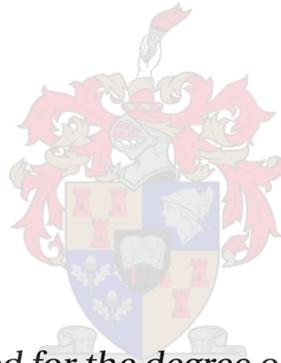


**Automatic sub-word unit discovery and  
pronunciation lexicon induction for automatic  
speech recognition with application to  
under-resourced languages**

by

Wiehan Agenbag



*Dissertation presented for the degree of Doctor of Philosophy  
in Electronic Engineering in the Faculty of Engineering at  
Stellenbosch University*

Supervisor: Dr. T.R. Niesler

March 2020

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

# Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: ..... March 2020 .....

Copyright © 2020 Stellenbosch University  
All rights reserved.

# Abstract

## **Automatic sub-word unit discovery and pronunciation lexicon induction for automatic speech recognition with application to under-resourced languages**

W. Agenbag

*Department of Electrical and Electronic Engineering,  
University of Stellenbosch,  
Private Bag X1, Matieland 7602, South Africa.*

Dissertation: PhD Eng

March 2020

Automatic speech recognition is an increasingly important mode of human-computer interaction. However, its implementation requires a sub-word unit inventory to be designed and an associated pronunciation lexicon to be crafted, a process that requires linguistic expertise. This step represents a significant bottleneck for most of the world's under-resourced languages, for which such resources are not available. We address this challenge by developing techniques to automate both the discovery of sub-word units and the induction of corresponding pronunciation lexica. Our first attempts at sub-word unit discovery made use of a shift and scale invariant convolutional sparse coding and dictionary learning framework. After initial investigations showed that this model exhibits significant temporal overlap between units, the sparse codes were constrained to prohibit overlap and the sparse coding basis functions further globally optimised using a metaheuristic search procedure. The result was a unit inventory with a strong correspondence with reference phonemes, but highly variable associated transcriptions. To reduce transcription variability,

two

lattice-constrained Viterbi training strategies were developed. These involved jointly training either a bigram sub-word unit language model or a unique pronunciation model for each word type along with the unit inventory. By taking this direction, it was necessary to abandon sparse coding in favour of a more conventional HMM-GMM approach. However, the resulting strategies yielded inventories with a higher degree of correspondence with reference phonemes, and led to more consistent transcriptions. The strategies were further refined by introducing a novel sub-word unit discovery approach based on self-organising HMM-GMM states that incorporate orthographic knowledge during sub-word unit discovery. Furthermore, a more sophisticated pronunciation modeling approach and a two-stage pruning process was introduced. We demonstrate that the proposed methods are able to discover sub-word units and associated lexicons that perform as well as expert systems in terms of automatic speech recognition performance for Acholi, and close to this level for Ugandan English. The worst performing language among those evaluated was Luganda, which has a highly agglutinating vocabulary that was observed to make automatic lexicon induction challenging. As a final step, we addressed this by introducing a data-driven morphological segmentation step that is applied before performing lexicon induction. This is demonstrated to close the gap with the expert lexicon for Luganda. The techniques developed in this thesis demonstrate that it is possible to develop an automatic speech recognition system in an under-resourced setting using an automatically induced lexicon without sacrificing performance, even in the case of a highly agglutinating language.

# Uittreksel

## **Outomatiese ontdekking van sub-woord eenhede en uitspraakwoordeboekinduksie vir outomatiese spraakherkenning met toepassing op tale wat beperkte hulpbronne het**

*(“Automatic sub-word unit discovery and pronunciation lexicon induction for automatic speech recognition with application to under-resourced languages”)*

W. Agenbag

*Departement Elektriese en Elektroniese Ingenieurswese,*

*Universiteit van Stellenbosch,*

*Privaatsak X1, Matieland 7602, Suid Afrika.*

Proefskrif: PhD Ing

Maart 2020

Outomatiese spraakherkenning is ’n toenemend belangrike manier van interaksie tussen mens en rekenaar. Die implementering daarvan vereis egter dat ’n inventaris van subwoordeenhede ontwerp word en dat daar ’n gepaardgaande uitspraakleksikon geskep moet word, ’n proses wat taalkundige deskundigheid vereis. Hierdie stap is ’n belangrike bottelnek vir die meeste van die wêreld se hulpbron-beperkte tale, waarvoor sulke hulpbronne nie beskikbaar is nie. Ons pak hierdie uitdaging aan deur tegnieke te ontwikkel om sowel die ontdekking van subwoordeenhede as die induksie van ooreenstemmende uitspraakleksikons te outomatiseer. Ons eerste pogings tot die ontdekking van subwoordeenhede het gebruik gemaak van ’n skuif- en skaalinvariante konvolusionêre ylkodering- en woordeboekleerraamwerk. Nadat aanvanklike ondersoeke getoon het dat hierdie model ’n beduidende temporale oorvleueling tussen een-

hede tot gevolg het, is die ylkodes beperk om oorvleueling te verbied, en die ylkoderingsbasisfunksies verder globaal geoptimeer met behulp van 'n meta-heuristiese soekprosedure. Die resultaat was 'n eenheidsinventaries wat 'n sterk ooreenstemming met verwysingsfoneme toon, maar met hoogs wisselvallige gepaardgaande transkripsies. Om die transkripsiewisselvalligheid te verminder, is twee tralie-beperkte Viterbi-opleidingstrategieë ontwikkel. Dit behels gesamentlike opleiding van óf 'n bigram-subwoordeenheidstaalmodel óf 'n unieke uitspraakmodel vir elke woordsoort, tesame met die eenheidsinventaris. Deur hierdie rigting in te neem, was dit nodig om die ylkodering te laat vaar ten gunste van 'n meer konvensionele HMM-GMM benadering. Die gevolglike strategieë het egter subwoordeenheidinventarisse gelewer met 'n hoër mate van korrespondensie met verwysingsfoneme, en het gelei tot meer konsekwente transkripsies. Die strategieë is verder verfyn deur 'n nuwe benadering tot die ontdekking van subwoordeenhede in te stel, gebaseer op selforganiserende HMM-GMM toestande wat ortografiese kennis insluit tydens die ontdekking van subwoordeenhede. Verder is 'n meer gesofistikeerde benadering tot uitspraakmodelering en 'n twee-fase snoeiproses ingestel. Ons demonstreer dat die voorgestelde metodes subwoordeenhede en gepaardgaande leksikons kan ontdek wat so goed presteer soos stelsels ontwerp deur deskundiges in terme van outomatiese spraakherkenning vir Acholi, en naby aan hierdie vlak vir Ugandese Engels. Die taal wat die swakste gevaar het onder die wat beoordeel was, was Luganda, wat 'n uiters agglutinerende woordeskate het wat waargeneem word om outomatiese leksikon-induksie uitdagend te maak. As 'n laaste stap het ons dit aangespreek deur 'n data-gedrewe morfologiese segmenteringsstap in te stel wat toegepas word voordat leksikon-induksie uitgevoer word. Dit word getoon om die gaping met die deskundige leksikon vir Luganda te sluit. Die tegnieke wat in hierdie proefskrif ontwikkel is, demonstreer dat dit moontlik is om 'n outomatiese spraakherkenningstelsel te ontwikkel in 'n hulpbron-beperkte omgewing met behulp van 'n outomaties geïnduseerde leksikon, sonder om prestasie in te boet, selfs in die geval van 'n uiters agglutinerende taal.

# Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Thomas Niesler, for all his advice, support, and encouragement, which kept me motivated and on track. I would also like to thank all DSP lab dwellers, and especially Ewald van der Westhuizen—for being so generous with his time and knowledge. Lastly, I want to acknowledge and thank the National Research Foundation and the Department of Arts and Culture for their support.

# Contents

<b>Declaration</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Uittreksel</b>	<b>iv</b>
<b>Acknowledgements</b>	<b>vi</b>
<b>Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xiii</b>
<b>Nomenclature</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Objectives of this study . . . . .	2
1.3 Research contributions . . . . .	3
1.4 Literature review . . . . .	3
1.5 Thesis overview . . . . .	10
<b>2 Automatic segmentation and clustering of speech using sparse coding</b>	<b>12</b>
2.1 Introduction . . . . .	12
2.2 Background . . . . .	13
2.3 Sub-word unit discovery with Coordinate Descent . . . . .	25
2.4 Experiments and results . . . . .	27

<i>CONTENTS</i>	<b>viii</b>
2.5 Conclusion . . . . .	36
<b>3 Non-overlapping sparse codes and metaheuristic search</b>	<b>37</b>
3.1 Introduction . . . . .	37
3.2 Background . . . . .	38
3.3 Implementation . . . . .	39
3.4 Experiments and results . . . . .	44
3.5 Summary and conclusion . . . . .	50
<b>4 Refining SWUs with lattice-constrained Viterbi training</b>	<b>52</b>
4.1 Introduction . . . . .	52
4.2 Motivation . . . . .	52
4.3 Acoustic modeling with HMM-GMMs . . . . .	53
4.4 Lattice-constrained Viterbi training . . . . .	58
4.5 Pronunciation modeling . . . . .	59
4.6 Evaluating the SWU inventory . . . . .	62
4.7 Results . . . . .	64
4.8 Automatic speech recognition . . . . .	65
4.9 Summary and conclusion . . . . .	70
<b>5 Lexicon induction for under-resourced languages</b>	<b>73</b>
5.1 Introduction . . . . .	73
5.2 Proposed approach to SWU discovery and lexicon induction . . . . .	74
5.3 Experiments and results . . . . .	87
5.4 Summary and conclusions . . . . .	100
<b>6 Lexicon induction for highly-agglutinating languages</b>	<b>102</b>
6.1 Introduction . . . . .	102
6.2 Background . . . . .	103
6.3 Experiments and results . . . . .	105
6.4 Summary and conclusions . . . . .	109
<b>7 Conclusion</b>	<b>111</b>
7.1 Summary . . . . .	111
7.2 Summary of contributions . . . . .	115

<i>CONTENTS</i>	<b>ix</b>
7.3 Future research . . . . .	117
<b>List of References</b>	<b>119</b>

# List of Figures

21	Differentiable diversity measures for the optimisation of sparse codes.	15
22	Overview of sparse code and dictionary learning with an initial code.	25
23	Overview of the dictionary update process.	27
24	Development of the reconstruction error, code sparsity and cost function over 20 iterations of training for $\beta = 9$ , $N_\phi = 20$ and $D = 50$ .	29
25	Final normalised reconstruction error vs. number of base atoms for various values of $\beta$ . $N_\phi = 20$ .	30
26	Number of coefficients used vs. number of base atoms for various values of $\beta$ . $N_\phi = 20$ . For comparison, the dataset contains 56377 reference phoneme instances.	31
27	Mel spectrogram representation of base atom dictionaries after training. Lower frequencies are at the bottom.	32
28	Coincidence matrix for $\beta = 10$ , $N_\phi = 20$ and $D = 60$ .	33
29	Normalised overlap scores for various training parameters.	34
210	Average consistency scores on the SA dataset with atoms learned in previous experiments.	35
31	Number of coefficients used and normalised reconstruction error for the elite individual for various values of $\beta$ and $D$ . For comparison, the reference transcription contains 56377 phoneme instances.	44
32	Generational development of the population fitness distribution for an experiment with $\beta = 8.3$ and $D = 55$ .	46
33	Terminal fitness distributions for an experiment with $\beta = 10.6$ and $D = 75$ .	47
34	Coincidence of learned basis functions with reference phonemes for $\beta = 10.6$ and $D = 50$ .	48

35	Mel spectrogram representation of the elite dictionary of sub-word units for $\beta = 10.6$ and $D = 50$ . Lower frequencies are at the bottom. . . .	49
36	Weighted average of the fraction of occurrences of the 20 most frequent words transcribed by one of their respective top 3 pronunciations. TIMIT's reference transcription achieves 0.69. . . . .	49
41	Three-state HMM used for acoustic modeling. . . . .	53
42	Topology of silence (sil) and short pause (sp) acoustic models [1]. . . .	55
43	Topology of composite phone-loop HMM. . . . .	56
44	Single-state word HMM. . . . .	59
45	Word pronunciation lattice model used to constrain SWU transcription.	61
46	Comparison of phoneme-SWU coincidence matrices and their corresponding entropic coding efficiencies for experiments 1, 2, and 3 in Table 53. The dashed lines show the weighted mean coding efficiencies. . . . .	72
51	Overview of the proposed SWU discovery and lexicon induction process, as described in Section 5.2. . . . .	74
52	Overview of the initial SWU and lexicon discovery procedure, as described in Section 5.2.2. . . . .	76
53	Overview of steps performed during full lexicon extraction, as described in Section 5.2.3. . . . .	80
54	Topology of the word pronunciation HMM. . . . .	82
55	The impact of the meta-parameters number of units ( $N_p$ ) and average unit length ( $R_p$ ) on the word error rates of the resulting ASR systems. . .	91
56	Word error rates for three consecutive lexicon extraction passes (Ugandan English). . . . .	94
57	Coincidence of automatically discovered units with reference phonemes (TIMIT). . . . .	95
61	Overview of the steps performed during lexicon induction using morphological segments. . . . .	104
71	Coincidence matrix of the SWU inventory produced by the sparse coding approach of Chapter 2. . . . .	112

72	Coincidence matrix of the SWU inventory produced by the non-overlapping sparse coding model of Chapter 3. . . . .	113
73	Coincidence matrix of the SWU inventory produced by the word level pronunciation model constrained Viterbi training introduced in Chapter 4. . . . .	114
74	Coincidence matrix of the SWU inventory produced in Chapter 5 through the agglomerative clustering of self-organising word-level HMM-GMM states. . . . .	115

## List of Tables

21	Training parameters . . . . .	28
22	Baseline consistency scores using reference phoneme transcriptions .	34
31	Improvement made in the cost function by the metaheuristic search compared to pure local search as a multiple of the search termination threshold. . . . .	46
32	Pronunciation statistics for the most consistent set of sub-word acoustic units. Statistics for TIMIT's reference transcriptions are in parentheses. . . . .	50
41	Summary of experimental results . . . . .	65
42	Overview of datasets used for ASR . . . . .	68
43	Summary of experimental results . . . . .	69
51	Summary of datasets used for experimental evaluation. #Words indicates the number of word tokens. . . . .	87
52	Summary of the vocabulary distribution and OOV rates for the three datasets introduced in Table 61. Tokens in the training set occurring more than three and nine times are indicated by $n > 3$ and $n > 9$ respectively. . . . .	88
53	Summary of ASR word error rates for baseline systems as well as systems using automatically induced sub-word units and associated lexicons. . . . .	90
54	Summary of ASR word error rates achieved when using ground truth word boundaries. . . . .	93

55	Example of the representation of the TIMIT phone k exhibiting right-hand context-dependence in the automatically induced lexicon for Ugandan English ( $N_p = 140$ , $R_p = 7.8$ ). The first two columns indicate symbols used in the TIMIT phonetic alphabet, while the last column indicates automatically-discovered SWU labels. . . . .	97
56	Example of multiple TIMIT phones (b, d, g) represented with an overlapping pool of units depending on context in the automatically induced lexicon for Ugandan English ( $N_p = 140$ , $R_p = 7.8$ ). The first two columns indicate symbols used in the TIMIT phonetic alphabet, while the last column indicates automatically-discovered SWU labels. . . . .	98
57	Sample of frequent n-grams found in the automatically induced lexicon ( $N_p = 50$ , $R_p = 7.8$ ) for Ugandan English. . . . .	99
61	Summary of the Luganda dataset used for experimental evaluation. . . . .	105
62	Vocabulary distributions for various levels of morphological segmentation (expressed as the average segment length in graphemes), and the associated ASR performance for each system. . . . .	106
63	The ASR performance of lexicons induced using context-dependant morphological segmentation for various pooling thresholds. A threshold of $\infty$ indicates context independent segments. . . . .	108
64	Summary of the best ASR performance for each of the various systems evaluated in this study. . . . .	109

# Nomenclature

## Acronyms and Abbreviations

AM	Acoustic model
ASR	Automatic speech recognition
CMLLR	Constrained maximum likelihood linear regression
CTC	Connectionist temporal classification
CVN	Cepstral variance normalisation
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
DNN	Deep neural network
DP	Dynamic programming
DP	Dirichlet process
DTW	Dynamic time warping
EM	Expectation-maximization
fMLLR	Feature-space maximum likelihood linear regression
G2P	Grapheme-to-phoneme
G2SWU	Grapheme-to-sub-word unit
GMM	Gaussian mixture model
HBM	Hierarchical Bayesian model
HDP	Hierarchical Dirichlet Process
HMM	Hidden Markov model
HTK	Hidden Markov model toolkit
MFCC	Mel-frequency cepstral coefficients
MLLR	Maximum likelihood linear regression

MLLT	Maximum likelihood linear transformation
MMI	Maximum mutual information
MP	Matching pursuit
OMP	Orthogonal matching pursuit
OOV	Out-of-vocabulary
RNN	Recurrent neural network
SAT	Speaker adaptive training
SGMM	Subspace Gaussian mixture modelling
SISC	Shift-invariant sparse coding
SWU	Sub-word unit
SVM	Support vector machine
TDNN	Time-delay deep neural network
VB	Variational Bayesian
WER	Word error rate

# Chapter 1

## Introduction

### 1.1 Introduction

Automatic speech recognition (ASR) allows digital computing devices to transform spoken language into text form. Such technology enables a new mode of human-computer interaction, which is both more natural for a general user and enables access to visually or hearing impaired individuals. It also allows spoken language to be searched and indexed, thereby providing digital access to a resource that has so far been inaccessible. However, these benefits of automatic speech recognition have not reached all languages equally. Instead, robust implementations are available for only a few broadly spoken languages in wealthy economies, while the most of the world's languages remain excluded. This inequality is due to the extensive resources required for the development of ASR systems. A few languages, such as English, have many resources and which are beyond the reach of especially less-developed economies.

One of the key resources required by the prevailing paradigm for ASR is a lexicon that represents every word in the language's vocabulary in terms of sub-word units, typically phonemes. The transcription of a word in terms of these units is called a *pronunciation*, and the list containing the pronunciations of all the words of interest is called a *pronunciation lexicon*. The availability of such a lexicon allows acoustic modelling to be performed at a sub-word level, which significantly improves the performance of ASR systems, since it mitigates data sparsity and enables the recognition of words not seen during training, as long

as pronunciations for those words exist in the lexicon.

The preparation of these lexicons is typically performed by trained phoneticians, which implies the need for established expert consensus about the phonemic contents of a language as well the resources required to employ these experts to transcribe every word of interest. For the primary languages and dialects of the world's most developed countries, these conditions are met. However, for many other languages, the resources required to produce a high quality pronunciation lexicon are not available.

In this work, the primary objective is to automatically discover inventories of sub-word units and to infer pronunciation lexicons in terms of these units. The motivation for this is to enable the implementation of ASR for languages for which such pronunciations or even the phonetic inventory needed to construct them are not available. However, it also remains possible that the established pronunciation dictionaries and associated sub-word inventories of well-resourced languages are not optimal for speech recognition. It is hoped that as the techniques for unsupervised pronunciation lexicon discovery mature, they may eventually help advance the state of the art in ASR for many languages.

## 1.2 Objectives of this study

Given only a corpus of recorded speech and associated orthographic transcriptions (which may be graphemic or logographic), the objectives of this study are:

1. to automatically discover acoustic sub-word units that are relevant for recognising words;
2. to derive pronunciation dictionaries in terms of these sub-word units; and
3. to evaluate the performance of an ASR system trained using these sub-word units and pronunciation lexicon.

