models of these multigram-units instead of the original units. The effect of this was a smaller number of units to model, with many of the units modelling longer sequences of data than those originally found.

Joint multigrams were introduced in Deligne et al. (1995) as an extension to the multigrams framework. Joint multigrams have all the properties of multigrams, yet now modelling *co-sequences*, variable length sub-sequences of *two* input streams. Joint multigrams provide a data driven way of aligning and segmenting two inputs such that the inputs can be expressed as a series of joint multigram models. The length of sequences from each stream is variable (a pairing of a sequence, length n, from one input stream with a sequence, length m, from the other), and joint multigrams are models of repeated co-sequences across both streams.

Joint multigrams have been used by Bisani & Ney (2002, 2003) for grapheme to phoneme conversion. A dictionary of phoneme n-grams to grapheme m-grams is created using joint multigrams to find the relationships between the phoneme and grapheme sequences in the data. This dictionary is applied to the task of recognition of new words (words unseen in training data) in automatic speech recognition.

This way of modelling two streams is used in this thesis for dictionary generation. The two input streams, the words and the sub-word units, are available, and no initial dictionary exists. So joint multigrams are built on the streams, resulting in a probabilistic dictionary.

4.3.1 Example illustrating joint multigrams

Before the details of the joint multigrams model are explained, a simple example of how joint multigrams can be used is presented in this section in order to give the reader a picture of how they are going to be used. In this example, and in the experiments that follow, the input stream of words is always segmented into individual words; sequences of words are not used in the joint multigrams here. (This is not necessarily a characteristic of joint multigrams, which are able to be more general. It is instead due to implementation constraints, as seen in Section 4.3.3 below.)

To illustrate the working of joint multigrams, consider two input streams, $\mathcal{W}$ and $\mathcal{U}$,
$\mathcal{W}$ = one seven one
$\mathcal{U}$ = w ^ n s E v ^ n w ^ n

Given that there are 3 words, we want to find all segmentations of $\mathcal{U}$ (11 units) into 3 partitions. The number of such segmentations is

$$\frac{(u-1)!}{(w-1)!(u-w)!} = \frac{10!}{2!.8!} = 45$$

Section 4.3.3 below explains how this calculation arises. These 45 segmentations are shown in Table 4.1. These joint segmentations give rise to 58 distinct joint multigrams (pairings of a word with a sequence of units), 17 involving the word “one”, and 41 involving “seven”. This implies that there are 17 possible pronunciations for “one”, and 41 for “seven” in this input data, since our interpretation of a joint multigram is a word and a pronunciation.

one	seven	one	one	seven	one
w	ˆ	n s E v ˆ n w ˆ n			
w	ˆ n	s E v ˆ n w ˆ n	w ˆ n s	E	v ˆ n w ˆ n
w	ˆ n s	E v ˆ n w ˆ n	w ˆ n s	E v	ˆ n w ˆ n
w	ˆ n s E	v ˆ n w ˆ n	w ˆ n s	E v ˆ	n w ˆ n
w	ˆ n s E v	ˆ n w ˆ n	w ˆ n s	E v ˆ n	w ˆ n
w	ˆ n s E v ˆ	n w ˆ n	w ˆ n s	E v ˆ n w	ˆ n
w	ˆ n s E v ˆ n	w ˆ n	w ˆ n s	E v ˆ n w ˆ	n
w	ˆ n s E v ˆ n w	ˆ n	w ˆ n s E	v	ˆ n w ˆ n
w	ˆ n s E v ˆ n w ˆ	n	w ˆ n s E	v ˆ	n w ˆ n
w ˆ	n	s E v ˆ n w ˆ n	w ˆ n s E	v ˆ n	w ˆ n
w ˆ	n s	E v ˆ n w ˆ n	w ˆ n s E	v ˆ n w	ˆ n
w ˆ	n s E	v ˆ n w ˆ n	w ˆ n s E	v ˆ n w ˆ	n
w ˆ	n s E v	ˆ n w ˆ n	w ˆ n s E v	ˆ	n w ˆ n
w ˆ	n s E v ˆ	n w ˆ n	w ˆ n s E v	ˆ n	w ˆ n
w ˆ	n s E v ˆ n	w ˆ n	w ˆ n s E v	ˆ n w	ˆ n
w ˆ	n s E v ˆ n w	ˆ n	w ˆ n s E v	ˆ n w ˆ	n
w ˆ	n s E v ˆ n w ˆ	n	w ˆ n s E v ˆ	n	w ˆ n
w ˆ n	s	E v ˆ n w ˆ n	w ˆ n s E v ˆ	n w	ˆ n
w ˆ n	s E	v ˆ n w ˆ n	w ˆ n s E v ˆ	n w ˆ	n
w ˆ n	s E v	ˆ n w ˆ n	w ˆ n s E v ˆ n	w	ˆ n
w ˆ n	s E v ˆ	n w ˆ n	w ˆ n s E v ˆ n	w ˆ	n
w ˆ n	s E v ˆ n	w ˆ n	w ˆ n s E v ˆ n w	ˆ	n
w ˆ n	s E v ˆ n w	ˆ n			

Table 4.1: Joint multigrams example: 45 possible joint segmentations of input ‘one seven one’, which consist of 58 unique joint multigrams.

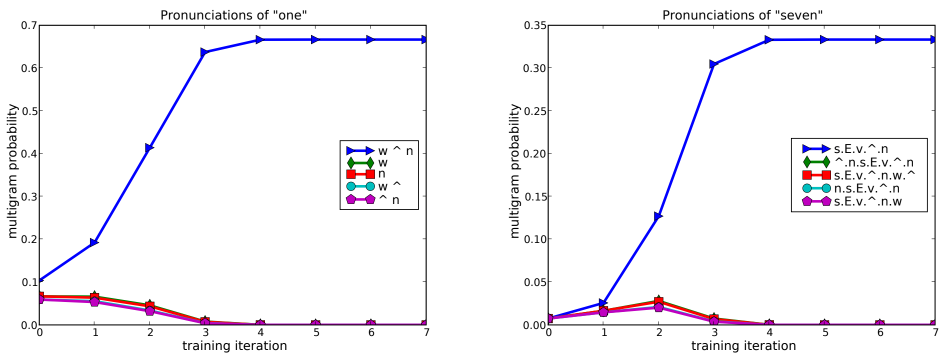


Figure 4.2: Joint multigrams example: Probabilities of the five most probable pronunciations for “one” and “seven”.

one	0.667	w ˆ n
seven	0.333	s E v ˆ n

Table 4.2: Joint multigrams example: final dictionary with probabilities

These 58 multigrams are given an initial probability by normalising the count of the occurrences of each. These initial probabilities were updated over 7 iterations of training using Equation 4.3, until the probabilities converge. The graphs in Figure 4.2 show the probabilities of the five most probable multigrams for each word (“one” and “seven”), as they change over training. The correct pronunciation is quickly and clearly distinguished from the others.

The final dictionary, with probabilities, is in Table 4.2. The segmentation receiving the highest probability, using multigram probabilities, after training is:

one	seven	one
w ˆ n	s E v ˆ n	w ˆ n

4.3.2 Formulation of joint multigrams

Joint multigrams are formulated as follows: Given two input streams of symbols, $\mathcal{W}$ and $\mathcal{U}$, a joint multigram model is fully defined by the set of co-sequence probabilities $\{p(\mu_i, \nu_j)\}_{i,j}$ such that $\sum_{i,j} p(\mu_i, \nu_j) = 1$, where (μ_i, ν_j) is a co-sequence, μ_i is a sequence from $\mathcal{W}$ and ν_j from $\mathcal{U}$. Streams $\mathcal{W}$ and $\mathcal{U}$ are assumed to be formed by the concatenation of two independent co-sequences (μ_i, ν_j). The dictionary D in the joint multigrams model is of $p(\mu_i, \nu_j)$ for all i, j. The general joint multigram (μ_i, ν_j) is referred to by z.

Multigram probabilities are calculated using an iterative maximum likelihood formula, Equation 4.3. Initialising the probabilities of the multigrams is done by a simple count of the appearance of each in all possible segmentations. The update equation takes into account the number of times each multigram exists *within* a joint segmentation, and favours segmentations that reuse multigrams.

$$p^{(k+1)}(\mu_i, \nu_j) = \frac{\sum_{S \in \{S\}} c(\mu_i, \nu_j; S) \mathcal{L}^k(\mathcal{W}, \mathcal{U}, S)}{\sum_{S \in \{S\}} c(S) \mathcal{L}^k(\mathcal{W}, \mathcal{U}, S)} \tag{4.3}$$

$c(\mu_i, \nu_j; S)$ is the number of occurrences of the co-sequence (μ_i, ν_j) in S.
$c(S) = \sum_{i,j} c(\mu_i, \nu_j; S)$ is the total number of co-sequences in S.
$\mathcal{L}^k(\mathcal{W}, \mathcal{U}, S)$ is the likelihood of segmentation S at the k^{th} iteration, calculated using the multigram probabilities from iteration k. $\mathcal{L}^k(\mathcal{W}, \mathcal{U}, S) = \prod_t z_t$ for multigrams z_t in segmentation S.

An intuitive understanding of Equation 4.3 follows: There are a number of possible joint segmentations of two input streams, as seen in the 'one seven one' example in Section 4.3.1. The probability of a multigram z is dependent on how often it is seen among these possible segmentations, and on how likely segmentations that contain it are. The term $c(z; S)$, the count of multigram z occurring in segmentation S, in the numerator means that multigram z gains no probability from segmentations that don't contain it. On the other hand the model favours segmentations that use a multigram repeatedly. The sum across all segmentations of the product $c(z; S)\mathcal{L}^k(\mathcal{W}, \mathcal{U}, S)$ is in a sense "normalised" by the denominator, since the denominator takes into account the number of multigrams present overall in each segmentation.

Given an input where the two streams are expressed as utterances, it makes sense to compute these updates utterance-by-utterance. In this case, the update equation becomes Equation 4.4.

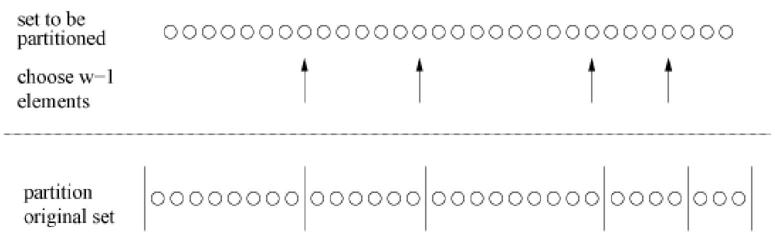


Figure 4.3: Partitioning a set into w ($w = 5$) subsets.

$$
\begin{aligned}
p^{(k+1)}(\mu_i, \nu_j) &= \frac{1}{\|U\|} \sum_{U} \frac{\sum_{S \in \{S_U\}} c(\mu_i, \nu_j; S) \mathcal{L}^k(\mathcal{W}, \mathcal{U}, S)}{\sum_{S \in \{S_U\}} c(S) \mathcal{L}^k(\mathcal{W}, \mathcal{U}, S)} \\
p^{(k+1)}(z) &= \frac{1}{\|U\|} \sum_{U} \frac{\sum_{S \in \{S_U\}} c(z; S) \mathcal{L}^k(\mathcal{W}, \mathcal{U}, S)}{\sum_{S \in \{S_U\}} c(S) \mathcal{L}^k(\mathcal{W}, \mathcal{U}, S)}
\end{aligned}
\tag{4.4}
$$

Where U is the set of utterances, and $\|U\|$ the number of utterances in the data set. S is a possible segmentation from the set of all segmentations of utterance U, $\{S_U\}$.