### 1.3 Research contributions

The research presented in this thesis has led to the following peer-reviewed publications:

#	Publication	Chapter
1.	Agenbag, W; Smit, W; Niesler, T.R. <i>Automatic Segmentation and Clustering of Speech using Sparse Coding</i> . Proceedings of the twenty-fifth annual symposium of the Pattern Recognition Association of South Africa (PRASA), Cape Town, South Africa, 2014.	2
2.	Agenbag, W; Niesler, T.R. <i>Automatic segmentation and clustering of speech using sparse coding and metaheuristic search</i> . Proceedings of INTERSPEECH, Dresden, Germany, 2015.	3
3.	Agenbag, W; Niesler, T.R. <i>Refining Sparse Coding Sub-word Inventories with Lattice-constrained Viterbi Training</i> . Proceedings of the 5th Workshop on Spoken Language Technology for Under-resourced Languages, SLTU 2016, 9-12 May 2016, Yogyakarta, Indonesia	4
4.	Agenbag, W; Niesler, T.R. <i>Automatic sub-word unit discovery and pronunciation lexicon induction for ASR with application to under-resourced languages</i> . Computer Speech & Language, Volume 57, 2019, Pages 20-40	5
5.	Agenbag, W; Niesler, T.R. <i>Improving automatically induced lexicons for highly agglutinating languages using data-driven morphological segmentation</i> . Proceedings of INTERSPEECH, Graz, Austria, 2019.	6

### 1.4 Literature review

The task of automatically generating a pronunciation lexicon from a word-annotated speech corpus requires addressing a number of subtasks. First, a set of sub-word units needs to be established. Much previous work on acoustics-driven automatic lexicon generation begins with a small seed lexicon and then expands the vocabulary by means of a large word-annotated speech corpus [2–5]. Alternatively, similar approaches have been used to expand an expert lexicon to include acoustically dominant pronunciation variants [6]. However, in the case where no seed lexicon is available, an inventory of sub-word units must first be derived in an unsupervised or semi-supervised fashion, or a phone set from another language must be adopted. Once such an initial inventory is established, pronunciations must be generated for each word. Due to the

acoustic variability of speech, this step often produces a number of possible candidates and therefore necessitates some form of scoring and pruning in order to achieve an acceptable number of variants per word.

### **1.4.1 Unsupervised sub-word unit discovery**

Many approaches to unsupervised sub-word unit (SWU) discovery rely on a discrete two-step process where speech is first segmented and the resulting segments are then clustered to form a compact set of sub-word units [7; 8]. Other approaches attempt to jointly derive both the segmentation and clustering. However, a crude segmentation and clustering step is often still used to initialize the joint optimization procedures [9].

#### **1.4.1.1 Speech segmentation**

Initial segmentation of speech into sub-word length units is most commonly performed by hypothesizing that segment boundaries occur at those instances in time where speech features exhibit relatively rapid change. This approach was adopted by Svendsen and Soong [10], who estimate the frame-to-frame spectral variation and perform peak-finding on this signal to determine segment boundaries. A related approach is to calculate a simple similarity score between successive frames and then insert a boundary when the score exceeds some pre-determined threshold, as proposed by Ten Bosch and Cranen [7]. These authors used the cosine distance between the averages of respectively the two preceding and the two succeeding feature frames. In order to make the segmentation more robust during silent speech portions, the score is multiplied by the log of the signal energy, prior to thresholding.

Extending this approach, a more robust segmentation can be obtained by applying a dynamic programming (DP) processing step to the sequence of similarity scores for a particular utterance [9–13]. The DP step addresses the tendency of an approach relying on a simple threshold to over-segment, by finding a segmentation that is optimal in terms of an objective that considers additional external constraints or penalty terms in addition to the frame-wise similarity scores. For example, the work by Van Vuuren et al. incorporated an explicit segment length model [13], while the work by Bacchiani and Ostendorf

constrains the segmentation in a way that ensures all instances of a word contain the same number of units [12].

#### 1.4.1.2 Clustering of segments

A variety of clustering approaches has been applied to the large pool of speech units resulting from the above segmentation strategies. These approaches are summarised in the following.

Bacchiani and Ostendorf employ a divisive approach to optimize a maximum likelihood criterion using clusters in which the segments of speech features are modeled as being generated by single-mixture GMMs [12]. Initially, all segments are assigned to the same cluster. Clusters are then repeatedly divided in such a way that the likelihood of the segments being generated by the resulting GMMs is maximised. This approach is often referred to as maximum-likelihood successive state splitting, and is also often applied to the modelling of allophonic variation. The authors constrain the division of clusters to ensure that all instances of a unit at a particular position in a word remain in the same cluster.

Varadarajan and Khudanpur also proposed the use of an HMM maximum-likelihood successive state splitting algorithm, similar to that used by Bacchiani and Ostendorf, but included several modifications and optimisations, such as permitting the algorithm to optimally leap-frog by simultaneously splitting several states at once [14].

Jansen and Church also begin with automatically segmented units initially pooled according to word context [15]. However, these authors estimate whole-word HMM-GMMs and cluster the states of these models using spectral clustering. In order to obtain a similarity measure between pairs of states, the correlation between the trajectories of the states' posterior probabilities (called a posterioqram cross correlation) is computed.

In the work by Wang et al., a posterioqram representation is also used for clustering, with the authors considering the posterior trajectories resulting from the components of a GMM [16]. The authors then derive a segment level Gaussian posterioqram by summing the posterior probability vector for all frames in the same segment. Wang et al. consider both normalized cut and non-negative matrix factorization of the matrices obtained by concatenating the

segment level posteriorgrams for all segments in the training set.

Instead of using automatically segmented units, Razavi et al. use context-dependent graphemes as their starting point, modeled using HMM-GMMs [17; 18]. In order to obtain a compact set of units, these authors perform decision tree clustering of the graphemes with singleton questions.

Finally, Goussard and Niesler as well as Lerato and Niesler have employed agglomerative hierarchical clustering, which initially assigns every segment to its own cluster, and then iteratively merges pairs of clusters according to some metric of cluster similarity [8; 19]. A dynamic time warping (DTW) based metric was used to calculate similarities between pairs of segments.

#### **1.4.1.3 Joint segmentation and clustering**

Segmentation followed by clustering can be argued to be sub-optimal, since the segmentation decisions are taken without regard for how appropriate they are for the subsequent clustering. This can be addressed by performing segmentation and clustering in a co-ordinated fashion. Such joint approaches have generally involved the estimation of a hierarchical Bayesian model (HBM), which incorporates multiple levels of problem analysis into a single coherent statistical framework. This has the advantage that model inference has to consider all subtasks simultaneously, in this case segmentation, clustering, and acoustic modeling of each cluster. However, the methods discussed below may substantially differ in their need for speech data.

Lee et al. proposed a non-parametric Bayesian model in the form of a Dirichlet process (DP) mixture model, with the segmentation, clustering, acoustic models and unit inventory designated as latent variables [20]. Dirichlet processes are commonly used in non-parametric clustering, since they can act as a prior for infinite mixture models, the use of which negates the need to specify the number of clusters in advance. In the case of Lee et al., each component drawn from the DP is a 3-state HMM representing a sub-word unit. This model was then used to jointly infer the segmentation and clustering of SWUs using a Gibbs sampler.

The work by Ondel et al. and Liu et al. used a similar approach, but modified the construction of the Dirichlet process to allow for the use of

variational Bayesian (VB) inference [21; 22]. This allows model training to be parallelized, which leads to faster convergence and the ability to process more data, compared to Gibbs sampling, but comes at the cost of potentially converging to a local optimum. However, in their experiments, Ondel et al. found the clusters resulting from VB to correspond better to the true phoneme labels than those produced by Gibbs sampling.

Torbati et al. extended these approaches by using a Hierarchical Dirichlet Process (HDP) model, which models each cluster as a DP where the base distribution (which is itself a DP) is shared among all clusters [23]. In particular, Torbati et al. constructed a HDP-HMM model, which is an HMM with an infinite number of states, with each state representing a SWU.

Lee et al., propose an HBM from which a latent set of SWUs and a corresponding grapheme-to-SWU mapping is inferred [24], also using a Dirichlet prior. Further work by Lee et al. proposed a hierarchical model that attempts the unsupervised discovery of both a latent set of SWUs and several layers of hierarchical linguistic structure using Adaptor Grammars [25], which are a non-parametric Bayesian extension of Probabilistic Context-Free Grammars.

An alternative set of approaches to jointly discovering SWU segmentations and clusters involves an iterative learning process in which a speech corpus is tokenized using a fixed set of SWU templates or models, and thereafter the templates or models are updated while the tokenization is held fixed. Siu et al. used such an approach for the training of a set of self-organising HMMs, each of which represented an individual SWU [9]. These authors constrained the tokenization with a phone-level language model, which was learned in tandem with the SWU model parameter updates.

Singh et al. proposed a probabilistic framework for the joint estimation of sub-word units, word boundary segmentation and pronunciations, while also allowing for the incorporation of external sources of information such as alphabetic graphemes [26]. This model was then simplified to allow for an iterative training process where only one component of the framework was updated at a time, with the others fixed.

### 1.4.2 Pronunciation candidate generation and pruning

When a seed lexicon and an alphabetic orthography is available, pronunciations for new words can be generated using grapheme-to-phoneme tools. Alternatively, candidates can be generated by means of a Viterbi decode pass, using acoustic models obtained from a seed lexicon [2; 6; 26; 27] or from automatically discovered units.

Once a set of candidate pronunciations has been generated, their number must be reduced to a compact set of “canonical” variants. This can be achieved through pronunciation scoring and subsequent pruning. The most straightforward of these scoring schemes simply uses the relative frequency of each candidate [3; 4; 8]. This works well for short words that occur frequently, but longer and less frequently occurring words often generate as many variants as there are occurrences, rendering pruning based on relative frequency ineffective.

Zhang et al. [2] make use of a likelihood-based pruning criterion which retains pronunciation hypotheses that are highly dissimilar. Such variants are needed for example for words with the same spelling but different pronunciations. The authors achieve this through the use of a greedy selection procedure, which iteratively removes individual pronunciation hypotheses, such that when the remaining pronunciations are scored, the reduction in per-utterance likelihood is minimal.

Alternative scoring approaches rely on graph structure representations of pronunciation variation, which enables scoring based on, for example, relative similarity to other hypotheses, even when no variant is observed more than once. Singh et al. collapse all the observed pronunciation hypotheses for each word into a graph, with the weights of each node proportional to the number of times that SWU is observed [26]. Every path through these graphs is then used to synthesise an expanded set of pronunciations. Each hypothesis is scored using the acoustic models, with the most acoustically likely candidate retained in the lexicon.

In the work by Lu et al. [5] the pronunciation hypothesis space is represented using lattices obtained from acoustic decoding. Subsequently the scores contained in these lattices are used to calculate pronunciation weights. A greedy threshold-based pruning step is used to reduce the number of hypotheses.

### 1.4.3 Sequence-to-sequence approaches to ASR

A competing approach to ASR that is gaining popularity is referred to as “sequence to sequence” modelling. These models attempt to learn a direct translation between acoustic feature sequences and sequences of linguistic tokens such as graphemes or words.

Sequence to sequence modeling is typically accomplished by systems trained using connectionist temporal classification (CTC) [28–31], which does not require the time boundaries of the target labels to be specified. This is accomplished by using a recurrent neural network (RNN) which outputs the probability of observing a label boundary (with the addition of a special symbol to indicate no boundary) at each frame, as opposed to a conventional RNN which outputs framewise label probabilities. The CTC network is trained by using the Forward-Backward algorithm to consider every possible alignment between the label sequences and the feature vectors, and then performing gradient descent. The CTC approach has been extended by the addition of a separate recurrent neural network language model [32; 33].

Alternatively, attention-based models can be used. Here an encoder network, which converts the input feature sequence into a sequence of high-dimensional representation vectors, is combined with an attention-based decoder network, which decodes the output of the encoder network into the target sequence [34–38]. The attention mechanism is used to weigh the contributions of the outputs of the encoder network in order to produce each output symbol.

Since sequence-to-sequence models forgo the need for external linguistic resources such as language and pronunciation models, they seem attractive in the under-resourced case where such linguistic resources are unavailable. However, in the absence of these linguistic resources, such systems show a large performance degradation unless a very large amount of transcribed training data is used [34; 39]. Several orders of magnitude more training data may be required than is available in the under-resourced setting, rendering these approaches currently unsuitable for such datasets.

## 1.5 Thesis overview

The remainder of this thesis is structured as follows.

In *Chapter 2* we investigate the application of sparse coding to the sub-word unit discovery task. A set of sparse coding atoms is trained with the Coordinate Descent algorithm to code a subset of the TIMIT corpus. Although some of the trained units exhibit strong correlation with specific reference phonemes, it is found that our sparse coding model does not place sufficient constraints for the activation of atoms to be temporally isolated, which rules out its direct application to speech segmentation. We also find that the sparse coding model generates codes that contain too much variation across instances of the same orthographic transcription for it to be useful for generating pronunciation dictionaries for ASR.

In *Chapter 3* we propose an adaptation of the sparse coding model that is constrained to use basis functions that do not overlap temporally. We introduce a novel local search algorithm that iteratively improves the acoustic relevance of the automatically-determined sub-word units from a random initialization. We also contribute an associated population-based metaheuristic optimisation procedure related to genetic approaches to achieve a global search for the most acoustically relevant set of sub-word units. We find that some of the automatically-determined sub-word units in our final inventories exhibit a strong correlation with the reference phonetic transcriptions. However, the resulting sub-word unit transcriptions are still too variable to be useful for deriving pronunciation lexicons for ASR.

In *Chapter 4* we investigate the application of two novel lattice-constrained Viterbi training strategies aimed at reducing SWU sequence variability and concomitantly improving the SWU inventories. The first lattice-constrained training strategy attempts to jointly learn a bigram SWU language model along with the evolving SWU inventory. We find that this substantially increases correspondence with expert-defined reference phonemes on the TIMIT dataset, but does little to improve pronunciation consistency. The second approach attempts to jointly infer an SWU pronunciation model for each word in the training vocabulary, and to constrain transcription using these models. We find that this lightly supervised approach again substantially increases

correspondence with the reference phonemes, and in this case also improves pronunciation consistency. We also perform our first attempt at constructing a pronunciation lexicon. We are able to obtain reasonable SWU pronunciations for short, frequent words, but the pronunciations inferred for longer and uncommon words remain poor. As a result, ASR experiments using these SWU pronunciation lexicons yield poor results.

In *Chapter 5* we attempt to improve the quality of the lexicons induced in Chapter 4 by using a more sophisticated pronunciation modeling and pruning approach. We also address lexicon quality by means of a novel SWU discovery approach based on self-organising HMM-GMM states that incorporate orthographic knowledge during SWU discovery. We apply our methods to corpora of recorded radio broadcasts in Ugandan English, Luganda and Acholi. We demonstrate that our proposed method is able to discover lexicons that perform as well as baseline expert systems for Acholi, and close to this level for the other two languages when used to train DNN-HMM ASR systems. The worst performance was observed for Luganda, which has a highly agglutinating vocabulary and therefore presents a particular challenge to lexicon induction.

In *Chapter 6* we present a method of improving the performance of automatically induced lexicons for highly agglutinating languages. We address the unfavorable vocabulary distribution of such languages by performing data-driven morphological segmentation of the orthography prior to lexicon induction. We apply this new approach to a corpus of recorded radio broadcasts in Luganda. The intervention leads to a 10% (relative) reduction in WER, which puts the resulting ASR performance on par with an expert lexicon. When context is added to the morphological segments prior to lexicon induction, a further 1% WER reduction is achieved. This demonstrates that it is feasible to perform ASR in an under-resourced setting using an automatically induced lexicon even in the case of a highly agglutinating language.

## Chapter 2

# Automatic segmentation and clustering of speech using sparse coding

### 2.1 Introduction

In this Chapter, we investigate the application of a mathematical framework that is known as *sparse coding and dictionary learning* to mine recorded speech for sub-word units. Sparse coding is a signal processing method that attempts to reconstruct an input signal using a linear combination of as few basis functions as possible. The signal that encodes the identity and time offset of the basis functions used to accomplish this and is known in the sparse coding literature as the *sparse code*. The basis functions are taken from an inventory known as the *dictionary*, and is a distinct concept to that of a *pronunciation lexicon*, which maps words to sequences of sub-word units.

In this study, the sparse coding basis functions will define the sub-word units that may be used in later stages to induce pronunciation lexicons, while the sparse codes themselves will act as a transcription of the speech utterances in terms of the sub-word units. Since neither the sparse codes nor the dictionary of basis functions are known in advance, they must be learned simultaneously. This joint learning of the sub-word units and their corresponding transcriptions is the primary motivation behind the choice of the sparse coding framework for

sub-word unit discovery. We suspected that, by allowing the transcriptions and sub-word units to be jointly optimised, we might extract some improvement over previous approaches that have largely relied on a sequential process of first segmenting recorded speech and then clustering the segments into sub-word units. Joint approaches to segmentation and clustering, such as the Hierarchical Bayesian Models described in Section 1.4, are difficult to train, while the sparse coding framework used in this chapter can efficiently be applied to larger datasets.

## 2.2 Background

This section gives an overview of the state of the art in the field of sparse coding, and concludes with an indication of how the methods of sparse coding have already been used in speech recognition systems.

### 2.2.1 Sparse coding

Sparse coding can be viewed as a solution to the problem

$$\operatorname{argmin}_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{such that} \quad \mathbf{y} = \mathbf{D}\mathbf{x} \quad (2.1)$$

with  $\mathbf{y} \in \mathbb{R}^{N \times 1}$ ,  $\mathbf{D} \in \mathbb{R}^{N \times M}$  and  $\mathbf{x} \in \mathbb{R}^{M \times 1}$  and where  $\|\mathbf{x}\|_0$  constitutes the  $l_0$  pseudo-norm of  $\mathbf{x}$ , i.e. the number of non-zero elements in  $\mathbf{x}$ . In other words, we are trying to reconstruct a sequence  $\mathbf{y}$  using a linear combination of as few as possible of the atomic sequences that constitute the columns of a dictionary matrix  $\mathbf{D}$ . The weights of the linear combination are given in  $\mathbf{x}$ , which is known as the *code*.

Sparse coding has popular application in feature extraction from images [40; 41], in which case  $\mathbf{y}$  is an image (which, for our purposes, may be represented as a stacked vector). Then, the dictionary  $\mathbf{D}$  represents a set of prototype images with which we may try to recover the input image by linear superposition of the prototypes. The information about the strength of the contribution of each prototype image is recorded in the code. A sparse coding model is attractive for visual feature extraction, since there is some evidence that such a mechanism is also present in the mammalian visual cortex [42].

A popular approach to encoding images is JPEG, which reconstructs  $8 \times 8$  image sub-blocks using a dictionary containing 64 normalised Discrete Cosine Transform (DCT) bases. In the case of JPEG,  $N = M = 64$ , yielding a complete dictionary. In the context of sparse coding, it is understood that  $M \gg N$ , which makes the dictionary overcomplete as there can be at most  $N$  linearly independent columns in  $\mathbf{D}$ . This also renders the problem ill-posed, since there can be infinitely many solutions to the predicate of Equation (2.1). Determining the optimal solution to Equation (2.1) is known to be NP-hard [43] in the general case where the dictionary atoms may be linearly dependent and exhibit mutual coherence.

In some cases, perfect reconstruction is not desirable, especially as it may come at the expense of sparsity. Then, an alternative formulation of the problem is to minimise a cost function [44]

$$C(\mathbf{x}) = \|\mathbf{D}\mathbf{x} - \mathbf{y}\|_2^2 + \beta\tau(\mathbf{x}). \quad (2.2)$$

Equation (2.2) expresses the cost in terms of a weighted sum of the reconstruction error and the code *diversity*  $\tau(\mathbf{x})$ , which produces low values for a sparse code and higher values for one that is less sparse. The most direct diversity measure is simply

$$\tau(\mathbf{x}) = \|\mathbf{x}\|_0. \quad (2.3)$$

The value of  $\beta$  represents the trade-off made between a small reconstruction error and a large sparsity.

### 2.2.1.1 Differentiable diversity measures

In order to facilitate the application of gradient-descent based approaches to the optimisation of Equation (2.2), it may be required that the diversity measure of the code be differentiable, which the  $l_0$  pseudo-norm is not. Examples of differentiable diversity measures include [44; 45]

$$(a) \quad \tau(\mathbf{x}) = \sum_i \log\left(1 + \left(\frac{\mathbf{x}(i)}{\sigma}\right)^2\right) \quad (2.4)$$

and

$$(b) \quad \tau(\mathbf{x}) = \sum_i G(\mathbf{x}(i)), \quad G(x) = \begin{cases} \frac{2x}{\sigma} - \left(\frac{x}{\sigma}\right)^2 & \text{if } x < \sigma \\ 1 & \text{if } x \geq \sigma \end{cases}. \quad (2.5)$$

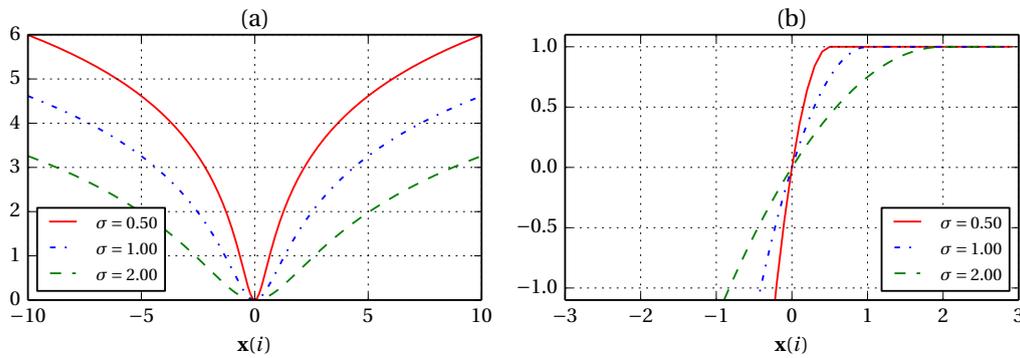


Figure 21: Differentiable diversity measures for the optimisation of sparse codes.

They are shown in Figure 21. In both cases, the parameter  $\sigma$  adjusts the shape of the diversity measure to accommodate differences in the variance of the code.

### 2.2.1.2 Dictionary learning

The sparse coding problem stated in Equation (2.1) implicitly assumes that  $\mathbf{D}$  is known. If it is not known, then we are faced with the problem of optimising the cost function  $C(\mathbf{D}, \mathbf{x})$  in both parameters simultaneously. In order to manage the complexity of the task, most approaches to dictionary training [45–47] choose to keep the code fixed and optimise only the dictionary with respect to the cost function. Simultaneous learning of both the code and dictionary thus typically proceeds by iteratively updating first one, and then using the result to update the other [48–50]. Since the standard dictionary learning approaches for sparse coding do not extend well to the convolutional setting (which we will introduce in Section 2.2.3 below), we will omit a discussion of them in this chapter.

### 2.2.1.3 Sparse coding and dictionary learning in a probabilistic setting

Approaches such as those proposed by Olshausen and Field [51] cast sparse coding and dictionary learning into a probabilistic framework. These authors show that minimising the Kullback-Leibler divergence between the distribution of sequences generated by our dictionary  $p(\mathbf{y}|\mathbf{D})$  and the actual distribution of the sequences  $p(\mathbf{y})$  is equivalent to maximising the expected log-likelihood

$E[\log p(\mathbf{y}|\mathbf{D})]$ . The distribution  $p(\mathbf{y}|\mathbf{D})$  can be obtained by marginalising  $p(\mathbf{y}, \mathbf{x}|\mathbf{D})$ :

$$p(\mathbf{y}|\mathbf{D}) = \int p(\mathbf{y}|\mathbf{D}, \mathbf{x}) p(\mathbf{x}) d\mathbf{x}. \quad (2.6)$$

We therefore need to specify two distributions: the probability of our observed sequence having been generated by the given code and dictionary  $p(\mathbf{y}|\mathbf{D}, \mathbf{x})$ , and a prior probability for a given code  $p(\mathbf{x})$ . Since  $p(\mathbf{y}|\mathbf{D}, \mathbf{x})$  can be interpreted as expressing the probability of observing a sequence  $\mathbf{y}$  that was generated by a certain fixed dictionary and code, any residual difference between our observation and the reconstruction formed by  $\mathbf{D}\mathbf{x}$  can be thought of as noise introduced during measurement. Olshausen and Field model the residual  $\mathbf{r} = \mathbf{D}\mathbf{x} - \mathbf{y}$  as additive Gaussian white noise. Therefore:

$$p(\mathbf{y}|\mathbf{D}, \mathbf{x}) = \frac{1}{Z_{\sigma_r}} e^{-\frac{\|\mathbf{D}\mathbf{x} - \mathbf{y}\|_2^2}{2\sigma_r^2}}, \quad (2.7)$$

where  $Z_{\sigma_r}$  is a normalising constant and  $\sigma_r^2$  is the variance of the additive noise.

The prior distribution,  $p(\mathbf{x})$ , is where sparseness can be introduced into the model, since we can shape it to peak at zero for each code element, and then monotonically decrease. Olshausen and Field parameterise the distribution as

$$p(\mathbf{x}) = \frac{1}{Z_\beta} e^{-\beta\tau(\mathbf{x})}. \quad (2.8)$$

Evaluating the integral given in Equation (2.6) involves summing over the entire code space, which is not feasible in practice. Olshausen and Field propose using the extremal value of  $p(\mathbf{y}, \mathbf{x}|\mathbf{D})$  as an approximation to the volume under the integral, which is an acceptable approximation as long as  $p(\mathbf{y}, \mathbf{x}|\mathbf{D})$  is strongly peaked in the code space.

Finally, we can express the sparse coding and dictionary learning problem in a probabilistic setting:

$$\begin{aligned} \mathbf{D} &= \arg \max_{\mathbf{D}} E[\log p(\mathbf{y}|\mathbf{D})] \\ &= \arg \max_{\mathbf{D}} E \left[ \max_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{D}, \mathbf{x}) p(\mathbf{x}) \right]. \end{aligned} \quad (2.9)$$

Substituting Equations (2.7) and (2.8) into Equation (2.9) and simplifying yields

$$\mathbf{D} = \arg \min_{\mathbf{D}} E \left[ \min_{\mathbf{x}} \left( \|\mathbf{D}\mathbf{x} - \mathbf{y}\|_2^2 + \beta\tau(\mathbf{x}) \right) \right], \quad (2.10)$$

which shows that Olshausen and Field's approximate maximum likelihood approach yields exactly the cost function chosen in Equation (2.2).

## 2.2.2 Determining sparse codes

In this section we give an overview of some of the sparse coding approaches found in the literature. All iteratively improve the code in some way. They differ in the quantity being optimised, be it the reconstruction error, some sparsity measure or a composite cost function of the two. They also differ in whether they optimise a cost function in a principled manner using mathematical optimisation or proceed in a heuristic or greedy fashion. In this section, we will focus on the greedy approaches to sparse coding.

### 2.2.2.1 Matching pursuit

Matching pursuit (MP) is a very simple greedy algorithm proposed by Mallat and Zhang [52; 53]. The algorithm starts by setting the initial reconstruction error  $\mathbf{r}$  and code such that

$$\mathbf{r}_0 = \mathbf{y} \quad \text{and} \quad \mathbf{x}_0 = \mathbf{0}. \quad (2.11)$$

The goal is then to iteratively minimise  $\mathbf{r}_n$  by choosing the dictionary atom  $\mathbf{d}_k$  which maximises the inner product between  $\mathbf{d}_k$  and  $\mathbf{r}_{n-1}$ :

$$\mathbf{d}_n = \underset{\mathbf{d}_k}{\operatorname{argmax}} (\mathbf{d}_k \cdot \mathbf{r}_{n-1}). \quad (2.12)$$

This leads to the updated residue and code:

$$\mathbf{r}_n = \mathbf{r}_{n-1} - (\mathbf{d}_n \cdot \mathbf{r}_{n-1}) \mathbf{d}_n \quad \text{and} \quad \mathbf{x}_n(k) = \mathbf{x}_{n-1}(k) + (\mathbf{d}_n \cdot \mathbf{r}_{n-1}) \quad (2.13)$$

The formulation given above does not directly address any sparseness constraints. Since each iteration either increases or maintains the number of dictionary atoms activated to reconstruct  $\mathbf{y}$ , one can directly control the  $l_0$  pseudo-norm of the code. Alternatively, the reconstruction error  $\mathbf{r}$  can be controlled by iterating until  $\|\mathbf{r}_n\|_2$  drops below a desired threshold, which it should do eventually, provided that the dictionary contains  $N$  linearly independent atoms [54]. For sparse coding, some compromise should be made between the two.

MP does not in general yield optimal codes. Research on speech applications of sparse coding [46] has also shown that it does not necessarily yield robust codes, since slight signal variations (such as can be introduced by noise)

can result in substantially different codes, which impedes effective pattern recognition. However, a lack of robust encoding with MP may be symptomatic of a dictionary which exhibits undesirable levels of coherence (or between-atom similarity). Since a suitably incoherent dictionary may well yield a robust coding with MP [53], and since the algorithm is efficient and quickly convergent, it should not be immediately dismissed.

### 2.2.2.2 Orthogonal matching pursuit

Orthogonal matching pursuit (OMP) is presented as a modification of MP by Pati et al. [55]. The key difference is that, after a new atom  $\mathbf{d}_n$  is chosen, the coding is recalculated by allowing all the activated dictionary atoms  $\mathbf{D}_n$  to participate in the reconstruction simultaneously. This addresses the fact that the code developed by MP for a given  $\mathbf{D}_n$  is generally not optimal in terms of normed residue, i.e. that there exists some other linear combination of the activated dictionary atoms that reconstructs  $\mathbf{y}$  more accurately.

In [53], the stated goal of OMP is to minimise  $\|\mathbf{D}_n \mathbf{x}_n^a - \mathbf{y}\|_2$ , where  $\mathbf{x}_n^a$  is a compacted code containing only the coefficients for the active atoms. An ordinary least squares (OLS) regression yields:

$$\mathbf{x}_n^a = (\mathbf{D}_n^T \mathbf{D}_n)^{-1} \mathbf{D}_n^T \mathbf{y} \quad (2.14)$$

An alternative formulation given by Pati et al. [55] is a regression on

$$\mathbf{d}_n = \mathbf{D}_{n-1} \mathbf{b}_{n-1} + \boldsymbol{\gamma}_{n-1} \quad \text{with} \quad \mathbf{D}_{n-1}^T \boldsymbol{\gamma}_{n-1} = \mathbf{0} \quad (2.15)$$

which asserts a decomposition of  $\mathbf{d}_n$  into a component that is linearly dependent on  $\mathbf{D}_{n-1}$  and a component that is orthogonal to all the columns in it. The code updates are then given by

$$\mathbf{x}_n^a(k) = \frac{\mathbf{r}_{n-1} \cdot \mathbf{d}_n}{\boldsymbol{\gamma}_{n-1} \cdot \mathbf{d}_n}, \quad k = n \quad (2.16)$$

$$\mathbf{x}_n^a(k) = \mathbf{x}_{n-1}^a(k) - \mathbf{x}_n^a(k) \mathbf{b}_{n-1}(k), \quad k = 1, \dots, n-1 \quad (2.17)$$

It is worth noting that [55] also derives a recursive expression for  $\mathbf{b}_n$  in terms of  $\mathbf{b}_{n-1}$ , offering a substantial improvement in computational efficiency over the naïve regression seen in Equation (2.14).

OMP is guaranteed to converge to a specified  $\|\mathbf{r}\|_2$  threshold in fewer iterations than MP, which means it also yields a sparser code. According to [55], the difference between MP and OMP in terms of computational cycles consumed is inconclusive, as it is dependent on the signal and dictionary used.

### 2.2.2.3 Coordinate descent

The coordinate descent algorithm proposed by Li and Osher in [56] greedily pursues a minimal  $l_1$  code (setting  $\tau(\mathbf{x}) = \|\mathbf{x}\|_1$ ) by iteratively sweeping through the code and optimising one coefficient  $x_k$  at a time, reducing the optimisation of the cost function to the one-dimensional problem

$$\tilde{x}_k = \underset{x_k}{\operatorname{argmin}} (x_k - p_k)^2 + \beta |x_k|, \quad (2.18)$$

where  $p_k$  represents the inner product of the  $k^{\text{th}}$  dictionary atom and the residual sequence that excludes the contribution of that atom. The one-dimensional problem has an optimal solution which is given in terms of a shrink operator:

$$\tilde{x}_k = \operatorname{shrink}\left(p_k, \frac{\beta}{2}\right), \quad \operatorname{shrink}(f, \mu) = \begin{cases} f - \mu & \text{if } f > \mu \\ 0 & \text{if } -\mu \leq f \leq \mu \\ f + \mu & \text{if } f < -\mu \end{cases}. \quad (2.19)$$

Li and Osher show that a simple sequential sweep through the code coefficients produces too many non-zero elements. Therefore, in order to promote the emergence of a sparser code, these authors ultimately employ an adaptive sweeping strategy by iteratively choosing the coefficient whose optimisation would yield the largest reduction  $\Delta C$  in the cost function. Li and Osher choose to approximate the reduction achieved by the optimisation of the  $k^{\text{th}}$  code coefficient  $\Delta C_k$ , such that

$$\Delta C_k \approx |x_k - \tilde{x}_k|. \quad (2.20)$$

This coordinate descent approach combined with adaptive sweeping appears to converge remarkably quickly, while also yielding codes that are more stable than those obtained by matching pursuit.

### 2.2.3 Convolutional sparse coding

Thus far we have considered a straightforward sparse coding formulation where we attempt to reconstruct a signal or set of signals using dictionary atoms spanning the entire length of the signals involved. This formulation is not particularly appropriate for the purpose of describing speech signals. In particular, there are two main deficiencies. The first of these is that phonemic sub-units generally occur at any point in a signal, which necessitates a highly redundant dictionary to accommodate all possible time shifts of that unit. The second deficiency is that the units can be compressed or stretched in time and still retain their meaning, leading to an even more redundant set of atoms.

#### 2.2.3.1 Shift invariance

To deal with the first deficiency, several authors [42; 46; 57] have proposed a convolutional extension to sparse coding—often referred to as Shift-Invariant Sparse Coding (SISC). The convolutional sparse coding problem can be posed in the same way as the straightforward sparse coding problem, with the exception that we now replace the dictionary-code product  $\mathbf{D}\mathbf{x}$  with a dictionary-code convolution, which we define as

$$\mathbf{\Phi} * \mathbf{S} = \sum_{j=1}^M \boldsymbol{\phi}^j * \mathbf{s}_T^j, \quad (2.21)$$

where  $\mathbf{\Phi} \in \mathbb{R}^{N_{\phi} \times M}$  is the convolutional dictionary,  $\mathbf{S} \in \mathbb{R}^{M \times N}$  are the coefficient sequences and  $\boldsymbol{\phi}^j$  and  $\mathbf{s}_T^j$  refer to the  $j^{\text{th}}$  column and row of  $\mathbf{\Phi}$  and  $\mathbf{S}$  respectively. Each dictionary atom  $\boldsymbol{\phi}^j$  is now associated with a coefficient sequence  $\mathbf{s}_T^j$  that represents not just whether a dictionary atom is being used, but also at what position in  $\mathbf{y}$ .

#### 2.2.3.2 Scale invariance

To address the second deficiency, scale invariance can be afforded to the convolutional sparse coding formulation by including each base atom at several time scales. This extension is less elegant than the shift-invariant extension described above, since it adds redundant atoms to the dictionary. However, at least this explicit redundancy allows scaled versions of base atoms to be associated with each other.

### 2.2.3.3 Applicability to speech signals

If one examines the phonetic hand transcriptions of speech corpora such as TIMIT, it becomes apparent that they represent (or could be interpreted as) a high-level convolutional sparse coding of speech. In particular, the phones act as a dictionary of units, and the associated time-aligned phone transcriptions of each utterance can be interpreted as a sparse code. It therefore seems plausible that a phoneme-like transcription could arise naturally from a convolutional sparse coding pursuit on the acoustic data.

### 2.2.3.4 Determining convolutional sparse codes

It is useful to note that we can express the result of the convolutional sparse coding as a matrix-vector product of the form

$$\mathbf{y} = \mathbf{\Phi} * \mathbf{S} = \mathbf{D}_\phi \mathbf{x}_s, \quad (2.22)$$

where  $\mathbf{D}_\phi = [\mathbf{C}(\phi^1) \mathbf{C}(\phi^2) \cdots \mathbf{C}(\phi^M)]$  consists of the  $M$  blocks  $\mathbf{C}(\phi^j) \in R^{N \times N}$ , each containing all  $N$  possible shifts of the atom  $\phi^j$  and  $\mathbf{x}_s = [\mathbf{s}_T^1 \mathbf{s}_T^2 \cdots \mathbf{s}_T^M]^T$  [57]. Since  $\mathbf{x}_s$  and  $\mathbf{S}$  are merely reshaped versions of the same set of coefficients, we can use an existing sparse coding algorithm to obtain  $\mathbf{x}_s$  and by extension  $\mathbf{S}$ . The only requirement we place on these sparse coding algorithms is that they are able to evaluate operations involving  $\mathbf{D}_\phi$  implicitly and efficiently by exploiting its sparse and redundant structure.

In cases where this is not possible, or as an additional optimisation, Plumbley et al. [58] suggest reducing the dimensionality of  $\mathbf{D}_\phi$  by only retaining those columns of  $\mathbf{C}(\phi^j)$  that correspond with shifts where the cross-correlation between  $\phi^j$  and  $\mathbf{y}$  are locally maximal.

### 2.2.3.5 Dictionary learning in the convolutional setting

Unfortunately, the dictionary learning algorithms are less amenable to direct application to the convolutional sparse coding problem, since they would produce updates to  $\mathbf{D}_\phi$  that violate the structural properties needed to unambiguously map back to  $\mathbf{\Phi}$ .

It is worth noting that we can arrive at an analytical solution to the dictionary update, which can be seen as a simplification of the method of Grosse et al. in

[47]. The form of the partial derivative of the cost function with respect to the dictionary atoms (given in the section below) exhibits tight coupling between all atoms, which means that greedily optimising one atom at a time (through e.g. some form of deconvolution and decorrelation) is likely to be globally suboptimal.