This update equation quickly leads to a converged solution of multigram probabilities.

4.3.3 Implementation constraints

The process of splitting the units into partitionings is a combinatorics exercise. We are interested in partitionings of $u = \|\mathcal{U}\|$ units into $w = \|\mathcal{W}\|$ (the number of words) partitions. Partitioning a set into w subsets is equivalent to choosing $(w-1)$ elements of the set, and putting a partition after each of these elements. The w^{th} partition is made up of the remaining elements. This is shown simply in Figure 4.3.

The number of ways of choosing these $w-1$ elements is

$$
{}^{u-1}C_{w-1} = \frac{(u-1)!}{(w-1)!(u-w)!}
$$

The number of ways is ${}^{u-1}C_{w-1}$ rather than ${}^{u}C_{w-1}$ since there will never be a partition after the last element of the set, so we are searching for $(w-1)$ places to put partitions among $(u-1)$ element.

The data in Table 4.3.3, shown graphically in Figure 4.4, shows how the number of ways of partitioning inputs quickly becomes very large as the number of words increases.

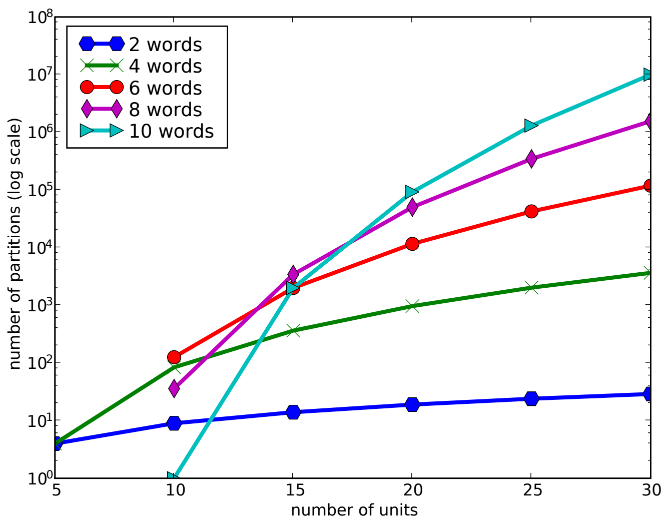


Figure 4.4: Relationship between input number of words and units, and number of partitions required, using equation $\frac{(u-1)!}{(w-1)!(u-w)!}$ (see text).

Note that in the figure the y-axis has log scale. Storage of these is required in order to find multigrams in the input data. The storage method I have used requires $u + 2$ Bytes per partition stored, i.e. $(u + 2)\frac{(u-1)!}{(w-1)!(u-w)!}$Bytes. Since this figure can easily be calculated before processing data for a particular utterance, it is possible to skip utterances that will create more information than there is space for. In this way, it is easy to skip utterances that will take too much time to process, if turn around time is important.

If we remove the constraint of always partitioning the word sequence into individual words, the number of ways of partitioning the two inputs increases. In this case, we're interested in all ways of jointly segmenting two input streams into one or more co-segments. The number of co-segmentations of a pair of input streams is given by

$$\sum_{i=0}^{w} \frac{(w-1)!}{(i-1)!(w-i)!} \frac{(u-1)!}{(i-1)!(u-i)!} \tag{4.5}$$

for streams of length w and u, where $w < u$.

In order to limit the number of segmentations found, a further parameter may be introduced into the model. Prior to doing any segmenting of the unit sequence $\mathcal{U}$, a limit can be set specifying the maximum length of sub-sequences of this input stream.

w	u	number of co-segmentations	
		individual words only	word sequences allowed
2	5	4	5
2	20	19	20
2	35	34	35
2	50	49	50
3	5	6	15
3	20	171	210
3	35	561	630
3	50	1176	1275
4	5	4	35
4	20	969	1540
4	35	5984	7770
4	50	18424	22100
5	5	1	70
5	20	3876	8855
5	35	46376	73815
5	50	211876	292825
6	5	0	126
6	20	11628	42504
6	35	278256	575757
6	50	1906884	3162510
7	5	0	210
7	20	27132	177100
7	35	1344904	3838380
7	50	13983816	28989675
8	5	0	330
8	20	50388	657800
8	35	5379616	22481940
8	50	85900584	231917400
9	5	0	495
9	20	75582	2220075
9	35	18156204	118030185
9	50	450978066	1652411475
10	5	0	715
10	20	92378	6906900
10	35	52451256	563921995
10	50	2054455634	10648873950

Table 4.3: Number of ways of partitioning two input streams of length w and u, comparing figures when stream W is only segmented into individual words, and when word sequences are allowed.

This threshold value, v, is a variable in the final dictionary generation process, since if it is set too low, valuable pronunciations will be lost. As v increases, the time taken to train the multigrams increases substantially, as does the storage requirement during processing.

There are two occasions where finding a joint segmentation is impossible in this implementation of joint multigrams, where the word sequence $\mathcal{U}$ is always segmented into individual words:

1. When $u < w$ where $u = \|\mathcal{U}\|$ is the number of units, and $w = \|\mathcal{W}\|$ is the number of words. If there are fewer units than words, there will be at least one word assigned to 0 units, which is not a valid joint multigram.
2. When $v * w < u$ where v is the maximum sub-sequence length of the unit stream, and u and w are defined as above. If $v * w$ is less than the number of units, there will be units which cannot be assigned to any word due to the sub-sequence length constraint. Thus a joint segmentation is impossible.

4.4 Experiment: Phonetic Dictionary learnt from data

The best way to test the value of a dictionary is by comparing word error rates in an automatic speech recognition experiment. Such an experiment has been carried out to determine the success of the automatic method of dictionary generation, using phonetically transcribed data and hidden Markov models (HMMs) of phones.

4.4.1 Data

These experiments were conducted using the OGI Numbers Corpus, release 1.3[2]. The divisions of the data are as described in the previous chapter, Section 3.5.3.

This corpus is supplied with hand transcriptions at the phone level and at the word level. These pairings of transcriptions for each utterance are used as inputs to dictionary generation processes.

The phone-level transcriptions of this data set are detailed, including many diacritics to more fully describe the perceived sounds. For these experiments, all diacritics were removed, since such a detailed labelling was unnecessary for the purpose of testing the dictionary generation algorithms. The input to both algorithms generating dictionaries is a simple phone sequence. The effect of this is that some pronunciations apparently have repeated phones.

4.4.2 Dictionaries

This experiment tests the various phone-based dictionaries built using the processes described above. Details of these dictionaries follow.

Baseline

The baseline dictionary is a hand written dictionary, written by the author, based on knowledge of the phoneset and the language. The baseline dictionary is shown in Table 4.4. Thirty words, each with single pronunciations, are in this dictionary. There

[2]http://www.cslu.ogi.edu/corpora/numbers/index.html

is one silence word, <pau>, with null output symbol (indicated by '[ ]'). A null output symbol is used simply so that this word is not scored as part of recognition results.

Semi-automatically generated

The supervised ('semi-automatic') algorithm described in Section 4.2 was used to generate a dictionary from the OGI Numbers corpus.

The output of the semi-automatic process is a very large dictionary with many pronunciations per word, some of which are very rare in the data. The mean number of pronunciations per word is 15.8, with a standard deviation across the 30 words of 9.97. There are 8 phone sequences shared between pairs of words. These confusions are shown in Table 4.5.

The large number of pronunciations per word in this dictionary increase the complexity of decoding, since there are many options available to transcribe each word. It has been seen in pronunciation variation work, and is again proved in the experiments in this chapter, that too much variation in the dictionary degrades recognition performance. Therefore this dictionary is tested by reducing it using thresholds involving pronunciation probability. Three thresholds were applied with different effects. These thresholds are listed in Table 4.6.

In each case, the final dictionary has the probabilities of included pronunciations normalised for each word to ensure it is a valid probabilistic dictionary. The effect of these thresholds on the number of pronunciations per word in each dictionary is shown in Table 4.7. A number of the threshold values yield dictionaries that are not useable in the recognition task, when they have no pronunciations for some words. This is seen in the column 'missing words'.

Automatically generated: joint multigrams

The joint multigram process described above was used to generate a dictionary using the OGI Numbers data. The input data was the same as that for the semi-automatic process above: word transcriptions and phonetic transcriptions, with diacritics removed. For

<pau> [ ]	.pau
eight	ei tc th
eighteen	ei tc th i: n
eighty	ei tc th i:
eleven	E l E v & n
fifteen	f I f tc th i: n
fifty	f I f tc th i:
five	f aI v
forty	f > R tc th i:
four	f > R
fourteen	f > R tc th i: n
hundred	h ^ n dc d R ^ dc d
nine	n aI n
nineteen	n aI n tc th i: n
ninety	n aI n tc th i:
oh	oU
one	w ^ n
seven	s E v E n
seventeen	s E v E n tc th i: n
seventy	s E v E n tc th i:
six	s I kc kh s
sixteen	s I kc kh s tc th i: n
sixty	s I kc kh s tc th i:
ten	tc th E n
thirteen	T 3r tc th i: n
thirty	T 3r tc th i:
three	T R i:
twelve	tc th w E l v
twenty	tc th w ^ n tc th i:
two	tc th u
zero	z I R oU

Table 4.4: Handwritten baseline phonetic dictionary for OGI Numbers experiment.

words sharing pronunciation	pronunciation
fifty == fifteen	f I f tc th i:
ninety == nineteen	n aI n tc th i:
twenty == one	w ˆ
seventy == seventeen	s E v & n th i:
sixteen == seventeen	s I kc kh s th i: n
sixty == sixteen	s I kc s tc th i:
three == thirty	T 3r i:
three == thirty	T R i:

Table 4.5: Pronunciation confusions in semi-automatically generated full dictionary

	Threshold type	Description
1	Top n:	Include in the dictionary the top n pronunciations for each word, ordered by pronunciation probability, adding the most likely first.
2	Probability above p:	Add to the dictionary only pronunciations with a probability greater than threshold value p.
3	Probability sum less than s:	For each word, starting with the pronunciation with the highest probability, add pronunciations while the cumulative probability (for that word) remains less than or equal to threshold value s.

Table 4.6: The three thresholds used to generate various dictionaries from a large dictionary which contains large amounts of variation

Dictionary		Pronunciations per word		Missing
		mean	st.dev.	words
Full dictionary		15.8	9.97	0
Top N	1	1.0	0.00	0
	2	2.0	0.00	0
	3	3.0	0.18	0
	4	3.9	0.40	0
	5	4.8	0.67	0
	6	5.5	0.99	0
	7	6.3	1.36	0
	8	7.0	1.77	0
	9	7.7	2.18	0
Prob >	0.001	10.6	7.07	0
	0.005	7.6	5.69	0
	0.010	5.2	3.09	0
	0.050	2.0	1.10	0
	0.100	1.5	0.62	0
	0.500	1.0	0.00	5
Prob sum <=	0.50	1.0	0.00	25
	0.75	1.7	0.62	18
	0.80	2.4	1.27	17
	0.95	4.8	4.63	7
	0.97	5.7	5.46	2
	0.99	8.0	6.58	0

Table 4.7: Statistics for the dictionaries generated semi-automatically.

this joint multigram process, no extra information is available, for example pauses are not treated differently to phones, and no timing information is used.