However, transforming the problem to the discrete frequency domain does allow decoupling of variables. This transformation relies on the property that the discrete Fourier transform of a sequence scales its  $l_2$  norm by a known constant factor (Parseval's theorem). Since the cost function  $C(\Phi)$  is simply the squared norm of the reconstruction error sequence, it is also equal to the squared norm of the DFT of the reconstruction error sequence divided by a known constant:

$$C(\Phi) = \|\mathbf{r}\|_2^2 = \frac{\|\hat{\mathbf{r}}\|_2^2}{K}, \quad \hat{\mathbf{r}} = \text{DFT}\{\mathbf{r}\}. \quad (2.23)$$

Therefore, minimising one is equivalent to minimising the other. Applying the DFT to the reconstruction error, we obtain

$$C(\Phi) = \frac{1}{K} \left\| \hat{\mathbf{y}} - \sum_{j=1}^M \hat{\boldsymbol{\phi}}^j \circ \hat{\mathbf{s}}_T^j \right\|_2^2, \quad \begin{aligned} \hat{\mathbf{y}} &= \text{DFT}\{\mathbf{y}\} \\ \hat{\boldsymbol{\phi}}^j &= \text{DFT}\{\boldsymbol{\phi}^j\} \\ \hat{\mathbf{s}}_T^j &= \text{DFT}\{\mathbf{s}_T^j\} \end{aligned} \quad (2.24)$$

where we have used the fact that time domain convolution results in element-wise multiplication (denoted here by  $\circ$ ) in the frequency domain, provided that adequate zero padding is applied to ensure the support we need. Note from the definition of the  $l_2$  norm, we can rewrite Equation (2.24) as the sum of the squared magnitudes of the individual discrete frequency components:

$$C(\Phi) = \frac{1}{K} \sum_{k=1}^K \left| \hat{\mathbf{y}}(k) - \sum_{j=1}^M \hat{\boldsymbol{\phi}}^j(k) \hat{\mathbf{s}}_T^j(k) \right|^2 = \frac{1}{K} \sum_{k=1}^K |\hat{\mathbf{y}}(k) - \hat{\mathbf{s}}_k \hat{\boldsymbol{\phi}}_k|^2, \quad (2.25)$$

$$\hat{\mathbf{s}}_k = [\hat{\mathbf{s}}_T^1(k) \quad \hat{\mathbf{s}}_T^2(k) \quad \dots \quad \hat{\mathbf{s}}_T^M(k)], \quad \hat{\boldsymbol{\phi}}_k = [\hat{\boldsymbol{\phi}}^1(k) \quad \hat{\boldsymbol{\phi}}^2(k) \quad \dots \quad \hat{\boldsymbol{\phi}}^M(k)]^T.$$

Minimising  $C(\Phi)$  for each  $\hat{\boldsymbol{\phi}}_k$ , Grosse et al. obtain:

$$\hat{\boldsymbol{\phi}}_k = \hat{\mathbf{y}}(k) (\hat{\mathbf{s}}_k^* \hat{\mathbf{s}}_k)^{-1} \hat{\mathbf{s}}_k^*, \quad \hat{\mathbf{s}}_k^* = \text{Conj}\{\hat{\mathbf{s}}_k^T\}. \quad (2.26)$$

Note that the inversion of  $\hat{\mathbf{s}}_k^* \hat{\mathbf{s}}_k$  requires it to be full-rank and numerically well-conditioned, which is not something that we can generally guarantee or expect. The actual approach used by Grosse et al., places further constraints on the optimisation in order to help overcome degeneracy in  $\hat{\mathbf{s}}_k^* \hat{\mathbf{s}}_k$ .

## 2.2.4 Applications in Automatic Speech Recognition

Sparse coding has found application in many sub-fields of signal processing. In this section we review work applying sparse coding to speech-related tasks.

### 2.2.4.1 Grosse et al. (2007)

In [47], Grosse et al. investigated the use of convolutional sparse coding features for five-way speaker identification. In particular, they operated within a “self-taught learning” framework, where few labeled data are available, but many unlabeled data—potentially corresponding to classes other than just those in the labeled training set. Grosse et al. employed the unlabelled speech samples by learning SISC atoms from them, which the labelled samples are then coded with to produce new features.

The results obtained are somewhat inconclusive, in so far as the raw spectrogram features actually gave better classification than the MFCC and SISC features for data that was noise free or had the same kind of noise (e.g. Gaussian) added to all samples. SISC features did, however, give the best performance across the board when different kinds of noise were added to the training and test sets.

### 2.2.4.2 Smit and Barnard (2009)

Smit and Barnard [46] have applied convolutional sparse coding to continuous speech recognition using the TIDIGITS dataset. The code they develop represents a temporal “spike stream” of acoustic events. These events form the dictionary and each one consists of 13 spectrotemporal frames representing 20 ms of acoustic data. The code atoms may represent anything from phonemes to entire words.

The dictionary atoms learned in [46] exhibit highly specialised spectrotemporal structure; however, they do not neatly correspond to phoneme-like units. The learned atoms are also often used in conjunction with each other to reconstruct sounds—in some cases to subtract a part of the contribution of another shifted atom.

Smit and Barnard also perform a classification task on the sparse code stream. They determine the posterior probability of a spike stream conditioned

on a model using a spike train model of order statistics and spike amplitudes. Since the atoms used are long enough to represent words, the models are forced to correspond to words. The most likely sequence of models and spike train segment lengths is determined by performing dynamic programming on the model transition lattice. The classification results obtained are inferior to an HMM-based approach using similar spectrogram-derived features.

#### 2.2.4.3 Sivaram et al. (2010)

Sivaram et al. [45] have investigated the use of sparse codes as feature vectors for a speaker independent phoneme recognition task on TIMIT. They use a small subset of spectro-temporal patterns from the TIMIT training data to learn a 429-atom dictionary. This dictionary is then used to code all the spectro-temporal patterns used in their investigation into sparse acoustic feature vectors.

A multi-layer perceptron is trained to estimate the posterior probabilities of phonemes conditioned on the sparse codes, which are used as the emission probabilities of tri-state HMM phoneme models. Phoneme recognition is achieved by decoding the sparse code stream using the Viterbi algorithm. Sivaram et al. demonstrate an 0.8% absolute improvement on phoneme recognition accuracy using sparse codes as feature vectors over standard PLP features. In noisy conditions, this improvement increases to 6.3%.

#### 2.2.4.4 Vinyals and Deng (2012)

In [49], Vinyals and Deng investigate the use of a fairly straightforward sparse coding and linear classification strategy on a TIMIT phoneme recognition task. They hypothesise that since non-linear mappings between acoustic features and phone labels can be arbitrarily well approximated using local piece-wise linear functions, a sparse coding step followed by a linear classifier (such as a support vector machine) may deliver comparable performance to deep learning approaches.

The results obtained in [49] show that while sparse coding followed by an SVM linear classifier dramatically outperforms linear classification based on the raw features (an 11.4% absolute improvement on the test set in terms of per frame phone state accuracy), it still delivers inferior performance relative to

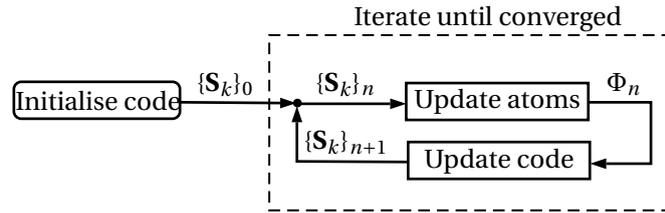


Figure 22: Overview of sparse code and dictionary learning with an initial code.

a deep architecture, which delivers an additional 4.2% absolute improvement. Vinyals and Deng also report a phoneme recognition accuracy on TIMIT of 75.1%, which outperforms the 67.7% obtained by Sivaram et al. [45].

## 2.3 Sub-word unit discovery with Coordinate Descent

In this section, we investigate the feasibility of the convolutional sparse coding framework described in Section 2.2.3 for the task of automatic SWU discovery. In particular, we use a version of Coordinate Descent (see Section 2.2.2.3) that has been adapted to the convolutional setting.

### 2.3.1 Implementation details

The development of a set of sparse codes and an associated dictionary of atoms proceeds in an iterative fashion (shown in Figure 22). Starting with some initial set of codes  $\{\mathbf{S}_k\}_0$ , we calculate the optimal set of atoms  $\Phi_0$  corresponding to those codes. These atoms are then used to calculate a new set of codes  $\{\mathbf{S}_k\}_1$ , which are then used to recalculate a new set of atoms  $\Phi_1$ . This is repeated until convergence of  $C(\Phi_n, \{\mathbf{S}_k\}_n)$ .

#### 2.3.1.1 Initialisation

There are many possible strategies for obtaining an initial code for each utterance. Here we make the somewhat arbitrary choice of filling each  $\mathbf{S}_k$  matrix with ones according to the following rules:

1. Each time frame in the utterance is given a chance to contain a single non-zero coefficient according to a prespecified probability  $P_\theta$ , and

2. for those time instants that may contain a non-zero coefficient, the corresponding feature index is randomly chosen from a uniform distribution.

The role of  $P_\theta$  is to encourage an initial  $l_0$  code sparsity. It is chosen to lead to an average “phoneme” rate roughly similar to that of a real phoneme transcription.

### 2.3.1.2 Obtaining sparse codes

In order to generate sparse codes, we make use of the Coordinate Descent algorithm [56], which uses the  $l_1$  norm as a diversity function. Coordinate Descent greedily optimises the cost function by iteratively choosing the code coordinate (i.e. atom and point in time) whose optimal value would yield the largest reduction in cost. The algorithm terminates once the relative reduction achieved by the next optimal choice of coordinate falls below some threshold.

### 2.3.1.3 Atom updates

During the atom update stage, we would like to use the updated codes to produce a new dictionary that further reduces the cost function. However, the only term in  $C(\Phi, \{\mathbf{S}_k\})$  that we can optimise is the reconstruction error

$$R = \sum_{k=1}^K \|\mathbf{y}_k - \Phi * \mathbf{S}_k\|_2^2 = \sum_{k=1}^K \|\mathbf{r}_k\|_2^2. \quad (2.27)$$

The atoms are updated using a frequency-domain approach inspired by Grosse et al. [47], which recognises that  $R$  remains invariant (up to a scaling constant), when the DFT is applied to  $\mathbf{r}_k$ . This makes it possible to write the atom-code convolutions as elementwise multiplications, yielding a tractable way to solve for all  $\phi^j$  simultaneously in a way that minimises the reconstruction error.

In this study, each of the  $N_t$  possible time-scales at which an atom can occur becomes a distinct atom for the purposes of coding. Furthermore, the atom update approach described above also neglects the relationship between scaled versions of the same atom. Thus, after the scaled atoms have been updated individually, yielding the preliminary updated atoms  $\phi_n^{j'}$ , it becomes necessary to reassert the relationship between those that belong to a particular class. This is done without much rigour by scaling the atoms to a common length (using spline interpolation), and then taking a weighted average according to the relative

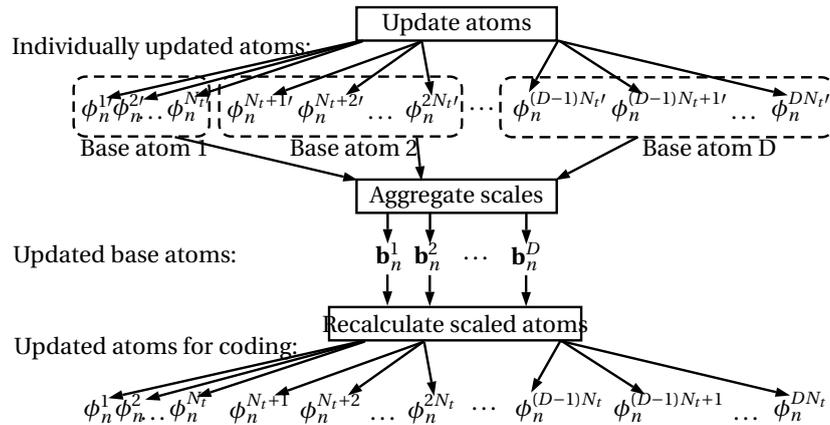


Figure 23: Overview of the dictionary update process.

frequency of the scales. These prototypical *base atoms*  $\{\mathbf{b}^d\}$  are then rescaled to all relevant lengths and made available for coding, as shown in Figure 23.

## 2.4 Experiments and results

In the following, we describe and report on the results of some experiments that were performed in order to determine the suitability of the sparse coding approach described in this chapter

### 2.4.1 Data selection and preprocessing

The following experiments use subsets of the TIMIT speech corpus. The 6300 utterances making up this corpus are divided into three categories of sentences, which are designed to expose different aspects of speech. These categories are:

- SA Dialect sentences**, which are meant to expose dialectal variation among 8 regions across the United States. The category consists of 2 sentences spoken by all 630 speakers.
- SX Phonetically compact** sentences designed to provide good coverage of phone pairs, with some emphasis on interesting or difficult contexts. The corpus contains 450 such sentences and each sentence is spoken by 7 speakers.
- SI Phonetically diverse** sentences are chosen to include diversity in sentence types and phonetic contexts and to maximise the variety of allophonic contexts found in a large corpus of text. The corpus contains 1890 SI sentences and each sentence is spoken only once.

Table 21: Training parameters

Parameter	Description
$\beta$	Diversity penalty
$N_\phi$	Maximum atom scale (number of frames)
$D$	Number of base atoms

We elect to use only the phonetically diverse (SI) sentences for training. The other categories contain so much repetition that their inclusion might bias the development of acoustic units that favour very specific contexts.

Before we apply sparse coding and dictionary learning to the selected training set, the utterances are converted to 12-coefficient MFCC feature vectors using the HMM Toolkit (HTK) [1]. The MFCCs are generated at a rate of one every 20 ms, with a window size of 32 ms, resulting in a 12 ms overlap between frames. For each speaker, the utterances are pooled for the application of a cepstral mean and variance normalisation.

## 2.4.2 Training overview

Using the selected data, initial experimentation was performed to identify the impact of varying the parameters available for tuning (see Table 21). For consistency, the same random initialisation was used for each set of parameters, and a fixed number of 20 training iterations was carried out. Each run took approximately 2 hours to complete using 64 AMD Opteron processor cores clocked at 2.1 GHz.

For the parameters used in this study, the training showed fast convergence. Figure 24 shows how the sparse coding and feature learning iteratively improves the cost-function until it converges to a local optimum.

Figures 25 and 26 show the effect of local variation of the diversity penalty and number of base atoms. Unsurprisingly, reducing the diversity penalty leads to codes that are less sparse and therefore a more accurate reconstruction. In most cases, increasing the number of base atoms yielded an improvement in the reconstruction error, with the exception of ( $\beta = 8, D = 70$ ). This may be symptomatic of numerical problems with the exact solver used for the feature updates, since much more severe instability was encountered in the cost

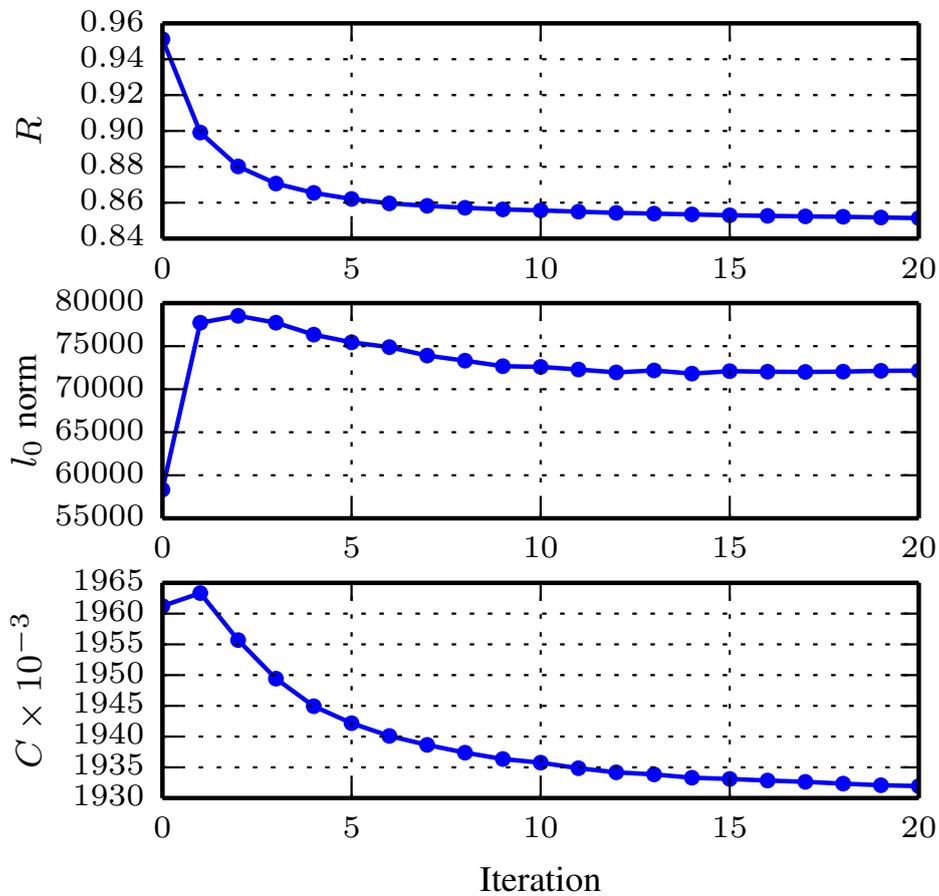


Figure 24: Development of the reconstruction error, code sparsity and cost function over 20 iterations of training for  $\beta = 9$ ,  $N_\phi = 20$  and  $D = 50$ .

function for larger values of  $N_\phi$  and  $D$ . Increasing the number of base atoms also yielded an increase in the number of non-zero code coefficients that was being used by the converged solution.

Figure 27 shows two of the atom dictionaries learned by our sparse coding and dictionary learning system. It is apparent that the atoms encode quite a bit of context to the left and right of the recognisable steady state areas. This seems to suggest that our system is more suitable to the discovery of context-dependent diphones or triphones.

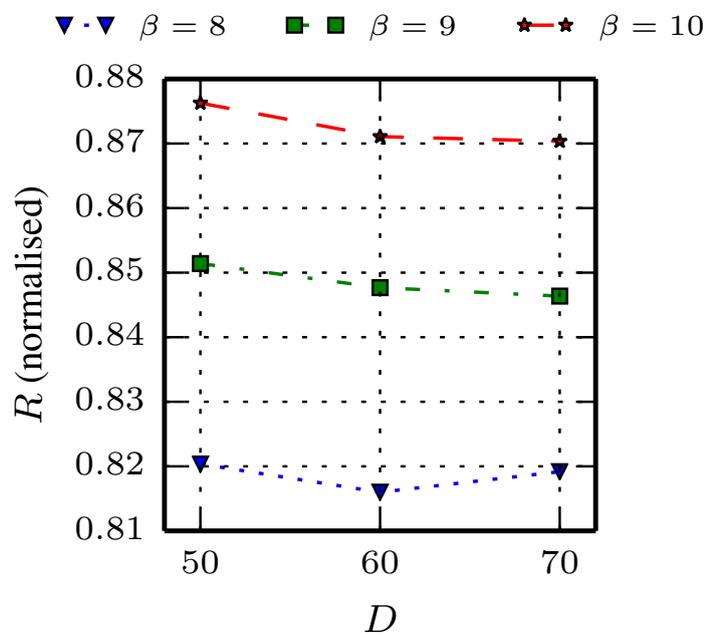


Figure 25: Final normalised reconstruction error vs. number of base atoms for various values of  $\beta$ .  $N_\phi = 20$ .

### 2.4.3 Coincidence with reference phonemes

In this section, we examine the relationship between our trained dictionary and the reference set of phonemes and transcriptions provided by TIMIT. Since we did not attempt to infer an optimal alignment between our sparse codes and the reference phoneme transcriptions, we simply count the number of times a non-zero coefficient corresponding to any given base atom occurs within a reference phonemic boundary. Those counts are then recorded in a 2D histogram which we call the *coincidence matrix*. In order to make it more graphically interpretable, the histogram is normalised so that the bins corresponding to each reference phoneme sum to one.

A disadvantage of this approach is that the coincidence matrix includes phonemic context—we are not just seeing which atoms are most frequently used to code certain phonemes, but also which atoms are often used just before or after a certain phoneme.

Figure 28 shows one such coincidence matrix. There are a noticeable number of cases where one phoneme is strongly coded by a small number of atoms. There are also many phonemes that are broadly coded using many different

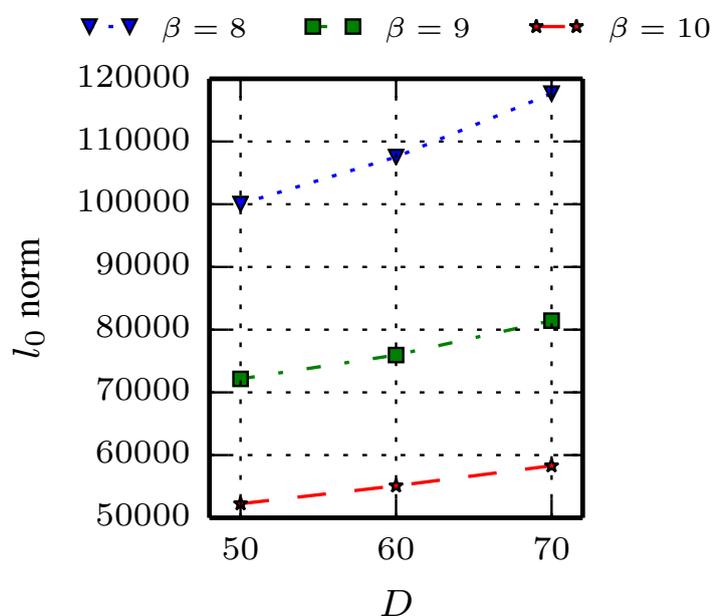


Figure 26: Number of coefficients used vs. number of base atoms for various values of  $\beta$ .  $N_\phi = 20$ . For comparison, the dataset contains 56377 reference phoneme instances.

atoms. In each case no particular atoms have developed that cater well for all occurrences of the phoneme. Overall, it is reasonable to conclude that at least some of the atoms learned are phonetically relevant.

#### 2.4.4 Usefulness to the generation of pronunciation lexicons

In order to use the resultant sparse codes and corresponding atom dictionaries to generate pronunciation lexicons using the atoms as sub-word units, it is necessary for the sparse codes to represent a reasonable segmentation of the acoustic data. At the very least, the sub-word units should not overlap in time.

We use a straightforward method to measure overlap: for each unique pair of non-zero code coefficients, we count the number of frames that overlap and add that to a running total. In order to compare the level of overlap between utterances of different lengths, the overlap score is normalised by the length of the utterance.

Figure 29 shows the average overlap scores for each set of parameters used in this study. In all cases, the overlap scores are much larger than unity, implying that the equivalent of each frame in the utterances is being coded (on average) by

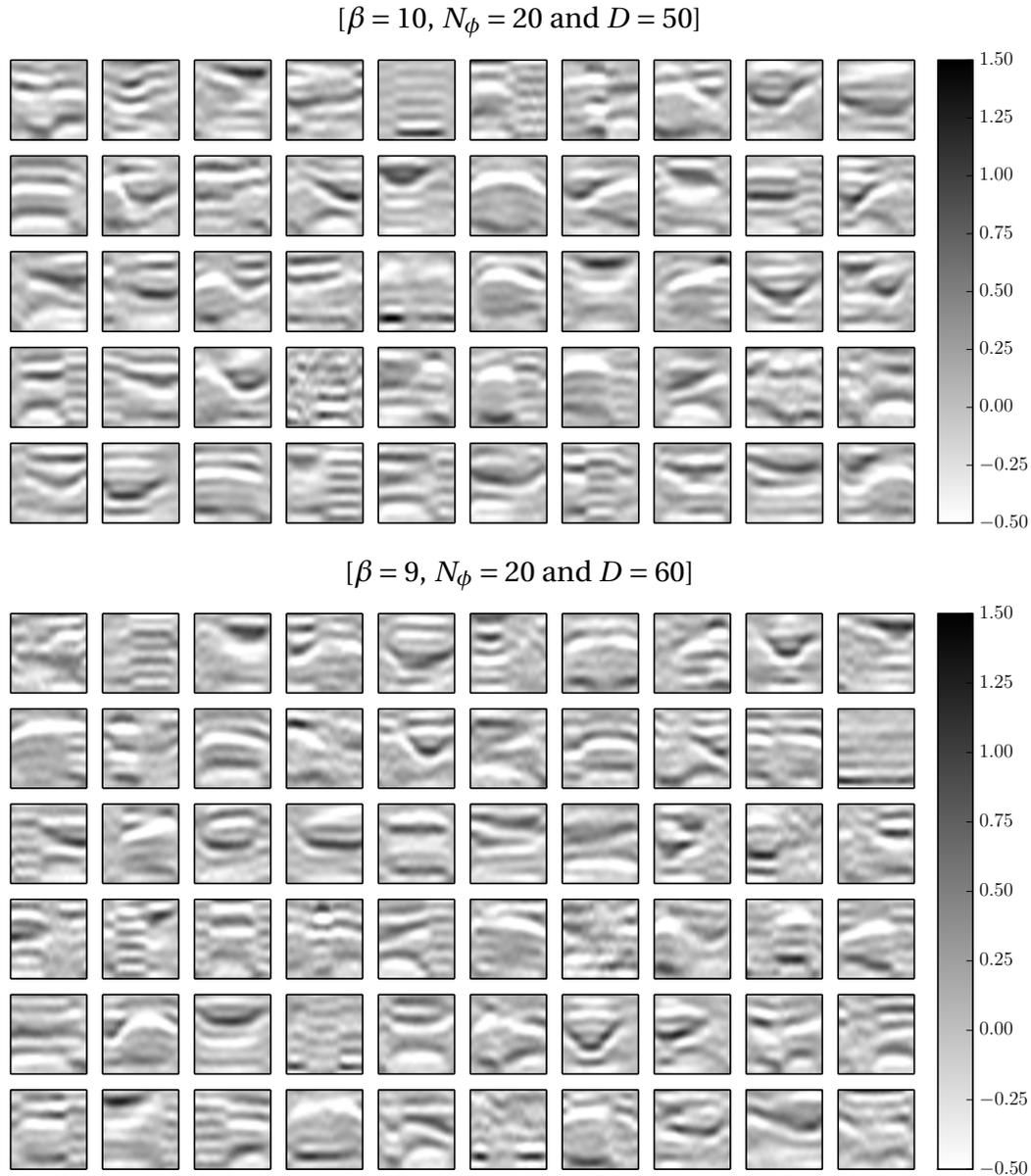


Figure 27: Mel spectrogram representation of base atom dictionaries after training. Lower frequencies are at the bottom.

several atoms. Clearly, the sparsity constraints we impose are not sufficient for a coding to arise that is localised enough in time to be useful for segmentation. It may be necessary to include a penalty term in the cost function to achieve a reduction in overlap.

The second requirement we place on our sparse codes and atoms for them to be fit for the generation of pronunciation lexicons, is that utterances that sound the same should be transcribed in the same way using our atoms. If that

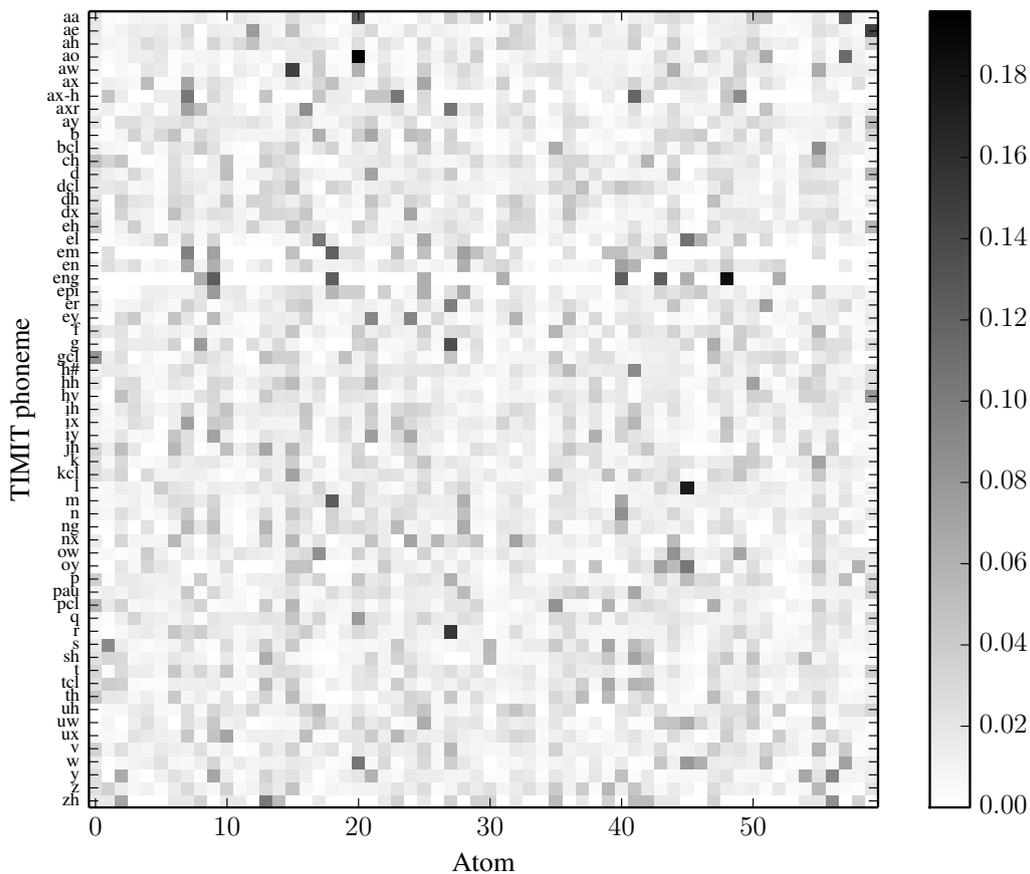


Figure 28: Coincidence matrix for  $\beta = 10$ ,  $N_\phi = 20$  and  $D = 60$ .

is not possible, we could end up with an impractically large set of competing pronunciations. Of course, not even the reference phonetic transcriptions for TIMIT are perfectly consistent for utterances with identical orthography, and multiple pronunciations for words often exist. Hence, it is important to establish a baseline consistency score.

A consistency score between two sparse codes can be obtained by calculating the Levenshtein (edit) distance between the corresponding sequences of base atoms, after they have been sorted according to their midpoints. We normalise the obtained edit distance by the length of the longer sequence, so that we have a score between 0 (when the sequences are exactly the same) and 1 (when there is no commonality at all). When we have multiple sequences to compare, we obtain a *within-class* consistency score by calculating the mean edit distance across all pairs of sequences corresponding to the same orthographic transcription.

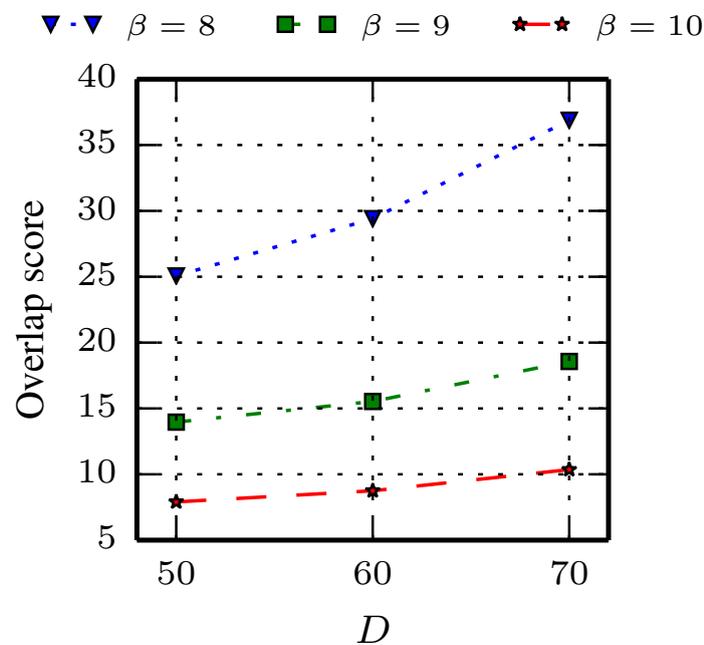


Figure 29: Normalised overlap scores for various training parameters.

Table 22: Baseline consistency scores using reference phoneme transcriptions

	SA1	SA2
Overall	0.221	0.232
New England	0.240	0.230
Northern	0.204	0.208
North Midland	0.204	0.222
South Midland	0.216	0.242
Southern	0.231	0.249
New York City	0.209	0.229
Western	0.218	0.237
Army Brat	0.205	0.207

We perform our consistency experiment on utterances from the SA dataset. This set contains only two orthographically unique sentences, and is spoken by all speakers in the TIMIT corpus. Table 22 contains the overall consistency scores of the reference transcriptions for these two utterances, as well as a breakdown per dialectical region. We then calculate sparse codes for the SA utterances

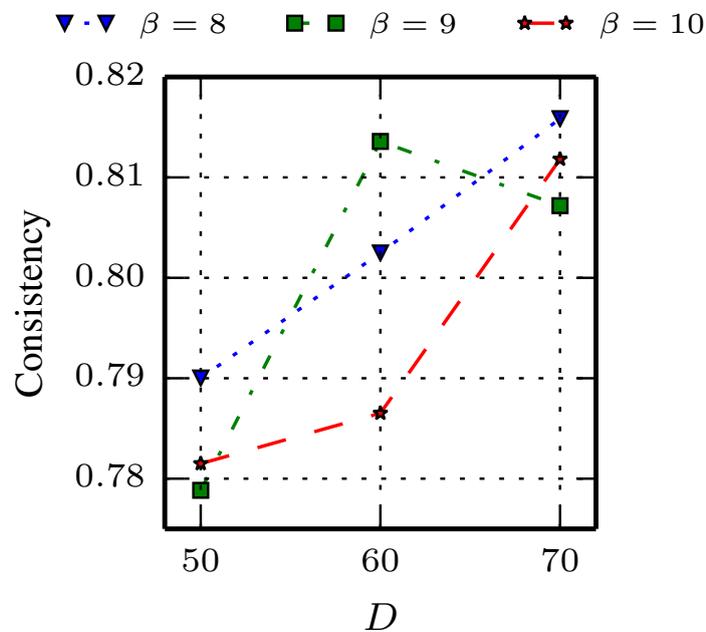


Figure 210: Average consistency scores on the SA dataset with atoms learned in previous experiments.

using the base atoms learned in our previous experiments, and score them for consistency. Figure 210 plots these scores, which range between 0.778 and 0.816. These figures indicate a level of consistency where there is only approximately 20% commonality in the transcriptions for different utterances with the same orthography. This is far too poor to be used to generate a compact pronunciation lexicon. Clearly there is a big margin for improvement.

The excessive level of overlap seen across all experiments may be partly to blame for the poor consistency scores. We hypothesise that, by allowing the system the freedom to develop codes with highly overlapping atom use, the development of atoms that are sufficiently able to represent temporally isolated acoustic events may be precluded, which may in turn lead to a poorer consistency score.

## 2.5 Conclusion

In this chapter, we found that it is possible to use sparse coding and dictionary learning to discover phonetically relevant units in acoustic data. However, the usefulness of these units for generating pronunciation lexicons is limited by the development of a segmentation that is not sufficiently localised in time and a transcription that is too inconsistent for utterances that are orthographically identical. In the next chapter we address the first limitation by constraining the sparse codes so that no two units may overlap with each other in time. Although we do not address the second limitation directly, we are optimistic that the no-overlap constraint will also improve coding consistency.

## Chapter 3

# Non-overlapping sparse codes and metaheuristic search

### 3.1 Introduction

In this chapter we present a novel implementation of sparse coding that is constrained to make it more appropriate for segmenting speech. In particular, we address the deficiency found in the previous chapter where the discovered sub-word units overlapped significantly with each other in their reconstruction of the input speech. The sparse codes developed in this chapter are constrained such that no two sub-word units overlap with each other in time. This enables the sparse codes to be unambiguously interpretable as a sequential transcription of the input signal in terms of a set of sub-word units. We also show that the constraint enables an optimal solution to the sparse coding sub-problem to be found. Together with a good approximation to the solution of the dictionary update sub-problem, this yields a convergent local search algorithm for sub-word units and their corresponding sparse codes. Finally, we embed this local sparse coding and dictionary learning algorithm inside a modified genetic search procedure to obtain a hybrid metaheuristic algorithm that explores the solution space more extensively.

## 3.2 Background

### 3.2.1 Metaheuristic search

Metaheuristic search refers to the use of higher-level search heuristics in order to explore the solution space of difficult combinatorial problems without exhaustively enumerating each solution. Two well-known examples are the class of genetic search algorithms, which mimic the biological process of natural selection, and simulated annealing, which uses a heuristic in which the probability of accepting or rejecting worse solutions is tied to a *cooling schedule* inspired by metallurgical annealing. The search algorithm we use in this chapter to optimise SWUs is inspired by genetic algorithms as well as by iterated local search, which is a lesser known example of metaheuristic search.

#### 3.2.1.1 Genetic algorithms

Genetic algorithms mimic the process of natural selection and evolution. They do this by maintaining a population of possible solutions, which is used to explore the solution space using the genetic operators of *selection*, *mutation* and *cross-over*.

Each iteration of the genetic algorithm begins with the *selection* of a subset of the previous generation to form breeding stock for the current generation. This selection is biased by some method, such as fitness proportional selection or tournament selection, to prefer fitter individuals (i.e. better solutions to the problem we are trying to solve). In this way the algorithm can heuristically improve the overall fitness of the population over time.

After a new set of parent solutions have been identified, they are recombined to form new individuals using the *cross-over* operator. Genetic algorithms assume that the solution to the problem one wants to solve can be expressed as a string of discrete tokens (genes) called a chromosome. In this step, some sections of each parent's genes are recombined sequentially to form a new chromosome. Cross-over between parent chromosomes could happen at one point (single point cross-over), or multiple points, with the extreme being to allow each gene position of the child chromosome a probability of originating from either parent.

The final step genetic algorithms perform when producing a new generation

of solutions, is to allow each child's chromosome a chance to *mutate*. This involves randomly replacing some genes with others. In the simplest representation of chromosomes represented by bit strings, this would entail flipping some bits.

### 3.2.1.2 Iterated local search

Iterated local search assumes the availability of a local search function that can reach the local optimum of the basin of attraction within which a candidate solution finds itself. Given some initial solution, the algorithm uses this local search function to reach its corresponding local optimum. A new solution is subsequently generated by randomly moving some distance away from the optimum, and treated as a new candidate for local search. This process can be repeated iteratively until a stopping criterion is met. Iterated local search works best when good local optima tend to cluster in each other's vicinity, so that one has a good chance of finding better solutions close to currently good solutions.

## 3.3 Implementation

### 3.3.1 Initializing the dictionary

To initialise our dictionary discovery process we apply a blind segmentation algorithm to the feature vectors extracted from the speech signal. This creates a pool of candidate basis functions from which we can draw to construct plausible initial dictionaries.

The approach used to perform the blind segmentation is that described by Ten Bosch in [7]. It inserts a boundary at  $t_i$  whenever the feature vectors immediately before and after ( $\mathbf{v}_{i-1}$  and  $\mathbf{v}_{i+1}$ , respectively), differ strongly enough. The degree of change is expressed by the vector cosine distance

$$D(\mathbf{v}_{i-1}, \mathbf{v}_{i+1}) = \arccos\left(\frac{\mathbf{v}_{i-1} \cdot \mathbf{v}_{i+1}}{\|\mathbf{v}_{i-1}\|_2 \|\mathbf{v}_{i+1}\|_2}\right). \quad (3.1)$$

The signal  $d(i) = D(\mathbf{v}_{i-1}, \mathbf{v}_{i+1})$ , is then smoothed by a symmetrical 3-point Hanning window, after which all local maxima of  $d(i)$  exceeding a certain threshold  $\delta$  are designated as segmentation boundaries.

### 3.3.2 Finding the optimal alignment of the basis functions

We now consider the task of determining the optimal coefficient sequence  $\mathbf{S}_k \in \mathbb{R}^{M \times N}$  for each input utterance  $\mathbf{y}_k$ , given a dictionary  $\Phi \in \mathbb{R}^{N_\phi \times M}$ . In other words, we want to find the best possible alignment of the input utterance with basis functions  $\phi_j$  from our dictionary  $\Phi$ . In order to quantify how good an alignment is, we use the cost function in Equation (2.2), with the  $l_0$  pseudo-norm as a sparsity constraint:

$$C(\Phi, \mathbf{S}_k) = \|\mathbf{y}_k - \Phi * \mathbf{S}_k\|_2^2 + \beta \|\mathbf{S}_k\|_0. \quad (3.2)$$

Exhaustively enumerating each possible alignment would be prohibitively computationally expensive. However, since we enforce the constraint that no basis function may overlap with another in the reconstruction, the optimal choice of coefficient given a time offset and basis function becomes unambiguous. This enables the alignment invariant calculation of a delta cost matrix  $\Delta C_k$ , with  $\Delta C_k[j, n]$  being the reduction in the cost function when basis function  $\phi_j$  is activated at time offset  $n$ . It is then possible to use a dynamic programming algorithm to find the path through the cost matrix that yields the largest total reduction in cost. The combination of the steps described below (Sections 3.3.2.1–3.3.2.2) lead to a novel algorithm for the optimal solution to the constrained sparse coding model.