The full dictionary created is much larger than that of the supervised process. There is a mean of 8539.2 pronunciations per word in this dictionary. This is due to the fact that all co-sequences seen during training appear in the dictionary. After computing and updating their probabilities, many of these receive a probability of zero. The same three threshold types were then used to produce simpler dictionaries which can be used in recognition. Table 4.8 shows some of the statistics of these generated dictionaries. As for the supervised dictionary above, some of the thresholds produced dictionaries with words missing, which are not useable in the recognition task.

4.4.3 Acoustic Modelling

Hidden Markov models (HMMs) were built for each of the 60 phones in the data set. This figure includes all the vocal noises and pauses, such as ".laugh" , ".cough" and ".pau". The phone set is shown in Table 4.9. The HMM topology used was 3 states per phone model. The HMMs were initialised with each state having the global mean and variance of the training data. The training data were parameterised using MFCCs, with deltas and double deltas, resulting in 39 dimensional data. The number of Gaussians per state is a variable in the system and has been investigated.

The models were trained using OGI Numbers' phonetic transcriptions, without any time information. HTK[3] was used to train this model set, using embedded training.

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

Dictionary		Pronunciations per word		Missing
		mean	st.dev.	words
Full dictionary		8539.2	12183.86	0
Top N	1	1.0	0.00	0
	2	2.0	0.15	0
	3	2.9	0.34	0
	4	3.8	0.59	0
	5	4.7	0.90	0
	6	5.5	1.23	0
	7	6.4	1.57	0
	8	7.2	1.92	0
	9	8.0	2.26	0
Prob >	0.001	24.2	17.91	2
	0.005	12.7	9.01	0
	0.010	8.1	5.03	0
	0.050	3.6	1.48	0
	0.100	2.7	1.03	0
	0.500	1.0	0.00	25
	0.750	1.0	0.00	30
	0.900	1.0	0.00	30
Prob sum <=	0.25	1.0	0.00	25
	0.50	1.6	0.74	5
	0.75	3.3	2.69	0
	0.80	4.3	3.75	0
	0.95	16.4	14.93	0

Table 4.8: Statistics for the joint multigram based dictionaries.

.bn	.glot	.pv	&r	3r	d	I	m	r	u
.br	.laugh	.sniff	@	9r	dc	i:	n	r(	U
.bs	.ln	.tc	ˆ	A	E	j	N	s	ux
.cough	.ls	.vs	+	aI	ei	kc	n=	T	v
.ct	.ns	&	>	aU	f	kh	oU	tc	w
.fp	.pau	&0	>i	b	h	l	pU	th	z

Table 4.9: Units used in OGI Numbers phone dictionaries experiment

4.4.4 Method

Training

1. Initialise 60 phone- and noise-models with the global mean and variance of the OGI Numbers training data. All models are 3-state HMMs with 1 Gaussian per state.

2. Train model set for 3 iterations using embedded training and a non-time-aligned phone transcription.

3. Extract the centre state of the pause model ('.pau') to create a new, single state, model sp ('short pause'). This is included to allow breaks between words. It is an internal model, and as such its use or otherwise in decoding an utterance is not output. Recognition of short pause is never included in word error rates. The inclusion of the model significantly improves word error rates.

4. In order to have models with different numbers of Gaussian mixture components, the single Gaussian components of these trained models are split using the HTK recipe for 'mixing up'. The number of Gaussians is increased in increments of 2. Three further training iterations are then carried out imbetween each 'mixing up' stage. The product of this step is a set of models with various numbers of mixture components trained on phonetic transcriptions.

5. For each dictionary, D, to be used in decoding, a realignment procedure is carried out. This is in order to ensure that only pronunciations that are in D are in the training data. Without this step, sub-word unit sequences that do not occur in the dictionary may be used in training, but cannot be recognised in decoding since the dictionary is integral to the decoding process. Realignment is necessary despite the fact that the dictionary generation algorithms attempt to reflect the whole of the training data, because various thresholds are used to limit the number of pronunciations per word as listed in Table 4.6. Therefore the final dictionaries to be tested may not be entirely reflective of the unit sequences seen in the training data.

   The realignment process, carried out using the HTK tool HVite, produces a new transcription for the data. The process uses the training data's word transcription, the dictionary, the trained models, and the acoustic training data. The output is the most likely sub-word unit sequence given these inputs, which for each word is the best match between one of the pronunciations of the word (from the dictionary) and the acoustic data.

This new transcription is used to train the models a further 5 iterations, ready for use in decoding using dictionary D. There is a large number of dictionaries to be tested, and a large number of models (the output of the previous step is a large set of possible models), so this step of realignment and retraining results in many model sets to be tested.

Decoding

To test dictionary D with models with g Gaussians per state, the model set M_D^g is used. This is the set of models trained in step 4 of the training process, using a transcription which reflects the pronunciations in dictionary D. The decoding parameters that need to be determined for each model set and dictionary combination are:

- Beam width: a pruning threshold can be set which removes all decoding hypotheses whose probability falls below this threshold value. This has the effect of speeding up the search.
- Language model scaling factor: Also called the grammar scale factor, this is a positive value which is used to adjust the contribution made to likelihoods during decoding of the language model.
- Word insertion penalty: This is a value which is added to the likelihood of each word through the decoding process, to control the number of insertions in the recogniser output.

These parameters, particularly the final two, can have a significant effect on recognition scores, and hence they are tuned over a held-out dataset, the validation data. Once a peak word recognition result, in terms of word error rate, has been reached across these variables on validation data, the values are used to generate a final score on test data, which is previously unseen. In this way, the dictionaries can be compared.

The grammar used in all decoding tasks is a simple word loop, where each word has equal probability of following each other word (including itself), and there is an optional short pause between words.

4.4.5 Results

The results of this phone-based experiment show that joint multigrams provide a good way of generating dictionaries automatically. Table 4.10 shows word error rate (WER) results for each dictionary type. The results reported on validation data are the maximum obtained for each dictionary over a search of the decoding space (as described in Section 4.4.4) and model sets (number of Gaussian mixture components). The number of mixture components is reported for each dictionary. The final column in the table is the WER for each dictionary when tested on unseen test data.

The baseline word error rate (WER) of 8.41% on validation data is a good baseline for monophone models: it is 0.29% better than the equivalent baseline (trained and tested on the same data) in Mariethoz & Bengio (2004).

The results for dictionaries generated using the top n and prob $> p$ thresholds are shown graphically in Figure 4.5. As is easily seen in the graphs, all dictionaries generated by joint multigrams out-perform semi-automatically generated dictionaries, comparing threshold by threshold. All multigram-based dictionaries achieve a lower word error rate than the baseline's test data score. An absolute gain of 2.08% on validation data is achieved using multigram dictionaries, which is further increased to 3.44% on test data. This gain exceeds that achieved using the dictionaries produced in a supervised, semi-automatic manner, where the improvements on the validation and test data are 1.42% and 2.64% respectively.

It is interesting to compare the 'winning' dictionaries from each dictionary generation method. The top performing semi-automatically generated dictionary is the top 1, and the top performing multigrams-based dictionary is the top 3. In comparing the pronunciations of each of the words in the dictionaries, the typical case is that the 3 pronunciation in the multigrams dictionary have the same basic phone sequence as the single semi-automatic pronunciation, with the pause label ('.pau') in different

dictionary	Validation data		Test data
	WER (%)	mix comps	WER (%)
hand written baseline	8.41	46	9.63
semi auto			
top N 1	6.99	40	6.99
2	7.01	46	6.88
3	7.02	56	7.07
4	7.66	66	7.48
5	8.41	66	8.2
6	8.73	66	8.48
prob sum <= 0.99	8.83	76	8.56
top N 7	8.87	66	8.49
8	9.46	46	9.02
9	10.08	40	9.94
prob > 0.100	14.82	40	7.83
0.050	15.23	40	8.04
0.010	17.32	40	9.79
0.005	17.95	36	11.02
0.001	18.78	40	11.53
multigrams			
top N 3	6.32	40	**6.19**
top N 6	6.36	70	6.43
top N 5	6.46	60	6.6
top N 2	6.56	40	6.22
prob > 0.100	6.61	40	6.49
prob > 0.050	6.61	46	6.51
top N 4	6.72	46	6.54
top N 1	6.74	50	6.76
prob sum <= 0.750	6.79	60	6.58
top N 7	7.27	56	7.16
prob sum <= 0.800	7.37	50	7.28
prob > 0.010	7.87	40	7.82
top N 8	7.87	46	7.7
top N 9	8.04	40	7.8
prob > 0.005	8.22	66	7.63
prob > 0.001	9.45	66	9.02
prob sum <= 0.950	9.78	50	9.57

Table 4.10: Word Error Rates for recognition experiments using all dictionaries.

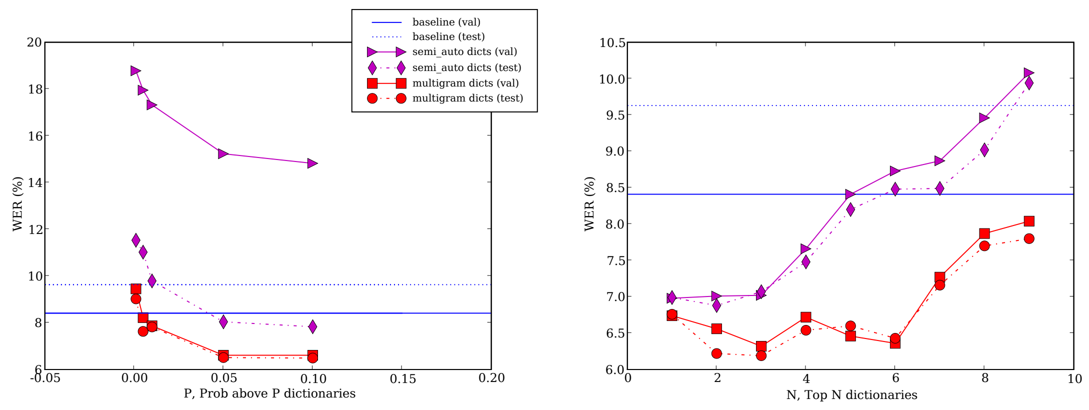


Figure 4.5: Word error rates for multigram and semi-auto dictionaries, compared to a hand-written baseline dictionary. The left figure shows results for dictionaries with pronunciations with probability above P, and the right dictionaries with N pronunciations per word (Top N).

places around the phones. Table 4.11 gives an example of this typical case, and the seven exceptions. This result implies that modelling pauses in the lexicon has a positive effect on word recognition, which is an unexpected outcome of creating these automatic dictionaries, where deliberately no special meaning was associated with pause (i.e. pause was treated in the same way as phonemes in the dictionary generation process). This may be a data-specific property, since utterances in the OGI Numbers data set have an average of 5 words, and there are only 30 words in total in the corpus, so most, if not all, words will appear at the beginning and the end of an utterance, hence after or before a pause. So while we cannot conclude that pauses should always be modelled in the lexicon in this way, we can have confidence in the reliable way that joint multigrams produce a dictionary which reflects the data appropriately and successfully.