#### 3.3.2.1 Optimal coefficient values

Given the input signal  $\mathbf{y}_k$  and the time-scale  $L_j$  of the basis function, the optimal coefficient for basis function  $\phi_j$  at the time-offset  $n$  can be calculated as

$$\mathbf{S}_{k, \text{opt}}[j, n] = \frac{\mathbf{y}_k[n : n + L_j] \cdot \phi_j}{\|\phi_j\|_2^2}, \quad (3.3)$$

with negative coefficients set to zero in order to disallow subtractive contributions in the reconstruction.

#### 3.3.2.2 Calculation of $\Delta C_k$

Since both the squared vector norm of the reconstruction residual and the  $l_0$  pseudonorm of the coefficient sequences are element-wise summations, the

change in the cost function is confined to the change in the norm of the reconstruction residual within the time interval during which the basis function  $\phi_j$  is active. Thus, we can calculate the reduction in cost function as

$$\begin{aligned} \Delta C_k[j, n] &= \|\mathbf{y}_k[n : n + L_j]\|_2^2 \\ &\quad - \|\mathbf{y}_k[n : n + L_j] - \mathbf{S}_{k,\text{opt}}[j, n]\phi_j\|_2^2 \\ &\quad - \beta. \end{aligned} \quad (3.4)$$

### 3.3.3 Determining the dictionary of basis functions

We now consider the task of obtaining the optimal set of basis functions given an alignment. Note that since we do not allow the basis functions to overlap in the reconstruction, it is possible to solve for each basis function independently. In order to determine the optimum  $\phi_{j,\text{opt}}$  for each basis function, we need to minimise the residual

$$R = \sum_n \|s_n \phi_j - \mathbf{y}_n\|_2^2, \quad (3.5)$$

where we have defined the set  $\{\mathbf{y}_n\}$  to contain all segments of the input signals for which the basis function  $\phi_j$  is used during reconstruction, and  $\{s_n\}$  the corresponding coefficients. Setting  $\partial R^2 / \partial \phi_j = 0$  and solving for  $\phi_j$ , we obtain

$$\phi_{j,\text{opt}} = \frac{\sum_n s_n \mathbf{y}_n}{\sum_n s_n^2}. \quad (3.6)$$

The update in Equation (3.6) does not take into account that we have made time-scaled versions of each basis function available for coding. The final step in updating the basis functions is therefore to reinforce this relationship among scaled versions of the same basis function. Suppose the set  $\{\phi_n^l\}$  contains the updated basis functions derived from the same prototype  $\phi^l$ , resampled to a common time-scale and normalised to unit norm. Also let  $\sigma_n^l$  denote the number of times the scaled basis function  $\phi_n^l$  is aligned with the input utterances, and  $L_n^l$  the length of that scaled basis function. A good approximation to the optimal prototype basis function can be calculated as

$$\phi^l = \frac{1}{Z_l} \sum_n L_n \sigma_n \phi_n^l, \quad (3.7)$$

where  $Z_l$  is a normalising factor. The factor  $L_n \sigma_n$  represents a very good approximation of the relative importance of  $\phi_n^l$  in reducing the cost function,

since each instance where a basis function is used approximately decreases the cost function by a constant value proportional to that basis function's length.

### 3.3.4 Local search through iterative improvement

Having determined procedures for finding an optimal alignment given a dictionary, and a close to optimal set of basis functions given an alignment, we now develop a novel local search procedure to improve from an initial dictionary. The procedure includes repeated alignment of the dictionary with the input data, and subsequent use of that alignment to produce a new dictionary.

### 3.3.5 Improving dictionaries by metaheuristic search

The search procedure developed in the previous section is capable of reliably improving initial dictionaries. However, the local search converges is likely to converge at a local rather than global optimum. One approach often taken in such cases to find a more globally optimal dictionary (especially in the context of clustering), is to simply try a multitude of random initialisations.

In this chapter, we turn to the metaheuristic strategies detailed below to improve our chances of finding good solutions. As an ensemble, these strategies can be seen as a modified genetic algorithm, retaining the notion of maintaining a population of solutions, as well as the genetic operators of selection and mutation. The *chromosome* of each individual in the population is in our case a dictionary of prototypical (i.e. scale invariant) basis functions, with the individual basis functions seen as the *genes*. The fitness of a particular individual is simply the cost (see Equation (3.2)), of the optimal alignment of its dictionary with the training data.

#### 3.3.5.1 Selection

When preparing a new generation of solutions, the first step is to select the individuals from the previous generation which are to be used as the basis of the new generation. Since we want to exploit those solutions that show promise, and abandon those that do not, more offspring should be derived from fitter

solutions. However, applying this bias too aggressively results in a loss of genetic diversity and hence memory of areas in the fitness landscape that show promise.

We therefore use a scheme for choosing the selection probabilities known as rank selection, where the fitnesses of the individuals in the previous generation are ranked with the best individual achieving rank  $N$  and the worst rank 1 [59]. The expected number of offspring of individual  $i$  is tied to this rank  $R[i]$  such that

$$E[i] = m_i + (m_a - m_i) \frac{R[i] - 1}{N - 1}, \quad (3.8)$$

where  $m_a$  is the variable selection pressure, and is constrained such that  $1 \leq m_a \leq 2$  and  $m_i = 2 - m_a$ .

We use stochastic universal sampling (SUS) to perform the selection itself, which guarantees that the actual number of offspring of each individual  $O[i]$  is constrained to  $\lfloor E[i] \rfloor \leq O[i] \leq \lceil E[i] \rceil$  [60].

### 3.3.5.2 Mutation

After selecting parents, new offspring can be produced by mutation, whereby each gene is given a small probability  $\mu$  of being modified. In order for the modification to represent a sensible new direction for the search, we choose a scheme where the affected gene is replaced by a segment randomly drawn from the pool of blind segmentations that was created during initialization. In order to ensure that the fitness of each generation is at least as good as the previous one, we protect one instance of each of the  $\epsilon$  best individuals from the previous generation from mutation.

### 3.3.5.3 Iterated search

The dictionaries developed through a random initialization, as well as those disrupted through mutation, are generally quite far from the minima of their basins of attraction. Even more problematically, they would not be at comparable distances from their eventual convergence point. When we compare the fitnesses of dictionaries, we would actually like to compare the minima of their respective basins of attraction. Failing that, we want all the dictionaries in the population to be roughly the same *distance* away from their minima. Anything else would lead to a scheme that cannot reliably distinguish between unfit individuals in deep basins and fit individuals in shallow basins.

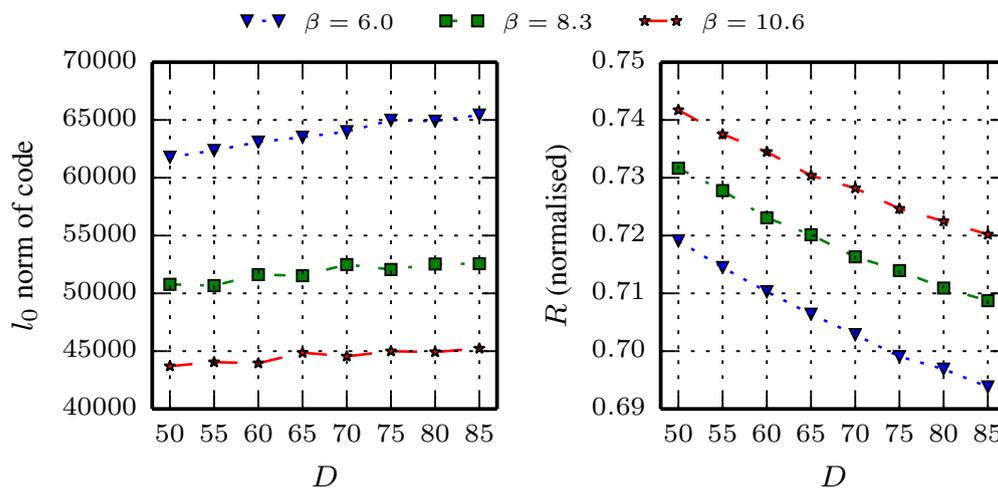


Figure 31: Number of coefficients used and normalised reconstruction error for the elite individual for various values of  $\beta$  and  $D$ . For comparison, the reference transcription contains 56377 phoneme instances.

Therefore, as a final step in producing a new individual, a local search is performed using the iterative realignment described in Section 3.3.4. The search terminates after the absolute per-iteration improvement  $\Delta C$  in the cost function falls below a certain threshold. This threshold is the same for all individuals, so that the fitness levels attained at the end of each search are more directly comparable. In this study the threshold was initially infinite, and after each generation, the median of the population's terminal  $\Delta C$  was used as the new threshold.

## 3.4 Experiments and results

### 3.4.1 Data selection and preprocessing

In all experiments that follow, a subset of the TIMIT dataset was used for training. In particular, we used the same 1386 SI training utterances that were used for the experiments in the previous chapter (see Section 2.4.1).

### 3.4.2 Experimental setup and training overview

In this study, a series of metaheuristic searches was applied to the training corpus in order to investigate the effect of the tunable parameters. The parameters in question were the number of prototypical basis functions  $D$  and the diversity penalty  $\beta$ . The remaining parameters such as the mutation rate  $\mu$ , the selection pressure  $m_a$  and the population size were fixed at values that appeared reasonable during initial informal testing.

Figure 31 summarises the elite individual found for each pair  $(\beta, D)$  on the training grid in terms of the number of sub-word units used by that individual in its transcription of the training audio, as well as the resulting normalised mean reconstruction error. From the figure it is clear that the diversity penalty still allows the intended trade off between reconstruction error and increased code sparsity. It is also observable that our search procedures are capable of using larger dictionaries to perform more accurate acoustic matching. In contrast, the development in the previous chapter indicated some numerical problems when solving for larger dictionaries. Furthermore, as the dictionaries become larger, with  $\beta$  held fixed, the number of coefficients used increases, implying that the average segment length decreases, i.e. that the learned basis functions start matching shorter acoustic events.

### 3.4.3 Performance of metaheuristic search

Figure 32 shows how the metaheuristic search influences the fitness distribution of a population over the course of generations. It is apparent that the algorithm manages to produce populations that consistently achieve higher fitnesses than previous generations. The generally smooth  $1/n$  progression may be attributed to the local search function combined with the adapting  $\Delta C$  threshold, whereas sudden jumps in fitness (most clearly visible at generation 12 and 16) are the result of fortuitous mutation.

Having shown that the metaheuristic search is effective, we would also like to show that it represents an improvement upon randomly guessed initial dictionaries. In order to do this, we saved the initial population of dictionaries for a subset of the experiments described in Section 3.4.2. For each experiment, these initial dictionaries were individually optimised using an exclusively local

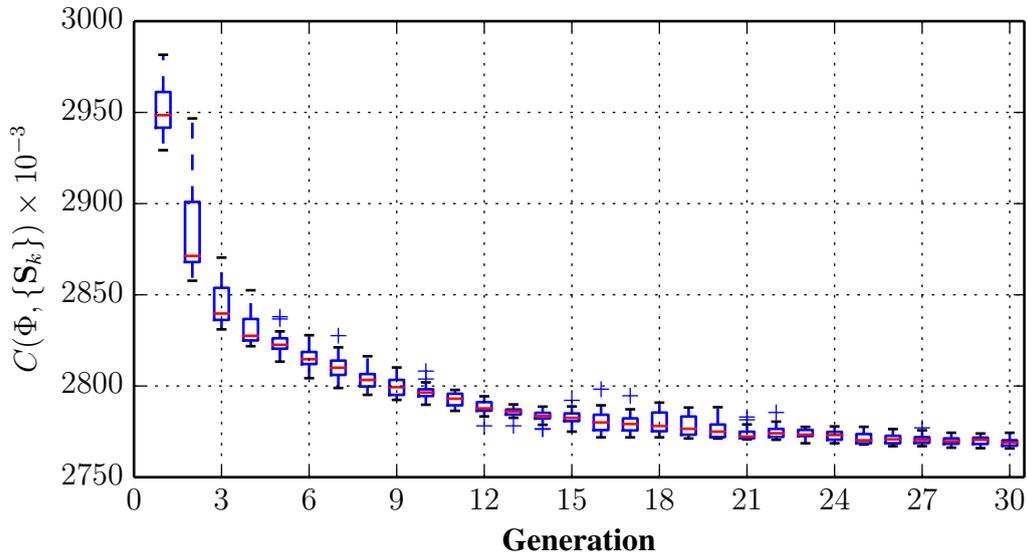


Figure 32: Generational development of the population fitness distribution for an experiment with  $\beta = 8.3$  and  $D = 55$ .

Table 31: Improvement made in the cost function by the metaheuristic search compared to pure local search as a multiple of the search termination threshold.

	$D = 55$	$D = 65$	$D = 75$
$\beta = 6$	36.63	36.74	38.15
$\beta = 8.3$	46.36	20.39	21.62
$\beta = 10.6$	32.15	24.87	60.03

search until the per-iteration reduction in the cost function fell below the terminal  $\Delta C$  threshold of the corresponding metaheuristic experiment.

Table 31 shows the absolute improvement (as a multiple of the terminal  $\Delta C$ ) made in the cost function by comparing the elite individual from the metaheuristic search to the elite individual resulting from iteratively improving the initial population. It is clear that the metaheuristic search consistently finds substantially better solutions. Figure 33 compares the terminal fitness distributions for the experiment that showed the greatest improvement when metaheuristic search was applied.

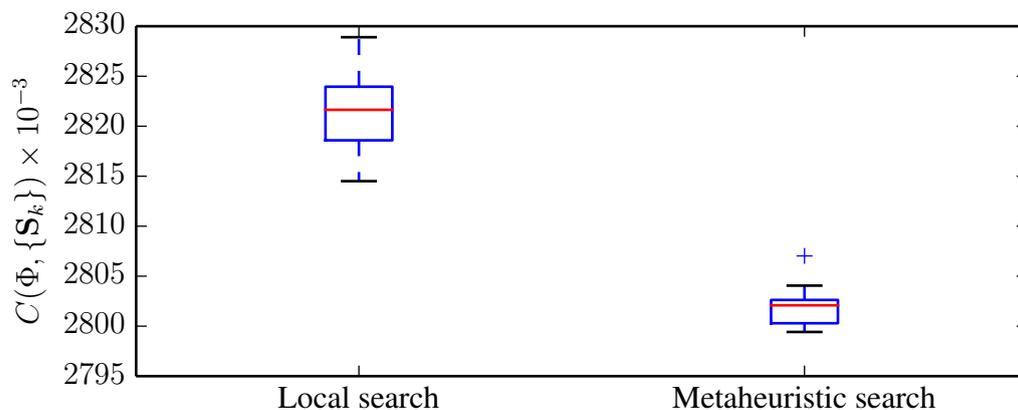


Figure 33: Terminal fitness distributions for an experiment with  $\beta = 10.6$  and  $D = 75$ .

### 3.4.4 Evaluation of basis functions as sub-word units

In this section we again evaluate whether the basis functions we learn in our experiments could be suited to the task of generating pronunciation dictionaries that can be used for ASR.

#### 3.4.4.1 Coincidence with reference phonemes

Figure 57 shows how our learned basis functions coincide with the reference phonemes described by TIMIT. Figure 34, in conjunction with informal listening tests, indicates that the clustering and segmentation is reasonably robust and acoustically meaningful. Many reference phonemes are strongly represented using only one or two basis functions. In addition, in cases where a single basis function is used to represent multiple reference phonemes, it is often because those reference phonemes are acoustically similar. This is in contrast with coincidence reported the previous chapter (see Section 2.4.3), where the concentration of basis functions used to code for a reference phoneme was much more dispersed.

#### 3.4.4.2 Pronunciation consistency

For our learned sub-word units to be useful for the generation of pronunciation dictionaries, it is necessary for the transcription of spoken words in terms of our units to be consistent. Ideally, every word in the training corpus would be

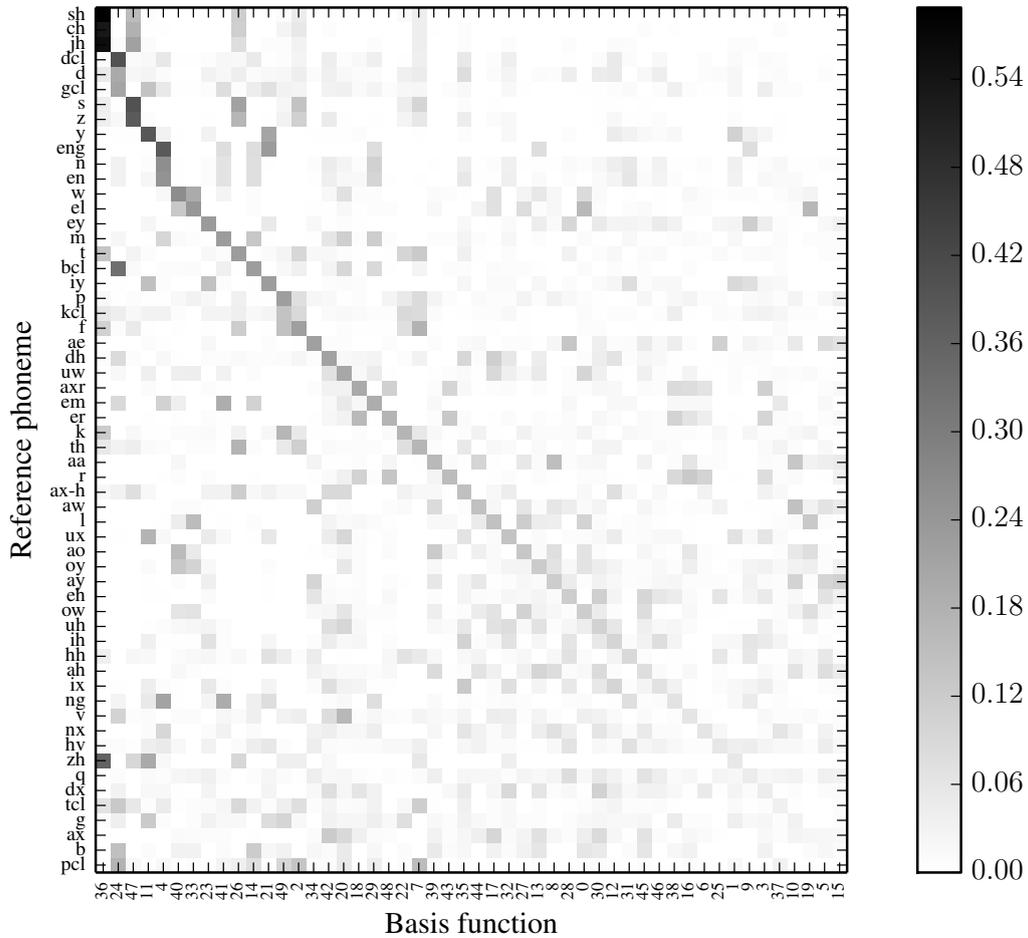


Figure 34: Coincidence of learned basis functions with reference phonemes for  $\beta = 10.6$  and  $D = 50$ .

transcribed by exactly one sequence of basis functions, or failing that, a highly compact set of sequences.