Typical comparison		
word	multigram dict	semi-auto dict
five	f aI v .pau f aI v .pau f aI v	f aI v

Phonetic variations		
word	multigram dict	semi-auto dict
eight	ei tc th .pau ei tc .pau ei tc th	ei tc th
eighty	.pau ei tc th i: .pau ei tc d i: ei tc d i:	ei tc th i:
eleven	& l E v & n .pau .pau I l E v & n .pau & l E v & n	& l E v & n
hundred	h ^ n dc d R I dc d .pau h ^ n dc d R I dc d h ^ n dc d R ^ dc d .pau	h ^ n dc d R I dc d
nineteen	.pau n aI n tc th i: n n aI n tc th i: n .pau n aI n tc th i: n	aI n tc th i: n n
twenty	tc th w & n i: tc th w & n i: .pau tc th w & n tc th i:	tc th w & n i:
two	tc th u tc th u .pau .pau tc th u	tc th u

Table 4.11: Comparing pronunciations in top performing multigram dictionary, top N 3, with top performing semi-automatic dictionary, top N 1. The upper table shows the typical case, where, stripped of .pau (pause), the 3 phonetic pronunciations in the multigram dictionary are identical to the 1 semi-automatic pronunciation. The lower table lists all seven exceptions to this rule.

4.5 Chapter summary

This chapter introduced the problem of automatic dictionary generation and presented a solution using joint multigrams. The experiment using the phonetic transcriptions of the OGI Numbers dataset showed that this method is very good: the multigram dictionaries consistently out-performed the dictionaries created by a supervised method, evidenced by word error rates.

Chapter 5

Speech recognition system based on automatically derived units

This chapter presents experimental investigations into the combination of automatically derived SWUs and joint multigram-based dictionaries.

5.1 Generic experimental procedure

The ASR systems tested here comprise a set of acoustic models $\mathcal{A}$, one for each SWU in a unit inventory, and a dictionary relating SWUs to words. The acoustic models were generated using the ergodic HMM procedure of Section 3.3.1, by embedded training on all OGI numbers training data. There are S units in an acoustic model set, where theoretically S is chosen using BIC. Each unit is modelled by 1 or 3 HMM states, depending on the set up of the ergodic HMM (see Section 3.5.3), and has G Gaussian mixture components per state. The dictionary $\mathcal{D}^v_{\mathcal{A}}$ for the model set $\mathcal{A}$ is generated using joint multigrams, as described in Section 4.3. During the generation of the dictionary, there is a constraint on the maximum length of a unit sequence (pronunciation), v. The full dictionary is reduced using one of the thresholds n, p or s as described in Table 4.6. These variables of the full system are listed in Table 5.1.

For an acoustic model set $\mathcal{A}$ and associated generated dictionary $\mathcal{D}^v_{\mathcal{A}}$, a recognition experiment is carried out. Dictionary $\mathcal{D}^v_{\mathcal{A}}$ is reduced using a threshold, generating a set

	Variables for experimentation
unit derivation	- number of states per unit (single or 3 state) - number of units (theoretically chosen by BIC) - number of Gaussians per state in HMMs
lexicon generation	- v (maximum pronunciation length, in terms of SWUs) - dictionary reduction method and threshold (top n, prob $> p$, prob sum $\leq s$)

Table 5.1: Variables in the full system, based on units derived using an ergodic HMM with BIC, and dictionaries generated using joint multigrams.

of dictionaries $\{\mathcal{D}^v_{\mathcal{A}}(n=1), \mathcal{D}^v_{\mathcal{A}}(p=0.005), \mathcal{D}^v_{\mathcal{A}}(s=0.5)...\}$ for various values of n, p or s. The procedure for a recognition experiment with each dictionary is as follows:

- split the ergodic HMM into individual HMMs, one for each SWU. The model for each SWU is either single state or 3state, depending on the original set up of the erogdic model.
- realign: generate a transcription of the data by forced alignment using the dictionary and the set of SWU models.
- retrain: use the transcription generated by realignment to further train the acoustic models of SWUs.
- decode: use the trained models and dictionary to decode validation data. The same decoding variables are searched over as described in Section 4.4.4, the language model scaling factor, word insertion penalty and beam width.

5.2 Initial results and analysis

Initial experiments were carried out using the 1state and 3state models trained on all OGI numbers training data, as described in Chapter 3. Joint multigrams-based dictionaries were built for a variety of these models, and tested in the standard way.

Table 5.3 shows results for model sets comprising single state units and dictionaries

3state	80units (35g)	120units (30g)	200units (30g)
top N 1 (v10)	50.54	-	29.19
top N 2 (v10)	55.12	41.5	39.3
top N 3 (v10)	57.58	43.71	44.51
top N 4 (v10)	47.09	48.29	50.27
top N 5 (v10)	44.78	49.05	51.64
top N 6 (v10)	48.79	44.56	52.87
top N 7 (v10)	48.31	45.39	55.01
top N 8 (v10)	47.89	45.2	52.32
top N 9 (v10)	46.29	46.57	53.96
prob > 0.001000 (v10)	40.17	35.34	42.86
prob > 0.005000 (v10)	52.49	52.74	64.06
prob sum <= 0.250000 (v10)	49.71	32.11	-
prob sum <= 0.500000 (v10)	28.0	24.66	-
prob sum <= 0.750000 (v10)	-	24.96	-
prob sum <= 0.800000 (v10)	-	25.45	-
prob sum <= 0.950000 (v10)	-	27.17	-

Table 5.2: WER results on validation data for various model sets (column headings show number of units in model set, with number of gaussian mixture components per state in brackets) for 3state base models and multigram dictionaries. Dictionary parameters are shown in the first column, including v, the maximum allowed length of unit sequences.

trained on each of the model sets. The best word error rate (WER) of 19.81% was achieved using 200 units and a dictionary with threshold $s \leq 0.75$. Table 5.2 shows equivalent results for model sets comprising 3state HMMs for each unit. The best result of 22.60% was achieved using 120 states and a dictionary with threshold $s \leq 0.75$.

These results are more than double the phone baseline, which is extremely poor. Analysis follows to determine the causes of these results. The analysis that follows is focused on the two models which achieve the lowest WER: for single state units, the model set with 200 units, each with 20 Gaussian mixture components per state (model set A), and for 3state units, the model with 120 units and 30 Gaussian mixture components per state (model set B).

1state	100units (30g)	150units (30g)	200units (10g)	200units (20g)	200units (30g)	250units (20g)
top N 1 (v15)	62.62	69.12	58.66	42.41	33.54	47.67
top N 2 (v15)	43.94	46.27	-	40.05	39.25	44.24
top N 3 (v15)	41.13	44.84	-	44.87	44.06	45.15
top N 4 (v15)	42.38	43.55	52.38	39.47	30.2	37.61
top N 5 (v15)	45.69	39.69	45.68	32.81	-	-
top N 6 (v15)	44.32	42.09	39.88	32.8	-	36.73
top N 7 (v15)	43.15	41.68	37.46	33.01	-	32.0
top N 8 (v15)	42.59	42.34	35.55	34.03	29.18	34.19
top N 9 (v15)	42.08	40.91	34.5	-	28.2	86.26
prob > 0.001000 (v15)	39.88	37.24	30.24	31.07	27.84	-
prob > 0.005000 (v15)	59.01	62.09	-	-	55.25	-
prob > 0.010000 (v15)	-	58.74	-	-	-	-
prob sum <= 0.500000 (v15)	23.25	-	-	20.24	23.73	-
prob sum <= 0.750000 (v15)	22.89	26.08	-	19.81	23.59	24.15
prob sum <= 0.800000 (v15)	23.18	23.61	-	23.98	20.62	24.57
prob sum <= 0.950000 (v15)	23.79	24.33	-	-	20.92	25.36

Table 5.3: WER results on validation data for various model sets (column headings show number of units in model set, with number of gaussian mixture components per state in brackets) for 1state base models and multigram dictionaries. Dictionary parameters are shown in the first column, including v, the maximum allowed length of unit sequences.

model	num utts	
	v10	v15
100units (30g)	615	5430
150units (30g)	422	4709
200units (10g)	141	2822
200units (20g)	157	3020
200units (30g)	180	3149
250units (20g)	174	3022

Table 5.4: Number of utterances successfully used in dictionary generation for SWU inventories with one state per unit, out of a possible 10250.

5.2.1 Effect of dictionary generation constraint, v

The maximum allowed length of unit sequences v is externally specified in joint multigrams training in order to constrain the number of joint multigrams found, and to make the search tractable. The effect of this limit is that sequences of units cannot be considered as pronunciations, if they are longer than v. For the 1state units, the mean number of units per word across all inventories trained is 15.3 units. For model set A, this value is 17.7 units. Considering the fact that there are words of various lengths in the data, longer words will be modelled by more than this figure. Since (due to computational constraints, memory requirements in particular) the maximum value of v tested was 15, it is unlikely that these longer words are receiving pronunciations of appropriate lengths.

If the value of v is such that $v * w < u$ for an utterance with w words and u units, the utterance cannot be jointly segmented (see page 68 of Section 4.3.3). Thus for low values of v, much training data is ignored, and the resulting multigram dictionary may contain no pronunciations for certain words. For this reason, a value of $v = 10$ is too small for the 1state models. The number of utterances used to train each 1state dictionary, out of possible 10250, is shown in Table 5.4.

For the 3state models, the mean number of units per word is 7.5 across all models, and 8.4 for model set B. Since this value is less than the value of v tested, and WER results for these models did not improve on the 1state models, other causes for the poor results were sought.

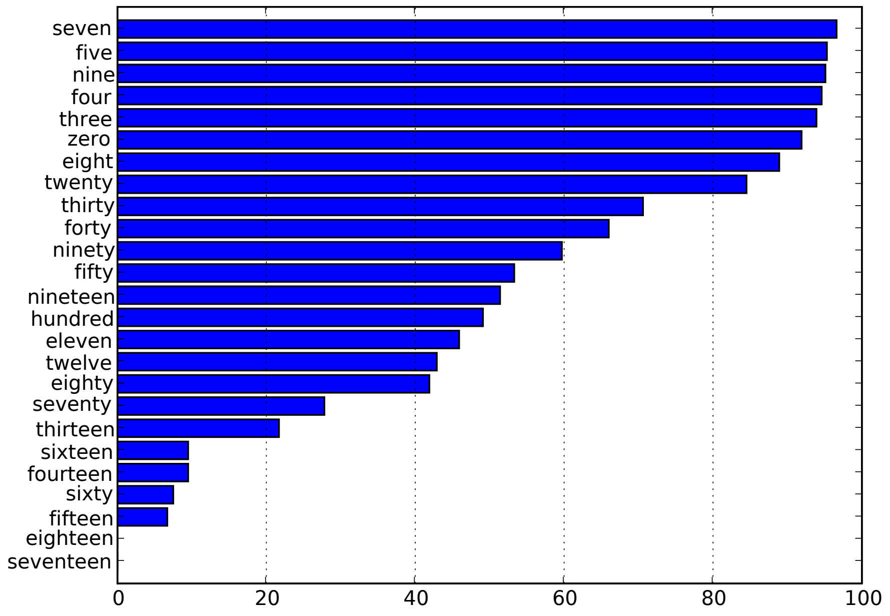


Figure 5.1: For each word, the percentage of times it is correctly recognised in validation data for single state model and dictionary achieving lowest WER (200 units, prob sum ≤ 0.75)

5.2.2 Error analysis

In analysing the word confusion matrix for the best result, insertions, substitutions and deletions are occurring across all words. Figure 5.1 shows the percentage of each word's tokens in the validation set being correctly recognised, for model set A. Words 'eighteen' and 'seventeen' are never recognised correctly in this result. Both of these words occur infrequently in the training data, and in the dictionary generation process, most of the utterances containing them were not used, due to the effect of parameter v discussed above. Only three utterances containing 'seventeen' and four containing 'eighteen' were used as part of multigrams training. This lack of representation in the multigrams dictionary training will surely degrade the quality of the pronunciation of any word, and affect its recognition.

5.2.3 Treatment of silence

The way silence is treated is an important factor in a speech recognition system. This was seen in the experiments of Chapter 4, where it became apparent that to achieve a reasonable baseline one must use a pause model and allow pauses between words as part of the language model (the way this pause model was set up is described in Section 4.4.4). In the ergodic model training of Chapter 3 used to derive the units used in these experiments, no information about pause locations was included. Instead, the ergodic model was required to find SWUs for silence as well as for speech. A closer look at what the ergodic model did suggests that silence detection may be an important pre-cursor to SWU derivation.

Analysis was carried out by comparing the locations of pauses available in the detailed phonetic transcriptions of OGI numbers data with the locations of the SWUs. The first thing to note is that in both systems all SWUs are used during a pause location at least some of the time. The plot in Figure 5.2 shows the percentage of the use of each unit occurring within a region labelled as 'pause' in the hand transcription. This figure is calculated for each unit U as

$$\frac{\texttt{occurrences of U within a pause region}}{\texttt{total occurrences of U in transcription}} * 100$$

For a good distinction between pause and speech modelling, we would expect some number of units to be used in pause regions almost all of the time, thus modelling pause, and the remainder of the units occurring during a pause region only rarely, as graphically represented (with hypothetical values) by the dotted green line in Figure 5.2. This is not seen in these model sets.

Table 5.5 shows the units occurring with a high likelihood during pause regions and the units used in the pronunciation of pause. It is clear from these data that no clean pause model has been learned as part of the unit inventory derivation and dictionary generation: there is very little overlap between the two lists for either model set (6 units for model set A, and 4 for B). We saw in the phone-based experiments of chapter 4 that the correct pause pronunciation ('.pau') was learned for the pause word ('<pau>'), so this is not a problem with the task, and is unlikely to be a problem with the dictionary generation process. Instead, the inconsistencies between the units used in pause and the units used to describe the 'pronunciation' of pause are likely to be due to the acoustic

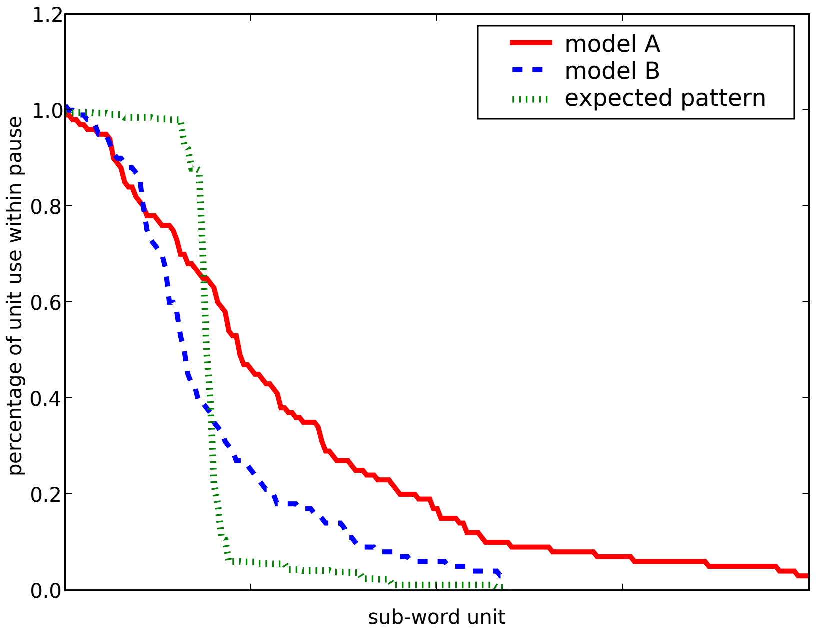


Figure 5.2: Analysing how pause is modelled by each of the model sets A and B (the models achieving the lowest WER for single state units and 3state units, respectively, see text). The plots show the percentage of the use of each unit that occur during a 'pause' region.

Model A	
Units 90% likely to occur within pause region	8 12 85 89 93 97 101 111 119 123 147 151 159 163
Occurring within top 3 pause pronunciations	6 11 12 27 28 33 40 61 67 85 92 97 101 123 128 129 137 147 151 155 169 192
Model B	
Units 90% likely to occur within pause region	26 44 65 71 83 122 158 188 233 236 275 293 311 317 338 344
Occurring within top 3 pause pronunciations	5 53 101 107 143 158 215 266 293 308 317

Table 5.5: Analysis of the modelling of pause regions for model sets A and B. The percentage $\frac{\texttt{occurrences of U within a pause region}}{\texttt{total occurrences of U in transcription}} * 100$ was calculated for each unit; those units occurring within a pause region with a frequency of $> 90\%$ are listed here. The units used in the top three pronunciations of the word <pau> are listed for comparison.

modelling. It would appear that expecting an ergodic HMM to discriminate between speech and silence without any prior on the difference between the two acoustic spaces is too difficult a task.

5.3 Modelling silence

In order to test the hypothesis that the poor initial results are due - at least in part - to the modelling of silence, new acoustic models were trained, given information about silence locations.

5.3.1 Silence experiment 1

In silence experiment 1, two models were trained: a 3 state pause model (as in the phone-based experiment in Section 4.4.4), and an ergodic model constructed as before (Section 3.3.1), resulting in a SWU inventory. The transcription used to train the two models preserved the locations of the '.pau' unit from the hand transcriptions, and compressed all other units to a single label. The ergodic model was then trained using embedded training on all the data except pause regions, and the pause model on pause regions.

Recognition results for these new unit inventories does not improve on the original models, with a best WER for 1state units of 20.55% for SWU inventory of 160 units (with 30 Gaussian mixture components per state) (referred to in analysis as model set C) tested with a dictionary with threshold $p > 0.001$. The best result for 3state units is 27.71% WER for SWU inventory of 40 units (with 35 Gaussian mixture components per state) (referred to as model set D) tested with a dictionary with threshold $n = 7$.

Analysis of the treatment of silence regions for these model sets exposes similar behaviours as noticed in the original models. For the unit inventory with 160 units and 1 state per unit (model C), the pause model is used only 12% of the total time hand-labelled as pause. The remaining 88% is modelled by all of the units at least some of the time, with 12 units (including '.pau') occurring during a pause region more with a frequency of more than 90% (calculated as above, for each unit U as

$$\frac{\texttt{occurrences of U within a pause region}}{\texttt{total occurrences of U in transcription}} * 100$$

). This is a similar distribution to the original models. This is typical of the 1state models trained as part of silence experiment 1.

Regarding the pause unit, '.pau', 98% of its use occurs during a pause region, suggesting that it is a good model for *some* of the acoustic space of silence - it is not getting confused with speech - but it does not account for much of the silence (12%).

The freedom of the ergodic model compared to the pause model must account for this behaviour. The ergodic model has many more states and transitions than the pause model, and so is able to consume large amounts of the silence space. The training procedure (embedded training) only ensures that the *sequences* of pause and speech found in the transcriptions are respected, not the locations of the boundaries. Boundaries are adjusted according to the maximum likelihood objective of HMM training: clearly many of the states of the ergodic model become better models of silence than the pause model. This in itself is not a problem; the problem is the lack of distinction for most of the states between whether they are modelling speech or silence, as discussed in Section 5.2.3 . Again, if the distribution of modelling of the speech / silence acoustics by states was as the green dotted line in Figure 5.2, these models would be expected to perform much better.

5.3.2 Silence experiment 2

A further experiment was carried out to determine finally whether the units derived using an ergodic HMM can be modelled by joint multigrams to provide a system achieving good error rates. To avoid these issues with modelling silence, silence regions were eliminated, and regions of data containing only speech were used for training. The procedure to train and use these 'speech only' ergodic models was as follows:

- Take the time aligned phonetic hand transcriptions available with the corpus, and map all noise words on to a single unit, '.pau' and all speech sounds to a single unit
- Chop the waveform of each utterance to extract regions containing only speech, and parameterise each speech waveform as 39 dimensional MFCCs
- Follow the standard training process of Section 3.3.1 to train various sets of automatically derived 1state- or 3state-units and transcribe the data in terms of the units
- Generate multigram dictionaries for all words in the training data. Append to the dictionary the transcription '.pau' for each noise word and pause.
- Use the model set and the pause model trained for the phone model experiment (including 'sp' - short pause, the centre state of the pause model - optionally between words) to do word recognition, in the standard way as described in Section 4.4.4. Realignment using the dictionary is only carried out on speech data: the only models realigned are the sub-word units, not the pause model.

The plots of BIC scores following this training configuration are seen in Figures 5.3 and 5.4. These curves clearly follow the expected shape of a BIC curve, with maximal scores at 200 units for 1state SWUs, and 120 units for 3state SWUs.

The values of the BIC scores for models trained on speech data alone is higher than for those trained using all training data, as can be seen by comparing the graphs in Figures 3.9 and 3.11 with those in Figures 5.3 and 5.4, noting the scales on the y-axis. This is directly due to the higher training likelihood of the models trained on speech

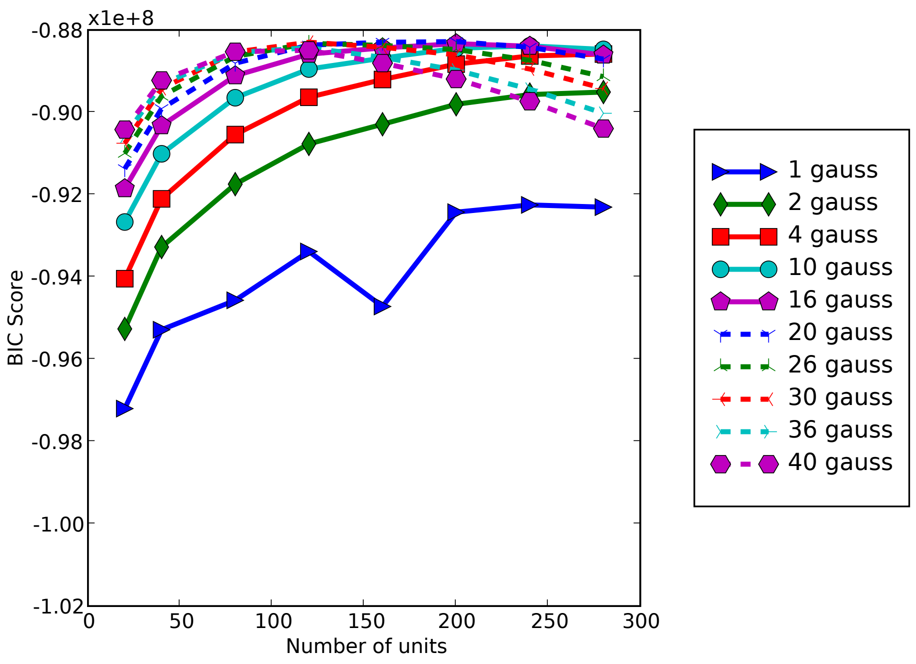


Figure 5.3: Best BIC score for ergodic model trained only on speech (silence experiment 2), across model size (number of 1state units) for various numbers of mixture components per state ('gauss' in legend).