In order to evaluate how well our sub-word transcriptions perform in this respect, we use the time-aligned word boundaries accompanying the TIMIT dataset to extract a pronunciation for each occurrence of a word. We then calculate the cumulative fractions of the occurrences of each word that are transcribed using the top  $N$  pronunciations for that word. Figure 36 reports the weighted average fraction of occurrences for the top 3 pronunciations for a selection of the 20 most frequent words. In the most consistent case, an average of 25% of the occurrences of the selected words were represented using a set of just 3 pronunciations. This is quite poor compared to TIMIT's hand

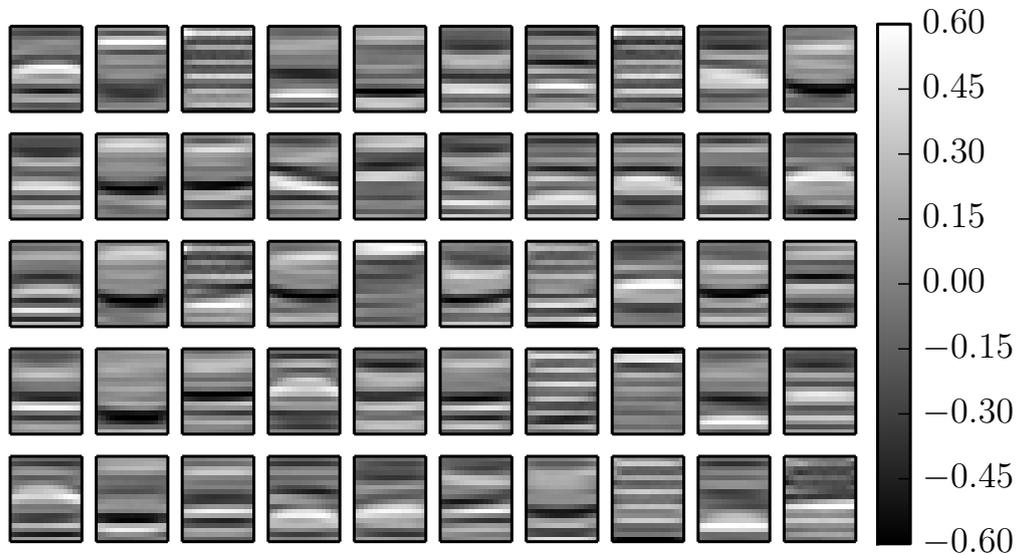


Figure 35: Mel spectrogram representation of the elite dictionary of sub-word units for  $\beta = 10.6$  and  $D = 50$ . Lower frequencies are at the bottom.

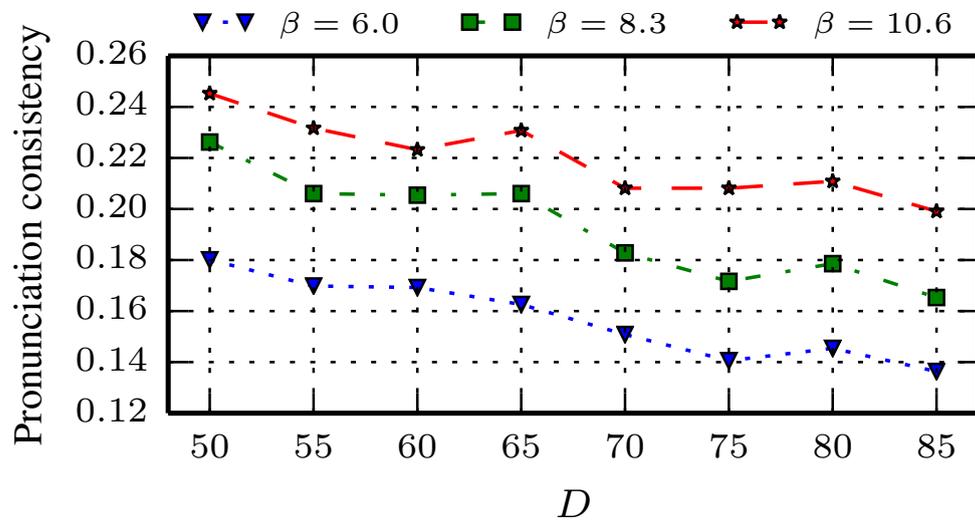


Figure 36: Weighted average of the fraction of occurrences of the 20 most frequent words transcribed by one of their respective top 3 pronunciations. TIMIT's reference transcription achieves 0.69.

Table 32: Pronunciation statistics for the most consistent set of sub-word acoustic units. Statistics for TIMIT’s reference transcriptions are in parentheses.

Word	# Occ.	# Pron.	Top pron.	Top 3 pron.	Top 5 pron.
<i>the</i>	508	55 (22)	14% (39%)	25% (81%)	34% (88%)
<i>a</i>	351	61 (20)	5% (36%)	15% (72%)	22% (81%)
<i>to</i>	269	86 (34)	7% (20%)	18% (43%)	26% (60%)
<i>of</i>	245	82 (25)	9% (35%)	20% (69%)	29% (80%)
<i>and</i>	226	102 (61)	8% (24%)	17% (36%)	24% (48%)
<i>he</i>	212	37 (10)	33% (63%)	51% (91%)	65% (96%)
<i>in</i>	184	75 (19)	8% (43%)	17% (71%)	25% (83%)
<i>is</i>	170	61 (17)	12% (46%)	29% (81%)	38% (88%)
<i>it</i>	150	54 (32)	12% (19%)	25% (45%)	35% (55%)
<i>his</i>	113	56 (14)	13% (27%)	32% (57%)	42% (78%)
<i>you</i>	106	37 (15)	25% (60%)	42% (76%)	54% (85%)
<i>this</i>	97	45 (12)	10% (72%)	26% (88%)	37% (93%)
<i>was</i>	96	69 (13)	5% (32%)	12% (73%)	19% (83%)
<i>they</i>	92	36 (5)	24% (89%)	37% (96%)	47% (100%)
<i>are</i>	92	51 (18)	8% (25%)	20% (57%)	28% (77%)

transcriptions, which achieve a fraction of 69%.

Table 32 reports in-depth pronunciation statistics for the most consistent experiment in Figure 36. It is clear that even hand transcription by experts yield a large amount of phonetic variation. For example, the word ‘and’ occurs 226 times in the training corpus, and is pronounced in 61 different ways. Given this level of inherent variability in speech and the current level of coding consistency, it is likely that a substantial number of instances of each relevant word will need to be available for training before our current regime will be able to confidently extract its prevailing pronunciation pattern.

### 3.5 Summary and conclusion

In this chapter, we have contributed a novel class of sparse codes for the specific task of unsupervised segmentation and clustering of speech. We also contributed the associated algorithms for the optimal calculation of the basis functions and the corresponding alignment with the audio features. Finally, we proposed a new optimisation strategy that embeds a local search within a metaheuristic search.

We found that the proposed metaheuristic search improved upon local search for the purpose of extracting acoustically relevant sub-word units from speech, on the basis of both an empirical cost function and informal listening tests. However, the resulting SWU transcriptions are still seen to be too inconsistent to be suitable to the task of inducing a lexicon. In the next chapter,

*CHAPTER 3. NON-OVERLAPPING SPARSE CODES AND METAHEURISTIC SEARCH* **51**

we attempt to further refine the discovered units and the consistency with which they are used to transcribe words.

## Chapter 4

# Refining SWUs with lattice-constrained Viterbi training

### 4.1 Introduction

In the previous chapter, we investigated the use of a shift and scale invariant sparse coding framework with non-overlapping basis functions for the unsupervised discovery of SWU inventories. This approach led to reasonable SWUs, but the transcription of the training utterances in terms of these units was generally inconsistent at the word level and as a consequence not directly useful for ASR. In this chapter, we investigate the application of novel lattice-constrained Viterbi training strategies to the task of improving the sub-word unit (SWU) inventories. The goal is to improve the SWU inventory and to extract more consistent transcriptions. Additionally, we make a first attempt at inducing a lexicon so that we can evaluate our techniques in terms of speech recognition accuracy and establish the usefulness of our SWU and pronunciation lexicon determination paradigm.

### 4.2 Motivation

The sparse coding approach presented in the previous chapter suffers from the following deficiencies which must be addressed to improve the quality of the discovered SWU inventories:

1. Sparse coding basis functions can only warp linearly, while speech generally warps non-linearly;
2. silences and pauses are not explicitly modelled;
3. no attempt is made to discover an underlying linguistic pattern in the sequential use of the discovered SWU's to transcribe speech, which could be reinforced to create more consistent transcriptions; and
4. the approach can not be easily extended to take advantage of knowledge about the particular word sequence that is known to be present in the utterance under consideration.

In the following sections we address these points in order to improve the quality of the discovered units and resulting transcriptions.

### 4.3 Acoustic modeling with HMM-GMMs

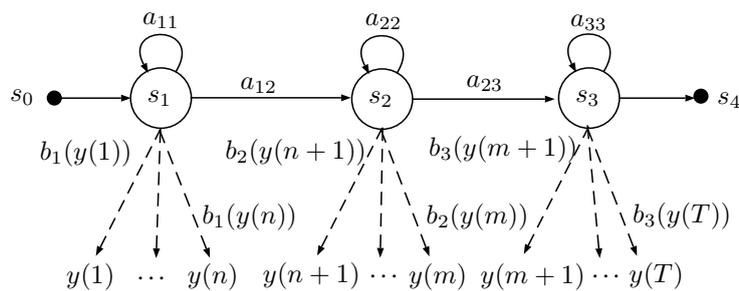


Figure 41: Three-state HMM used for acoustic modeling.

The first deficiency of sparse coding that we have identified is that the basis functions can only warp linearly, while speech generally warps non-linearly. This can be addressed by modelling each acoustic realisation  $\mathbf{y}$  of a SWU as generated by a  $n$ -state left-to-right hidden Markov model (HMM) with a Gaussian mixture model (GMM) governing each state's emission probabilities. The choice  $n = 3$  has been typical for this model in ASR applications for many years (see Figure 41). The alignment between the frames  $\mathbf{y}(1), \mathbf{y}(2), \dots, \mathbf{y}(T)$  of a speech fragment  $\mathbf{y}$  and a model  $M$  is described by a latent and arbitrary state sequence

$S = s(1), s(2), \dots, s(T)$ , which allows the model flexibility to adjust to any time distortion in the acoustic realisation.

The likelihood of  $\mathbf{y}$  being generated by  $M$  and a known state sequence  $S$  is given by:

$$P(\mathbf{y}, S|M) = \prod_{t=1}^T a_{s(t), s(t+1)} b_{s(t)}(\mathbf{y}(t)), \quad (4.1)$$

where  $a_{i,j}$  are the probabilities of transitioning from state  $i$  to state  $j$ , and  $b_i(\mathbf{y}(t))$  is the likelihood of the observation vector  $\mathbf{y}(t)$  being emitted by state  $i$ . In this case the emission probabilities are modeled by a GMM:

$$b_i(\mathbf{y}(t)) = \sum_{m=1}^{M_s} c_{im} \mathcal{N}(\mathbf{y}(t); \mu_{im}, \Sigma_{im}), \quad (4.2)$$

where  $c_{im}$  is the  $m$ 'th mixture weight for state  $i$  and  $\mathcal{N}(\mathbf{y}; \mu, \Sigma)$  is a multivariate Gaussian distribution with mean  $\mu$  and covariance  $\Sigma$ .

This modeling approach differs from sparse coding in the sense that we are now concerned with maximising the likelihood that a speech signal is generated by a sequence of models, rather than by how well the signal is reconstructed from a sequence of basis functions. There is a potential information loss incurred by the imposition of this, since the prototype basis functions used during sparse coding could capture many more frames of temporal information than the HMM's used in speech typically have states. However, this loss is compensated for by the ability of GMM's to capture much more of the observed acoustic variance at each state.

### 4.3.1 Silence modeling

Another weakness of the sparse coding approach used in previous chapters is that silences and pauses are not explicitly modeled, which means one or more of the units in the SWU inventory would need to perform that function implicitly. In this chapter, we address silences and pauses explicitly, by adding units modelling silences and pauses to the SWU inventory. This increases the chances that the SWU inventory is reserved for actual SWUs.

The topologies of the silence (sil) and short pause (sp) models are depicted in Figure 42. This topology has been used extensively in HMM-GMM base ASR systems [1]. The *sil* model is intended to represent longer non-speech segments,

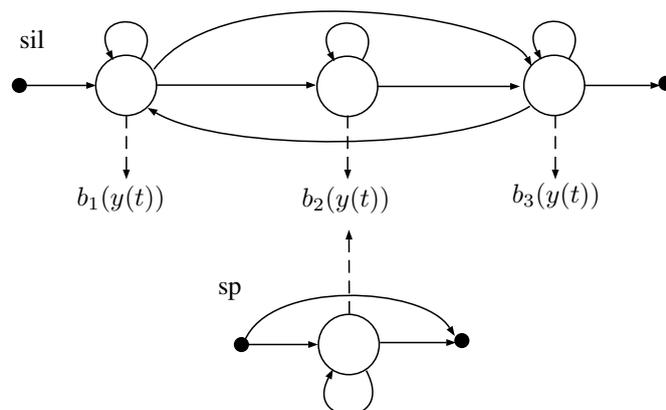


Figure 42: Topology of silence (*sil*) and short pause (*sp*) acoustic models [1].

especially at the start and end of utterances, while *sp* models shorter inter-word pauses. Note that for the sake of more robust parameter estimation, the emission distribution of the central state of the *sil* model is shared with the *sp* model. In order to train the *sil* unit, we can take advantage of the fact that there is a large likelihood that silences occur at the beginning and end of an utterance. The *sp* unit has a transition between the non-emitting start and end states, which allows making the model mandatory between words without necessarily aligning with any acoustic frames.

### 4.3.2 Viterbi training

In order to jointly train both the SWU inventory and the resulting SWU transcriptions, we make use of a similar strategy to the one used for sparse coding and dictionary learning. This involves successively updating either the SWU transcription or the SWU acoustic models (i.e. HMM-GMM parameters), while the other is fixed. In the context of HMMs, this strategy is called *Viterbi training*, so named because the single most likely SWU sequence (determined by means of the Viterbi algorithm), for each speech utterance is used to update the HMM-GMM parameters. Viterbi training is a computationally efficient alternative to the more accurate expectation-maximization (EM) procedure, which would marginalize over every possible SWU sequence when updating the SWU acoustic models.

### 4.3.2.1 Obtaining SWU transcriptions

In the case where the speech sample  $\mathbf{y}$  represents a single isolated SWU, the task of classifying  $\mathbf{y}$  is simply to pick the SWU  $u_k$  corresponding to model  $M_k$  with the largest likelihood of generating  $\mathbf{y}$ :

$$\arg \max_k \{P(\mathbf{y}|M_k)\}, \quad (4.3)$$

where  $P(\mathbf{y}|M_k)$  can be efficiently calculated by the forward algorithm.

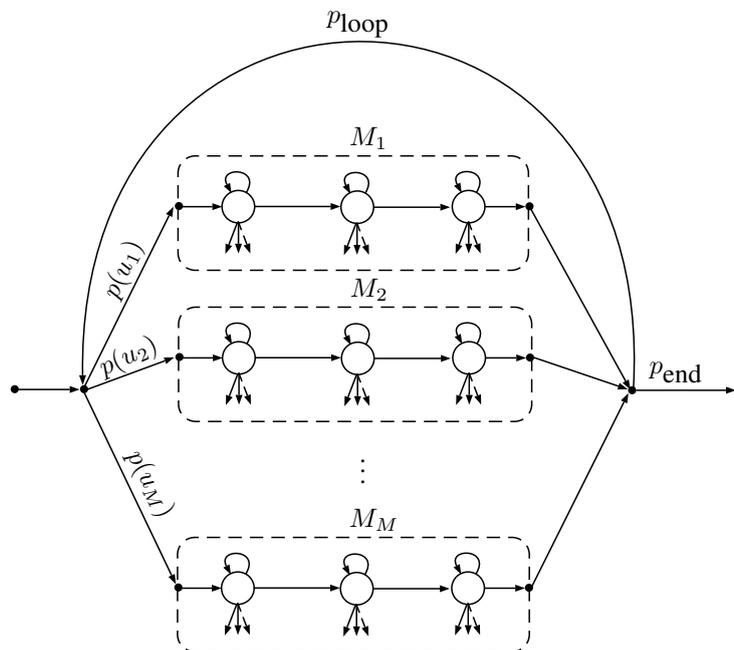


Figure 43: Topology of composite phone-loop HMM.

In general, however, we are interested in decoding an utterance-level speech sample into a sequence of SWUs. In order to do this, we need to combine the models of the individual SWUs into a larger, composite HMM. The simplest of these is known as a *phone loop* (see Figure 43), where every SWU can follow any other. This model includes transition probabilities  $p(u_n)$  that govern the relative occurrences of each SWU, and which we will refer to as the *pronunciation model*. The pronunciation model, as well as the loop-back probability  $p_{loop}$  together act as constraints on the nature of the decoded SWU sequence. We will refer to this structure as the *decoding lattice*.

With a decoding lattice  $M$  in place, we can use the Viterbi algorithm to find the most likely state sequence  $S$  (and directly from that the SWU sequence) given a speech utterance:

$$\operatorname{argmax}_S \{\log P(\mathbf{y}, S|M)\}. \quad (4.4)$$

We can write the optimisation objective of Equation 4.4 in terms of the decoded SWU sequence  $\{u_1, u_2, \dots, u_N\}$ :

$$\operatorname{argmax}_{\{u_1, u_2, \dots, u_N\}} \left\{ \sum_{n=1}^N [\log p(\mathbf{y}_n|M_n) + \log p(u_n) + \log p_{\text{loop}}] \right\}, \quad (4.5)$$

where  $p(\mathbf{y}_n|M_n)$  is the likelihood of  $\mathbf{y}_n$ , the subsequence of  $\mathbf{y}$  that aligns with  $u_n$ , being generated by the HMM corresponding to  $u_n$ . The term  $\log p(\mathbf{y}_n|M_n)$  is known as the *acoustic score*, while we refer to  $\log p(u_n)$  as the *pronunciation score*. The term  $\log p_{\text{loop}}$  is added for every SWU in the sequence, and therefore acts as a sparsity constraint on the length of the decoded SWU sequence.

Generally, the dynamic range exhibited by the acoustic scores is much larger than that of the pronunciation scores. It is therefore necessary to scale the scores in the decoding lattice to ensure that they materially affect the produced SWU transcriptions:

$$\operatorname{argmax}_{\{u_1, u_2, \dots, u_N\}} \left\{ \sum_{n=1}^N [\log p(\mathbf{y}_n|M_n) + \alpha \log p(u_n)] + \beta N \right\}, \quad (4.6)$$

Further, we observe that the penalties imposed by the pronunciation scores tend to encourage the acoustic decoder to produce transcriptions that use fewer, longer sub-word units. In some cases, spurious insertions are also observed. In order to stabilize the average SWU production rate, we make use of an insertion penalty/reward  $\beta$  which is tuned on a small set of samples to target a specific average SWU length.

#### 4.3.2.2 Estimating SWU parameters

Once an SWU transcription has been obtained for every utterance in the training corpus, the SWU HMM-GMM parameters are updated by applying several iterations of the embedded Baum-Welch reestimation algorithm.

## 4.4 Lattice-constrained Viterbi training

The use of a decoding lattice, as described in Section 4.3.2.1, allows us to impose external constraints on the resulting SWU transcriptions. In this section we propose strategies to constrain the SWU transcriptions in a way that attempts to make them more consistent. Further, we propose jointly learning these sequence constraints along with the SWU parameters, by updating their parameters alongside the SWU parameters during each iteration of the Viterbi training. In this way the quality of the SWU inventory has the potential to be much higher, since their parameter updates benefit from increasingly more consistent transcriptions. Below, we discuss two strategies to constrain the produced SWU sequences. The work presented in [9; 12] also employs Viterbi training to improve SWU inventories and transcriptions, with [9] incorporating the learning of a language model at the utterance level. However the word-level lattice-based constraints that we describe below, are, as far as we know, novel.

### 4.4.1 Bigram constrained Viterbi training

The third deficiency of the sparse coding approach described in the previous chapter is that it does not attempt discover and reinforce an underlying linguistic pattern in the sequential use of the discovered SWU's to transcribe speech. This can be dealt with by attempting to jointly learn an N-gram SWU language model along with the SWU inventory, by iteratively reestimating the language and acoustic models from the produced SWU transcriptions, and then using those language and acoustic models to produce new transcriptions. We hypothesise that this could reinforce the use of more frequent SWU sequences, while diminishing the use of rare sequences and in doing so result in more consistent transcriptions.