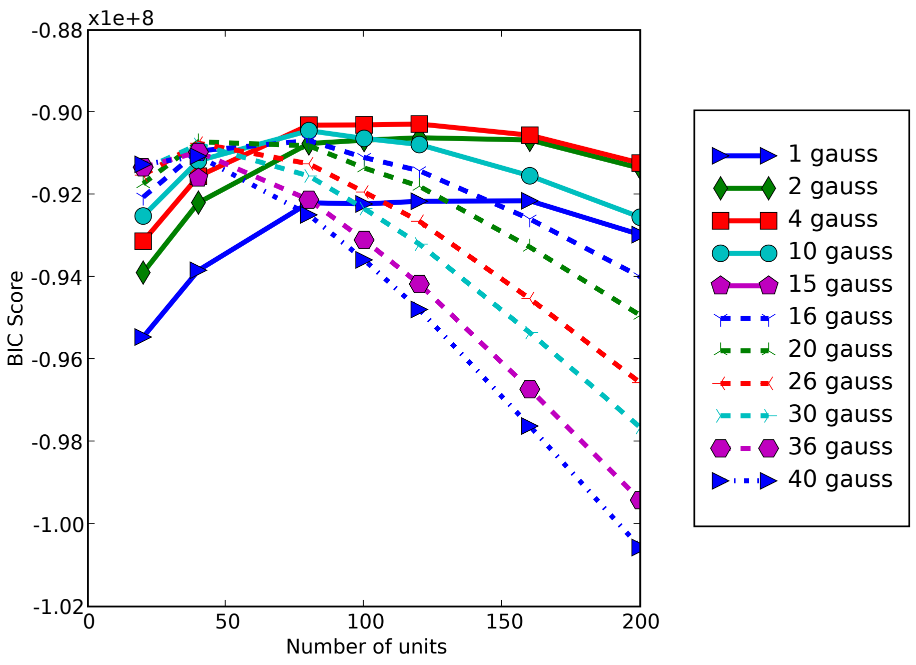


Figure 5.4: Best BIC score for ergodic model trained only on speech (silence experiment 2), across model size (number of 3state units) for various numbers of mixture components per state ('gauss' in legend).

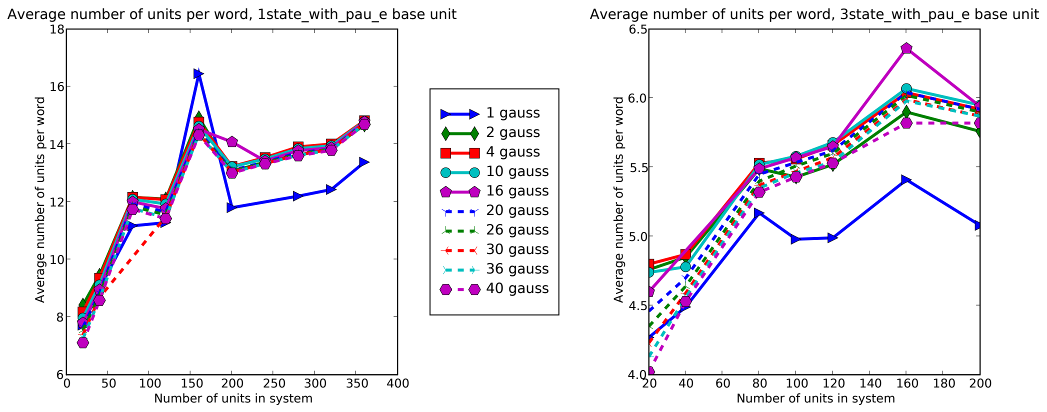


Figure 5.5: Average number of BIC units per word, as number of states increases, for models trained in silence experiment 2.

only. This higher likelihood is indicative of the comparative homogeneous nature of the data contained in speech-only regions, compared to data across both speech and silence regions.

The average number of units per word for each of the model types are shown in Figure 5.5.

Results

The recognition experiments using these speech-only models and dictionaries generated from them yielded a much improved WER of 11.07% (11.18% on test data), for 1state SWUs from a unit inventory of size 200units. Table 5.6 shows WER results for all generated dictionaries on validation data. It is seen that a number of the models and related dictionaries achieve improved scores compared to all earlier results. This demonstrates the importance of careful modelling of silence.

The best WER of 11.07% was achieved using a SWU of 200 units, each unit being modelled by single state HMMs with 20 Gaussian mixture components per state. The dictionary generated from this SWU achieving this best score was generated using the threshold $n = 1$, i.e. the dictionary contains the most likely single pronunciation for each word. While this WER is a large improvement on the scores reported for other model sets in this thesis, it is still worse than the phonetic baseline of 9.63% (this baseline is for monophone models and a handwritten dictionary, see Section 4.4). This

1state (silence exp 2)	80units (40g)	120units (30g)	200units (20g)	240units (20g)
top N 1 (v15)	18.71	13.32	11.07	24.59
top N 2 (v15)	-	15.54	11.9	21.89
top N 3 (v15)	20.6	23.91	15.01	-
top N 4 (v15)	19.68	19.28	16.73	20.4
top N 5 (v15)	22.26	22.02	16.55	22.05
top N 6 (v15)	-	22.39	17.26	21.83
top N 7 (v15)	-	24.01	-	21.65
top N 8 (v15)	-	24.86	-	21.7
top N 9 (v15)	22.84	25.39	-	19.6
prob > 0.001 (v15)	27.17	38.28	28.48	29.91
prob > 0.005 (v15)	29.56	36.54	25.85	-
prob > 0.010 (v15)	-	28.75	22.81	-
prob sum <= 0.75 (v15)	20.92	29.73	-	-
prob sum <= 0.80 (v15)	20.75	27.17	-	-
prob sum <= 0.95 (v15)	20.61	21.77	-	-

Table 5.6: WER results on validation data for various model sets (column headings show number of units in model set, with number of gaussian mixture components per state in brackets) for 1state (silence experiment 2) base models and multigram dictionaries. Dictionary parameters are shown in the first column, including v, the maximum allowed length of unit sequences.

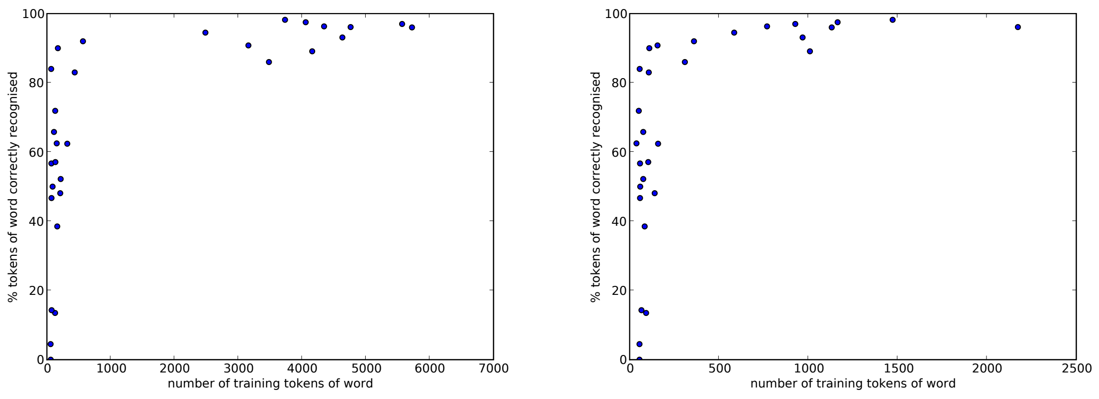


Figure 5.6: The correlation between the number of tokens of each word and the recognition rate of the word (for model and dictionary achieving the lowest WER). The left plot shows the total amount of training data (used for the SWU generation), and the right the amount of training data used in dictionary generation.

is likely to be due in part to the problem of data-sparsity when generating the dictionary automatically, caused by the dictionary threshold v as noted in Section 5.2.2 for the initial experimental results.

The number of training tokens of each word in the data appears to play a significant role in the recognition rates of words. The data in Table 5.7 shows the percent correct scores for each word for the 'winning' model set (200 SWUs, each with 20 Gaussian mixture components per state) and dictionary (threshold $n = 1$). Some of these scores are poor, notably 'seventeen' and 'fourteen', recognised correctly 0% and 4.5% of the time, respectively. These two words occur in the training data infrequently. The first graph in Figure 5.6 shows the relationship between the percent correct scores and the number of training tokens of each word in the training data. It is clear from this plot that there is a high correlation between the amount of training data for each word and the word's recognition rates. Since the dictionary generation process does not use every utterance, it is possible that pronunciations are learnt for some words based on smaller amounts of data. The second plot in Figure 5.6 shows the relationship between the percent correct scores and the number of training tokens of each word used to train the dictionary. This graph exhibits a slightly different shape, showing some words were correctly recognised a high proportion of the time without a lot of training data. This distinction implies that there is a higher dependency on the amount of training data available for acoustic modelling than on the amount used to generate pronunciations.

word	percent correct	num training tokens	num training tokens used in dict. gen
seventeen	0.00	60	55
fourteen	4.50	58	55
seventy	13.50	128	92
sixteen	14.30	74	66
nineteen	38.50	162	84
eighteen	46.70	72	58
fifty	48.10	209	140
twelve	50.00	89	59
ninety	52.20	216	76
fifteen	56.70	70	58
sixty	57.10	133	104
forty	62.40	319	159
eighty	62.50	154	38
eleven	65.80	111	76
ten	71.90	130	51
thirty	83.00	436	107
thirteen	84.00	67	56
eight	86.00	3480	309
four	89.10	4158	1009
hundred	90.00	172	110
oh	90.80	3158	156
twenty	92.00	565	360
nine	93.10	4631	968
zero	94.50	2484	585
one	96.00	5722	1131
seven	96.10	4760	2173
three	96.30	4344	769
two	97.00	5567	927
five	97.50	4056	1164
six	98.20	3732	1472

Table 5.7: Analysis of results for model and dictionary achieving the lowest WER in silence experiment 2 (200units, 1state per unit, dictionary threshold $n = 1$). For each word, the percent correct scores are shown, along with the number of training tokens of each word in the full training data set, and the number of training tokens of each word used in the dictionary generation.

5.4 Summary

Directly combining the SWU inventories of Chapter 3 with joint multigram dictionaries did not lead to promising results. Investigating the modelling of silence among the SWUs showed there was a lack of distinction between speech and silence modelling. Most SWUs were being used to model both speech and silence, and those SWUs modelling silence with a high likelihood were not reflected in the 'pronunciation' of pause in the learned dictionary.

Revising the training method to include a 3-state pause model to train on the regions transcribed as pause, with the same ergodic HMM training on speech gave no improvement. The same inconsistencies were seen: in the embedded training process the boundaries between pause and speech regions were significantly altered, with the pause model only accounting for 12% of the pause regions, and most of the SWUs occurring during a pause region some of the time.

Finally, forcing the ergodic model to only generate SWUs for speech was achieved through cutting each utterance at speech / silence boundaries and training the ergodic HMM on speech only. This process led to a much improved WER of 11% for single state SWUs.

Chapter 6

Conclusions

6.1 Thesis summary and conclusions

The investigations described in this thesis result in a method for the automatic determination of sub-word units for use in automatic speech recognition. In particular, solutions to the following questions were presented:

- how many SWUs should the unit inventory have?
- how should the acoustic data be segmented to give the data for each SWU, and clustered to allow modelling of each SWU?
- what type of acoustic model should be used, and how can the acoustic models and the unit inventory size be jointly optimised?
- how are the SWUs related to words, or how are words pronounced in terms of the SWUs?

All solutions are automatic and data-driven.

6.1.1 Dictionary generation

The method presented for automatic dictionary generation involves joint multigrams. Joint multigrams (JMs) are co-segmentations of sequences of data from two data streams. Co-segmentations receive a probability according to how often they occur in the data. This framework can be directly applied to streams of words and SWUs, with the product being a probabilistic dictionary. There is no dependence in using this method on linguistic knowledge, including word boundaries. No initial dictionary is required.

Phone based dictionaries

Experiments using JM dictionary generation to automatically discover phonetic pronunciations from hand transcribed data confirm that JMs expressed as a probabilistic dictionary can be useful for ASR. In these experiments (see Section 4.4), a JM-based dictionary, generated fully automatically, achieved a lower WER than a canonical dictionary baseline (6.19%, improving on 9.63%). These results were also compared to dictionaries generated from the data in a supervised manner, and the JM based dictionary also outperformed these (the lowest WER achieved by a supervised dictionary was 6.88%).

The WER results of this phone-based experiment confirm again what has been concluded from pronunciation variation research: as the number of pronunciations in the dictionary increases, the accuracy of recognition decreases. This is easily seen in the graphs of Figure 4.5 on page 81.

Impact of parameter v on dictionary generation

A full set of joint multigrams is initialised by finding all co-segmentations of two data streams. In practice, this search must be constrained in some way, since the number of all pairings of all possible segmentations of two streams gets very large very quickly as the length of the streams increases (see Equation 4.5 on page 66). Instead, the maximum number of units in a sub-sequence of each of the streams is externally defined prior to JM initialisation, to make the search feasible. In experiments reported in this thesis, the word stream is always segmented into individual words, i.e. the maximum length of sub-sequences of words is 1. The maximum length of sub-sequences of units is a variable, v. The value of v can have a significant impact on the dictionaries generated. If v is too small, co-segmentations may be impossible for some utterances: precisely if the product of v and the number of words is less than the number of units in an utterance (see page 68). This can lead to words in the dictionary receiving no pronunciation, and rendering the dictionary useless. This is not encountered using phonetic data, with $v = 10$. However, for many of the SWU inventories generated, 10 or even 15 was not enough to ensure coverage. If v is too large, the computational cost in terms of time and memory requirements can make the process infeasible or impossible. Experiments in this thesis use a maximal value of $v = 15$ due to the computational constraint.

6.1.2 Automatically generated SWUs

A sub-word unit generation method was presented in which an ergodic HMM is trained using available training data, and the individual states of the HMM are defined to be the SWUs representing the data. This method is simple, data-driven and automatic. The process requires no information other than the parameterised acoustic data, and initial HMM topology (number of states, and number of mixture components per state). It is desirable that the number and complexity of SWUs is derived automatically. For this, the Bayesian information criterion (BIC) is employed. BIC scores are calculated for trained ergodic HMMs with different numbers of states and various numbers of mixture components per state, and the model(s) achieving maximal BIC scores are assumed to be optimally modelling the data.

The number of HMM states per sub-word unit was either 1 or 3. To derive SWUs with 1 state per unit, a straightforward erogdic HMM was trained. To derive SWUs with 3 states per units, some of the transitions of an ergodic HMM are disallowed, leading to an HMM topology as shown simply in Figure 3.8 on page 47. This variation was introduced to mimic the typical topology used for phone modelling: one 3-state left-to-right HMM for each phone.

SWU inventories derived in this way cover the data with a near-uniform distribution: no single unit models a large proportion of the data, and all units model some data.

6.1.3 Combining the two methods

The combination of the two methods provides a SWU inventory, model set, transcription of the data in terms of SWUs, and a probabilistic dictionary relating words to

SWUs. The dictionary generated for a particular SWU inventory contains entries for all possible co-segmentation of training utterances, and must be reduced to be useful. Reduction is carried out according to one of three thresholds, n, p or s as listed in Table 4.6 on page 72.

The importance of careful silence modelling

Recognition results following this combination of procedures were initially poor, with best WERs of 19.81% and 24.66% on validation data for the 1state and 3state models respectively. The reason for this unexpectedly poor outcome was hypothesised to be due to the indiscriminant treatment of silence by the ergodic HMM. It was noticed that the states of the HMM used to model silence were also used to model speech in most cases, whereas it is expected that less overlap is necessary for good modelling of either acoustic space. Further, the pronunciations generated for the pause word in the automatically generated dictionaries do not closely reflect the units that model pause frequently.

Training two models, one for speech (ergodic HMM as before) and one for silence (3-state model, as for phone-based experiments) did not improve on these results. The models were trained using embedded training, and it was seen that states of the ergodic model were still being used to model regions of silence, so the distinction between speech and silence models was still not clear.

Finally, to ensure that the ergodic HMM (and thus the SWUs) is modelling only speech, the speech portion of each utterance was cut out of the waveform, and only that data used to train the ergodic HMM and dictionary. A pause model trained for the phone-based experiment was added to the SWU inventory in order to carry out recognition. This procedure achieved a much improved WER of 11.18% (on test data) for single state SWUs.

This result shows that an important pre-requisite in generating SWUs automatically using an erogdic HMM is some silence detection, such that the SWUs focus on modelling speech. This requirement is implicit in most of the existing SWU generation methods, since word boundaries are used in the procedures, thus allowing the modelling to ignore silence. The system of Singh et al. (2002) does not require word boundaries, but no mention of the treatment of silence is included in the paper.

Potential system weaknesses

The combination of the two methods does not yet outperform phones (11.18% WER compared to 9.63% for phones on test data). Some possible reasons for this follow:

- The search for units and pronunciations is unconstrained. Perhaps some tighter constraints to limit the number of pronunciations per word are necessary to achieve better pronunciation modelling. Examples of such constraints are seen in the procedure of Bacchiani (1999).
- Word sequences are not included in the acoustic model training. Word sequences are of course vital in the dictionary generation phase; not using them as part of the SWU generation is a loss of a potential information source. Including this information into the SWU generation is not necessarily straightforward, however.
- The constraint on the maximum number of units per word used in dictionary generation (parameter v) is not included in the search for SWUs. This mismatch means that in many cases, dictionaries generated are missing words, due to the joint multigrams process being unable to use utterances containing those words, for the specific reasons noted in Section 4.3.3. This is discussed in Section 5.2.1.

6.2 Discussion

Returning to the higher level research aims, we now consider what new understanding has been obtained through this thesis.

Firstly, **how feasible is it to do ASR in this way?** This thesis shows that it is possible to do ASR based on automatically derived sub-word units. A full system can be trained and used for recognition based on automatically derived units. The task chosen to demonstrate this is a small vocabulary continuous speech corpus. Is it feasible to do ASR using this method on large vocabulary tasks? It is not possible to assess from these experiments how feasible large vocabulary ASR might be using this system design. There is nothing theoretically limiting in the processes to mean that large vocabulary ASR is impossible. Unit derivation using ergodic HMMs needs careful initialisation as the number of states increases so that each state becomes a unit modelling some proportion of the data. This is possible just by taking data points as the initial means for each state.[1] Training of the ergodic HMM takes more time as the number of states increases, as may be necessary for data with more variety, but the extra time required is not restrictive. The dictionary generation process requires a large amount of disk space, increasing as the number of words in the vocabulary increases. While JMs are not theoretically limited by the amount of data in either stream, in practical terms this may be a constraining issue in changing the task to large vocabulary.

How well does it work? As we have seen, this automatically derived unit inventory and dictionary do not achieve an improved word recognition rate compared to a standard phone-based system. However, the initial data requirements of the ASWU system are fewer and involve a lot less knowledge. A phone-based system includes a large amount of acquired linguistic knowledge in the pronunciation dictionary and/or phonetic transcriptions. The approach taken in this thesis requires only a word transcription along with the acoustic data, and correctly recognises 4 out of 5 words using only this information. When including silence detection, 9 out of 10 words are correctly recognised. What can we learn about phone-based systems from this? What does linguistic knowledge add to the ASR system? There are two factors that are present in the phone-based system which are lacking in the SWU system which are likely to account for the difference in performance between the two.

1. Word identity is used during training in a phone-based system, not just phone identity.

   In a phone-based system, there is discrimination between words from the outset: the dictionary is known to contain different pronunciations for words that are different. The acoustic model training should reflect this, such that different words are generated by different sequences of acoustic models. In this ASWU generation process there is no equivalent objective explicit in the training: nothing about the way the units are found guarantee discrimination between words. Given that this is fundamental in speech recognition, it is interesting that without using word

[1]See Section 3.5.3 on page 48 for comments on ergodic HMM initialisation.

information it is possible to correctly recognise nearly 9 out of 10 words.

2. Co-incidence of word bounds and sub-word unit bounds is ensured in a phone-based system, by design.
   As well as the explicit discrimination between words that is in a phone-based system, there is also a (implicit or explicit) knowledge of word boundaries. The existance of a dictionary prior to training implicitly defines where word boundaries are. Phonetic transcriptions or pronunciations also ensure that the sub-word unit boundaries coincide with word boundaries. Again, there is no equivalent feature in the automatic unit derivation process.

It is important to note that these factors do not relate to phones, but to words. It is not possible to conclude that the sub-word unit choice of phones is the significant factor in the system performance. It is likely that a constraint disallowing identical pronunciations for different words, and the use of information about word boundaries would result in better recognition rates in the automatic system. If this were tested by including word boundaries into the SWU generation process, and the rates were to increase, we could conclude that the knowledge of word bounds is significant to the performance of phone-based systems. This is different to concluding that the use of phones is the best choice for ASR.

Many of the **strengths and weaknesses** of using automatically derived sub-word units have been discussed already. To summarise, the main strengths of the approach of this thesis are

- The process is data driven, allowing different data to be represented by different SWU inventories easily
- There are few pre-processing or initialisation requirements
- It is portable to new languages easily, without requiring linguistic knowledge in the new language

The main weaknesses are

- The cost of dictionary generation in terms of time and compute power

- Words which are seen infrequently in training are poorly recognised (see Section 5.2.2)

6.3 Directions for future work

- Match the constraint on the number of units in a pronunciation, v, with the SWU generation process, to avoid having to reject utterances in dictionary generation as described in Section 4.3.3. This could be achieved using a duration constraint on the HMM states (minimum duration) which is calculated on-the-fly according to the length of a given utterance and the number of words it has.

- Investigate the use of these automatically determined units for other tasks. The requisites of only acoustic data and word transcriptions (with some silence detection / silence boundaries) mean that the processes developed here can easily be ported to other domains and languages, without the need for linguistic knowledge of the target language, or detailed transcriptions. Experiments testing these methods on data in other languages is an interesting avenue for future work.

- Allow inclusion of multi-words in dictionary. Strik & Cucchiarini (1999) notes that modelling within-word and cross-word variation is likely to be necessary to improve recognition of spontaneous speech. The pronunciation modelling investigation of Kessens et al. (1999) and Sloboda & Waibel (1996) indicate this outcome. Allowing multi-words is a way of introducing a slightly higher linguistic structure to the pronunciation modelling. The multigrams framework is easily adapted to allow multi-words: all that is required is the relaxation of the constraint that the word sequence is always segmented into individual words. In deMarcken's use of multigrams for dictionary generation (de Marcken (1996)), word sequences were allowed, and multi-word entries were automatically discovered, such as 'national football league' and 'goldman sacks', which are likely to be spoken 'as one word'. This amendment to the process is unlikely to yield much gain on the OGI numbers task, since the data just contains 30 words, without many common word sequences. However, on a large vocabulary task, the inclusion of multi-words is likely to have a significant impact on performance, due to the high frequency of certain phrases in natural speech.

- Allow longer pronunciations in the dictionary. The dictionaries used for the initial experiments, with 1state HMMs per 'BIC-unit' were generated using between 27% and 53% of the training data. Other utterances could not be used due to the effect of the constraint on unit sequence length v. (See Section 5.2.1.) Since at best only half of the training data was used, there will is likely to be a discrepancy between the dictionary and the acoustic models. Obviously there is a realignment so the transcription matches the dictionary, but some of the potential modelling power of the unconstrained unit derivation is likely to have been lost. Extending the maximal allowed pronunciation length was not carried out in this thesis due to computational constraints. The memory requirements in the initialisation of joint multigrams in my implementation tended to exceed availability when the constraint on the unit sequence lengths was relaxed. With more efficient coding and storage, these experiments would be possible: the constraints are not inherent within the joint multigrams framework.

- Investigate the use of multigrams for predicting pronunciations in terms of SWUs for words not present in the training data. The multigrams framework has been successfully used for grapheme to phoneme conversion in Bisani & Ney (2002, 2003). Theoretically the multigrams framework allows this directly: alongside the modelling of the pronunciation of words by SWUs using joint multigrams, pronunciations of letters and groups of letters could be modelled. So the JM dictionary generation process would carry out two parallel tasks, one for learning frequently occurring sequences for whole words, and the other for learning frequently occurring sequences for letters.

- Investigate reducing the computational cost of JM training. The largest expense is in the initialisation of the dictionary, where every co-sequence of words and pronunciations is collected and counted. This process gives the initial probabilities (normalised counts) for each word-pronunciation pairing. Before these initial counts exist, it is not possible to prune pronunciations, since there is no criteria for doing so; every co-sequence seen in the training data must be stored somewhere. In these experiments with OGI Numbers data it became necessary to use hard disk space along with virtual memory for storage, as the memory requirements became too large. The initialisation process then must repeatedly access these stored data, costing more time. As compute power continues to increase, these problems will decrease. However, a more efficient representation of the word and unit pairs would aid the reduction of the computational cost of training.

Bibliography

Adami, A. G. & Hermansky, H. (2003), Segmentation of speech for speaker and language recognition, *in* ‘Eurospeech 2003, Geneva’.

Bacchiani, M. (1999), Speech recognition system design based on automatically derived units, PhD thesis, Boston University.

Bacchiani, M. & Ostendorf, M. (1999), ‘Joint lexicon, acoustic unit inventory and model design’, *Speech Communication* pp. 99–114.

Bahl, L. R., Bellegarda, J. F., do Souza, P. V., Gopalakrishnan, P. S., Nahamoo, D. & Picheny, M. A. (1993), Multonic markov word models for large vocabulary continuous speech recognition, *in* ‘IEEE Transactions on Speech and Audio Processing’, Vol. 1.

Bahl, L. R., Brown, P. F., do Souza, P. V., Mercer, R. L. & Picheny, M. A. (1993), A method for the construction of acoustic markov models for words, *in* ‘IEEE Transactions on Speech and Audio Processing’, Vol. 1.

Ben, M., Bester, M., Bimbot, F. & Gravier, G. (2004), Speaker diarization using bottom-up clustering based on a parameter-derived distance between adapted gmms, *in* ‘ICSLP’.

Bimbot, F., Pieraccini, R., Levin, E. & Atal, B. (1995), Variable-length sequence modeling: multigrams, *in* ‘Signal Processing Letters’.

Bisani, M. & Ney, H. (2002), Investigations on joint-multigram models for grapheme-to-phoneme conversion, *in* ‘ICASSP’, Vol. 1, Denver, CO, USA, pp. 105–108.

Bisani, M. & Ney, H. (2003), Multigram-based grapheme-to-phoneme conversion for lvcsr, *in* ‘Eurospeech 2003, Geneva’.

Cernocky, J. (1998), Speech processing using automatically derived segmental units: Applications to very low rate coding and speaker verification, PhD thesis, University of Paris.

Chen, S. S. & Gopalakrishnan, P. S. (1998*a*), Clustering via the bayesian information criterion with applications in speech recognition, *in* 'ICASSP'.

Chen, S. S. & Gopalakrishnan, P. S. (1998*b*), Speaker, environment and channel change detection and clustering via the bayesian information criterion, *in* 'DARPA Broadcast News Transcription and Understanding Workshop'.

Couvreur, L. & Boite, J.-M. (1999), Speaker tracking in broadcast audio material in the framework of the thisl project, *in* 'European Speech Communication Association (ESCA) European Tutorial and Research Workshop (ETRW) on Accessing Information in Spoken Audio, Cambridge, UK', pp. 84–89.

de Marcken, C. G. (1996), Unsupervised Language Acquisition, PhD thesis, MIT.

Deligne, S. & Bimbot, F. (1995), Language modeling by variable length sequences: Theoretical formulation and evaluation of multigrams, *in* 'ICASSP', Vol. 1, Detroit, MI, USA, pp. 169–172.

Deligne, S., Yvon, F. & Bimbot, F. (1995), Variable-length sequence matching for phonetic transcription using joint multigrams, *in* 'EuroSpeech', Vol. 3, Madrid, Spain, pp. 169–172.

Digalakis, V., Rohlicek, J. R. & Ostendorf, M. (1991), A dynamical system approach to continuous speech recognition, *in* 'ICASSP'.

Fosler-Lussier, J. E. (1999), Dynamic pronunciation models for automatic speech recognition, PhD thesis, ICSI, Berkeley.

Frankel, J. (2003), Linear dynamic models for automatic speech recognition, PhD thesis, CSTR, University of Edinburgh.

Fukada, T., Bacchiani, M., Paliwal, K. K. & Sagisaka, Y. (1996), Speech recognition based on acoustically derived segment units, *in* 'ICSLP'.

Fukada, T., Yoshimura, T. & Sagisaka, Y. (1998), Automatic generation of multiple pronunciations based on neural networks and language statistics, *in* 'Workshop Modeling Pronunciation Variation for Automatic Speech Recognition', Kerkrade.

Hain, T. (2002), Implicit pronunciation modeling in asr, *in* 'ISCA Tutorial and Research workshop on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology', Estes Park, Colorado.

Heine, H., Evermann, G. & Jost, U. (1998), An hmm-based probabilistic lexicon, *in* 'Workshop Modeling Pronunciation Variation for Automatic Speech Recognition', Kerkrade.

Hersch, M. (2003), Data-driven speaker and subword unit clustering in speech processing, Master's thesis, ICSI.

Hieronymous, J. (1993), 'Ascii phonetic symbols for the world's languages: Worldbet', *Journal of the International Phonetic Association* .

Holter, T. & Svendsen, T. (1997), Incorporating linguistic knowledge and automatic baseform generation in acoustic subword unit based speech recognition, *in* 'Eurospeech'.

Jurafsky, D. & Martin, J. H. (2000), *Speech and language processing*, Prentice Hall.

Kannan, A. & Ostendorf, M. (1998), A comparison of constrained trajectory segment models for large vocabulary speech recognition, *in* 'IEEE Transactions on Speech and Audio Processing', Vol. 6.

Kessens, J. M., Wester, M. & Strik, H. (1999), 'Improving the performance of a dutch scr by modeling within-word and cross-word pronunciation variation', *Speech Communication* **29**, 193–207.

Ljolje, A. & Riley, M. D. (1991), Automatic segmentation and labeling of speech, *in* 'ICASSP'.

Manning, C. D. & Schutze, H. (2001), *Foundations of statistical natural language processing*, MIT Press.

Mariethoz, J. & Bengio, S. (2004), A new speech recognition baseline system for numbers 95 version 1.3 based on torch, IDIAP Research Report IDIAP-RR 04-16, IDIAP.

Ostendorf, M. (1999), Moving beyond the 'beads-on-a-string' model of speech, *in* 'IEEE ASRU Workshop'.

Ostendorf, M. & Digalakis, V. (1991), The stocastic segment model for continuous speech recognition, *in* 'IEEE', pp. 964–968.

Ostendorf, M., Digalakis, V. & Kimball, O. A. (1996), 'From hmms to segment models: A unified view of stochastic modeling for speech recognition', (EDICS 1.6.3).

Paliwal, K. (1990), Lexicon building methods for an acoustic sub-word based speech recognizer, *in* 'ICASSP', Vol. 2, pp. 729–732.

Pellom, B. L. & Hansen, J. H. L. (1998), 'Automatic segmentation of speech recorded in unknown noisy channel characteristics', *Speech Communication* .

Petek, B., Andersen, O. & Dalsgaard, P. (1996), On the robust automatic segmentation of spontaneous speech, *in* 'ICSLP', Vol. 2, pp. 913–916.

Rabiner, L. & Juang, B.-H. (1993), *Fundamentals of speech processing*, Prentice Hall.

Riley, M. D. (1991), A statistical model for generating pronunciations networks, *in* 'ICASSP'.

Riley, M. & Ljolje, A. (1995), 'Automatic generation of detailed pronunciation lexicons'.

Schwarz, G. (1978), 'Estimating the dimension of a model', *The Annals of Statistics* **6**(2), 461–464.

Sharma, M. & Mammone, R. (1996), "blind" speech segmentation: automatic segmentation of speech without linguistic knowledge, *in* 'ICSLP'.

Singh, R., Raj, B. & Stern, R. M. (2000), Automatic generation of phone sets and lexical transcriptions, *in* 'ICASSP'.

Singh, R., Raj, B. & Stern, R. M. (2002), 'Automatic generation of subword units for speech recognition', *IEEE Transactions on Speech and Audio Processing* **10**(2).

Sloboda, T. (1995), Dictionary learning: performance through consistency, *in* 'ICASSP'.

Sloboda, T. & Waibel, A. (1996), Dictionary learning for spontaneous speech recognition, *in* 'ICSLP', Vol. 4, Philadelphia, PA, USA, pp. 2328–2331.

Strik, H. & Cucchiarini, C. (1999), 'Modeling pronunciation variation for asr: A survey of the literature', *Speech Communication* **29**, 225–246.

Svendsen, T., Paliwal, K. K., Harborg, E. & Husoy, P. O. (1989), An improved sub-word based speech recognizer, *in* 'ICASSP', pp. 108–111.

Svendsen, T. & Soong, F. K. (1987), On the automatic segmentation of speech signals, *in* 'ICASSP', Vol. 1, pp. 77–80.

ten Bosch, L. & Cremelie, N. (2002), Pronunciation modeling and lexical adaptation using small training sets, *in* 'ISCA Tutorial and Research workshop on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology', Estes Park, Colorado.

Westendorf, C.-M. & Jelitto, J. (1996), Learning pronunciation dictionary from speech data, *in* 'ICSLP', Vol. 2, Philadelphia, PA, USA, pp. 1045–1048.

Wester, M. (2002), Pronunciation variation modeling for dutch automatic speech recognition, PhD thesis, Nijmegen University.

Wester, M. (2003), 'Pronunciation modeling for asr - knowledge-based and data-derived methods.', *Computer Speech and Language* **17**, 69–85.

Wester, M., Kessens, J. M. & Strik, H. (1998), 'Improving the performance of a dutch csr by modelling pronunciation variaion'.

Yang, Q., Martens, J.-P., Ghesquiere, P.-J. & van Compernolle, D. (2002), Pronunciation variation modeling for asr: large improvements are possible but small ones are likely to achieve, *in* 'ISCA Tutorial and Research workshop on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology', Estes Park, Colorado.