### 4.4.2 Word-level SWU pronunciation modeling

The last deficiency identified in Section 4.2 will be addressed by incorporating knowledge from the orthographic transcriptions accompanying the speech audio by attempting to learn an SWU pronunciation model for each word in the training corpus, and then constraining the transcription of each utterance by a

decoding lattice formed by chaining the word models of each utterance together (see Section 4.5 for more detail). This would allow pronunciation knowledge to be aggregated from all instances of a word and encourage all SWU transcriptions of that word to become more consistent. Since this approach requires word transcriptions of the training data, it can be considered lightly supervised. This is similar to the work in [8], which uses HMM's with unigram SWU emission probabilities for pronunciation modelling, but stops short of jointly learning these models while producing new SWU transcriptions.

## 4.5 Pronunciation modeling

In this section we describe the approach taken in this chapter to model word pronunciations. We follow the approach proposed in [8], of modelling each word  $w_j$  in the vocabulary as a single-state HMM (shown in Figure 44) emitting SWU's according to a unigram word pronunciation model  $p(u_i|w_j)$ . The self-transition probability  $a_s$  and word-exit transition probability  $a_e$  can be thought of as governing the length of the word in terms of the number of SWU's used to pronounce it.

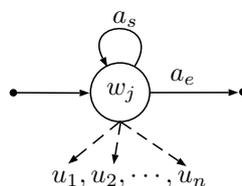


Figure 44: Single-state word HMM.

These single-state HMM's can subsequently be chained according to the word order of a particular utterance to form a composite HMM, which can be used to perform forced alignment or embedded reestimation.

### 4.5.1 Model initialisation

We choose uniform distributions for the initial  $p(u_i|w_j)$ . The transition probabilities of word  $w_j$  with character length  $n_j$  are set to

$$a_s(w_j) = \frac{n_j}{n_j + 1}, \quad a_e(w_j) = 1 - a_s(w_j). \quad (4.7)$$

This initialisation ensures that longer words have higher self-transition probabilities than shorter words.

### 4.5.2 Model reestimation

Given a set of observed SWU sequences, as well as some prior estimates for our word pronunciation models, we can produce updated model parameters by applying a number of iterations of embedded Baum-Welch reestimation.

One of the estimates that is produced as part of the Baum-Welch algorithm, is the expected number of times  $\gamma_{i,j}$  that an SWU  $u_i$  is aligned with word  $w_j$ . This estimate can be used directly to produce the updated SWU emission models  $p'(u_i|w_j)$ :

$$p'(u_i|w_j) = \frac{\gamma_{i,j}}{\sum_i \gamma_{i,j}}. \quad (4.8)$$

However, many words in the vocabulary occur very infrequently, leading to very poor (and overly confident) estimates in those cases. To combat this, we apply add-one smoothing:

$$p'(u_i|w_j) = \frac{\gamma_{i,j} + 1}{\sum_i \gamma_{i,j} + N}. \quad (4.9)$$

This affects infrequent words disproportionately, since their expected counts will be smaller, effectively *backing off* to the uniform distribution, whereas the counts of frequent words are less affected.

### 4.5.3 Lattice structured pronunciation model

Each word's pronunciation is encoded as a bigram SWU lattice, where unigram scores are used regardless of context, but a bigram structure is enforced to disallow immediate repetitions of a SWU within a word. The word's self-transition score  $a_s$  is applied after every SWU insertion while the word's exit-transition score  $a_e$  is applied after the final insertion.

Finally, for each utterance in the training corpus, the corresponding orthographic transcription is used to chain together the corresponding pronunciation lattices to form an utterance level lattice. Each word pronunciation lattice must be followed by either a silence or an optional short pause. These allow the acoustic decoder to capture between-word pauses. The mandatory silence

or short pauses also provide a marker in the SWU transcription which indicate word boundaries in the lattice. This allows word pronunciations to be directly extracted from the transcription without performing a separate subsequent Viterbi decode pass.

#### 4.5.4 Combined training procedure

Once we have obtained word pronunciation models, we can use this knowledge to produce refined SWU transcriptions of the training data. In order to present the word pronunciation models to the speech recogniser, we encode its parameters into a word pronunciation lattice as shown in Figure 45. The short pause and silence models at the end of each word lattice allow the acoustic decoder to insert a silence before transitioning to the next word. The word-level sub-lattices are then chained into utterance-level lattices and presented, along with the SWU acoustic models, to a speech recogniser to produce new SWU transcriptions.

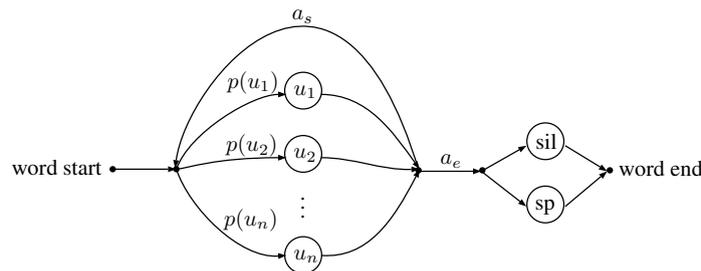


Figure 45: Word pronunciation lattice model used to constrain SWU transcription.

With these steps in place, we can use Viterbi training to accomplish complete joint SWU and pronunciation model learning as follows:

1. With the SWU transcriptions fixed, produce updated pronunciation and SWU acoustic models.
2. Produce new SWU transcriptions with the updated pronunciation and acoustic models.
3. Repeat steps 1 and 2 until some termination criteria are met.

## 4.6 Evaluating the SWU inventory

In this section we perform some experiments and evaluate the effect of the lattice-constrained Viterbi training strategies, described in the previous sections, on the intrinsic quality of the discovered SWU inventory and resulting pronunciation lexicons.

### 4.6.1 Experimental setup

We again used the 1386 SI training utterances of the TIMIT corpus for experimental evaluation (see Section 2.4.1). The selected utterances were converted with HTK [1] to 39-dimensional feature vectors, consisting of twelve MFCCs, with the addition of log energy, and first and second order differential coefficients. In order to facilitate comparison with reference phonemes, the SWU rate-controlling parameters (i.e. sparse coding penalty and Viterbi decode insertion penalty  $\beta$ ) were chosen to produce units comparable in duration with phonemes, although this is not necessarily an optimal choice.

### 4.6.2 Coincidence with reference phonemes

The left-hand column of Figure 46 shows the coincidence between our SWU inventories and the TIMIT reference phonemes. These 2D coincidence histograms are computed by counting the number of times at least 50% of the span of one of our SWU's occurs within the boundaries of a reference phoneme in the TIMIT transcriptions. Further, each row is normalised to show the fraction of occurrences of a phoneme that is coded by a particular SWU.

These coincidence histograms illustrate the correspondence of our SWU inventories with those chosen by phonetic experts. However, it would be better if we could objectively quantify the correspondence in some way. To do this, we turn to two figures of merit: the entropic coding efficiency of our SWU's and the mutual information between the SWU's and the reference phonemes.

### 4.6.3 Weighted mean entropic coding efficiency

The weighted mean coding entropic coding efficiency is calculated by taking a weighted mean of the entropic efficiency of each reference phoneme's coincidence distribution with our set of SWU's:

$$\bar{\eta}_w = \sum_j p(\phi_j) \eta_j, \quad (4.10)$$

where  $\eta_j$  is the entropic coding efficiency of the  $j$ 'th phoneme

$$\eta_j = \frac{H(u|\phi_j)}{H_{\max}} \in [0, 1]. \quad (4.11)$$

The values of  $\eta_j$  for the experiments in this study are shown in the right-hand column of Figure 46. The term  $H(u|\phi_j)$  refers to the conditional entropy of the distribution  $p(u|\phi_j)$ :

$$H(u|\phi_j) = - \sum_i p(u_i|\phi_j) \log_2(p(u_i|\phi_j)), \quad (4.12)$$

where  $u_i$  is the  $i$ 'th unit in our SWU inventory and  $\phi_j$  is the  $j$ 'th phoneme in the reference set.  $H(u|\phi_j)$  can be interpreted as a measure of how spread out the corresponding conditional distribution is, ranging between zero where only one SWU is used to code the given phoneme, and  $H_{\max}$ , when all SWU's coincide equally with that phoneme. Thus, if a good correspondence with the reference phonemes is desired,  $\bar{\eta}_w$  must be minimised.

### 4.6.4 Coding coincidence mutual information

As an additional figure of merit, we consider the mutual information between the incidence of the reference phonemes  $\{\phi_j\}$  and our set of SWU's  $\{u_i\}$ :

$$I_m(u; \phi) = \sum_i \sum_j p(u_i, \phi_j) \log_2 \frac{p(u_i, \phi_j)}{p(u_i) p(\phi_j)}. \quad (4.13)$$

The mutual information  $I_m(u; \phi)$  is maximised when the random variables  $u$  and  $\phi$  uniquely determine each other, i.e. when each reference phoneme corresponds to exactly one SWU.

The joint distributions are obtained by counting the number of times at least 50% of the span of one of our SWU's occurs within the boundaries of a reference phoneme in the TIMIT transcriptions. The mutual information  $I_m(u; \phi)$  is maximised when the random variables  $u$  and  $\phi$  uniquely determine each other, i.e. when each reference phoneme corresponds to exactly one SWU.

### 4.6.5 Pronunciation consistency of extracted lexicon

Finally, we consider the consistency with which our SWU inventories transcribe the input audio into word pronunciations. We use the time-aligned word transcriptions included in TIMIT to extract word-level pronunciations from the SWU transcriptions to form a lexicon. This lexicon is then evaluated in terms of its average pronunciation entropy  $\bar{H}_p$ , as defined by Lee et al. [24]:

$$\bar{H}_p = \frac{-1}{|V|} \sum_{w \in V} \sum_{b \in B(w)} p(b) \log_2 p(b), \quad (4.14)$$

with  $V$  the vocabulary of the task and  $B(w)$  the observed pronunciations for word  $w$ .

The average pronunciation entropy provides a measure of the variation and spread of the pronunciations in the lexicon. It will be lower when there is a compact, dominant set of pronunciations for each word.

## 4.7 Results

Table 41 summarises the results of the following experiments:

1. Baseline SWU transcriptions determined using the sparse coding approach described in the previous chapter.
2. 40 iterations of sequence-level bigram constrained Viterbi training (SLB) as described in Section 4.4.1.
3. 40 iterations of word-level unigram constrained Viterbi training (WLU) as described in Section 4.4.2.
4. 40 iterations SLB Viterbi training followed by 40 iterations of WLU Viterbi training.

In all cases, the models were initialised from the sparse coding SWU inventory and transcriptions, and the HTK tools were used for acoustic modelling and decoding.

It can be seen that we have produced substantially improved SWU inventories in all cases. However, it is hard to pick a clear winner from the

Table 41: Summary of experimental results

Experiment	$\bar{\eta}_w$	$I_m$ (bits)	$\bar{H}_p$ (bits)
1) Baseline	0.655	2.030	2.383
2) Baseline + SLB	0.529	2.622	2.380
3) Baseline + WLU	0.501	2.644	2.266
4) Baseline + SLB + WLU	0.501	2.750	2.314
TIMIT reference transcript	—	—	1.180
CMUDICT	—	—	0.181

approaches examined here. In terms of inventory quality (i.e. entropic coding efficiency and reference phoneme mutual information) both approaches work equally well, which is promising, since the sequence-level bigram training is fully unsupervised.

In terms of pronunciation consistency, the WLU system performs best, while the combined SLB + WLU system does slightly worse. However, the overall improvement seen in pronunciation consistency was not as great as anticipated. This may be a symptom of an overly simplistic pronunciation model, which models only the frequency of incidence of SWU's, and not their order. In order to put  $\bar{H}_p$  into context, we also include this figure for a lexicon extracted from TIMIT's phone transcriptions, as well as for a hand-crafted lexicon defined by experts (CMUDICT).

## 4.8 Automatic speech recognition

In this section we evaluate the ASR performance achieved using the lexicons resulting from the lattice constrained training procedure described in Section 4.5. In the previous section we found the strategies developed in the chapter to be effective at improving the quality of the SWU inventory in terms of its mutual information with reference phonemes. However, the pronunciations extracted directly from that stage still suffer from alignment errors and excessive variation. In the sections below, we describe new strategies to improve the quality of the extracted lexicons and then use those lexicons to perform recognition.

### 4.8.1 Pronunciation extraction

One way to cope with unwanted pronunciation variation, is to iteratively align the acoustic features of the training data with our derived dictionary and then to prune out the most infrequent pronunciations of each word until only a small number of dominant pronunciations remain. However, that step relies on the number of misaligned hypotheses being small, since it otherwise leads to the dominance of correct fragments, rather than whole pronunciations.

The first of our pronunciation extraction strategies relies on the observation that word frequencies are distributed according to Zipf's law, which implies that, for the most frequently occurring words, we have a disproportionate abundance of evidence of possible pronunciations. These words would also be less affected by misaligned pronunciation hypotheses that can lead to pathological pruning effects. Although these words also exhibit a lot of variation in observed pronunciations, we can be reasonably confident that some of the dominant pronunciations are likely correct.

It therefore makes sense to initially only extract and prune the pronunciations of a few of the most frequent words. We can then fix these pronunciations in the decoding lattices, while still constraining the other words by their pronunciation models. The fixing of the pronunciations of even just a few of the most frequent words results in a dramatic reduction in the branch factor of the lattices used for further pronunciation refinement and extraction. Crucially, as long as we have chosen good pronunciations for the initial batch of fixed words, subsequent decoding passes should yield fewer misalignments for the remaining words adjacent to fixed words.

We therefore propose a hybrid pronunciation learning and extraction approach:

1. Initially, all words are constrained only by their pronunciation models and we perform several passes of the joint training procedure described in Section 4.5.
2. Once the pronunciation models no longer improve, we prune and fix the  $N$  most frequent words.

3. We then perform several more passes of joint SWU pronunciation model training, in order to allow the system to refine pronunciations of unfixed words.
4. Steps 2 and 3 are iterated, while fixing the pronunciations of progressively larger sets of frequent words.

The pronunciation pruning performed in this study comprises of two separate passes. The first aims to reject some of the misaligned pronunciation hypotheses. We achieve this by estimating the length of each word in terms of SWUs by taking a weighted mean of the lengths of the hypothesized pronunciations. We then reject all pronunciation hypotheses that are not within a predetermined interval of the estimated word length. The second pruning pass attempts to reduce the number of hypothesized pronunciations of a word by rejecting the most infrequent pronunciations of a word until the remaining pronunciations account for a predetermined fraction of all occurrences of a word. These two pruning passes are iterated in tandem with a forced alignment pass in order to quickly reduce the number of remaining pronunciations.

#### **4.8.2 Joint-multigram pronunciation refinement**

Up to this point, we have not attempted to exploit the graphemic information available in the word transcriptions of our training data. This is intentional, since we aim to develop techniques that are useful for performing ASR on under-resourced languages, where a graphemic writing system may not be available. However, graphemes represent a potent source of phonetic information for languages where they are established, and it would be unwise to ignore their utility in constructing pronunciation dictionaries.

Thus, as a second pronunciation extraction procedure, we attempt to train joint-multigram grapheme-to-SWU models from our induced SWU sequences, and use those models to generate pronunciation hypotheses. In this way, pronunciation knowledge can be discovered and aggregated at the grapheme rather than word level. This approach should also yield better pronunciations for words that are seen infrequently, because in these cases the pronunciation regularity among grapheme sequences of a language will provide assistance.

In this study, we use the Sequitur G2P tool to perform the grapheme-to-SWU (G2SWU) modeling [61].

We initialize the G2SWU models by presenting the raw utterance level SWU sequences and utterance level orthography as training samples. We thus allow Sequitur to perform its own utterance level grapheme to SWU alignment, and avoid contaminating the G2SWU models with potentially poor alignment choices made earlier.

We then apply the trained G2SWU models to the training vocabulary in N-best mode, in order to gather multiple pronunciation hypotheses. These hypotheses are then presented to the SWU recogniser, and evidence is gathered about the relative popularity of the hypothesized pronunciations. This updated evidence is used to update the G2SWU models. We can perform several passes of these steps, incrementing the G2SWU model order after each pass. In order to obtain a final pronunciation dictionary, we can again apply the iterated pruning and forced alignment described in the previous section.

### 4.8.3 Experimental setup

We have employed the 3696 SI+SX training utterances of the TIMIT corpus to train a continuous speech recogniser so that we can evaluate the performance of our SWUs and associated pronunciation lexicons in terms of the word error rate (WER). Table 61 summarizes the datasets used in the speech recognition experiments. The training set consists of all the SI and SX utterances in TIMIT. The development and test sets were constructed as subsets of TIMIT's SI test partition, such that there were no speaker overlap between development and testing. Since every SI utterance is unique, there was also no prompt overlap. A bigram language model was constructed from the Brown corpus (excluding all dev and test prompts) as well as all the TIMIT training prompts.

Table 42: Overview of datasets used for ASR

<b>Set</b>	<b># Utts.</b>	<b># Speakers</b>	<b>Hours</b>
<i>Train</i>	3696	462	3.15
<i>Development</i>	153	51	0.15
<i>Test</i>	351	117	0.33

#### 4.8.4 Results and discussion

Table 43: Summary of experimental results

<b>Experiment</b>	$I_m$ (bits)	<b>WER</b>
<i>Sparse coding baseline</i>	2.030	–
<i>Grapheme recogniser</i>	–	77.40 %
<i>Monophone recogniser</i>	–	56.12 %
<i>Blind extraction</i>	2.523	83.94 %
<i>+ G2SWU refinement</i>	2.940	66.30 %

In this section, we evaluate four competing recognizers. The first two of these are considered baseline systems: a grapheme-based recognizer that employs letters as sub-word units and a monophone system that used traditional phonemes and an established pronunciation dictionary (CMUDICT). We then compare the performance of these systems to that achieved by systems trained using the automatically-determined sub-word units and the associated automatically-induced pronunciation dictionaries. Two such systems are considered:

1. a system derived by the blind pronunciation extraction approach described in Section 4.8.1, and
2. a system derived by the grapheme to SWU joint-multigram pronunciation refinement step described in Section 4.8.2, which was applied to the output of the previous experiment.

We also compare the SWU inventories produced by the two experiments described above, to the initial SWU inventory determined through sparse coding, in terms of their mutual information with the reference phonemes provided with TIMIT.

Table 43 summarises the results of the experiments performed in this study. It can be seen that all recognition results are quite poor. This can be ascribed to some extent to the dataset used in this study, since TIMIT is generally regarded as intended for phoneme recognition experiments rather than continuous word recognition. Further performance degradation can be ascribed to the use of

context independent units such as graphemes and monophones. However, the intent of this investigation is not to produce and compare state-of-the-art word error rates. Rather, it is to demonstrate that our automatically determined SWUs are at least in principle useful for ASR.

From Table 53, it is clear that the pronunciation dictionary determined through blind pronunciation extraction performs worst, although it remains comparable to the performance achieved with a grapheme-based pronunciation dictionary. The table also shows that, when we apply the G2SWU refinement to the SWU transcriptions produced through blind pronunciation extraction, we achieve a large performance gain.

The ASR performance of blind extraction with G2SWU refinement system compares favourably to that of a system built using expert-defined monophones. This indicates that our SWU inventories are of suitable quality for ASR, although our ability to extract pronunciations in terms of these SWUs without resorting to a mapping to graphemes is still lacking. Informal inspection of the pronunciation dictionaries produced by blind extraction confirms this. Our system extracts high-quality pronunciations for short, frequently occurring words, but experiences frequent misalignments and spurious deletions or insertions for less frequent to moderately frequent words.

Another factor that likely degraded the performance achieved by blind pronunciation extraction, is the inclusion of the SX training partition of TIMIT, which consists of prompts that are repeated by seven different speakers. This led to repeated co-occurrences of otherwise relatively rare word pairs being observed, degrading the pronunciation model's ability to distinguish between the adjacent words in these cases. We give credence to this hypothesis by observing pathological and systematic misalignment in some moderately frequent words, where fragments of the pronunciation of preceding or following words are included.

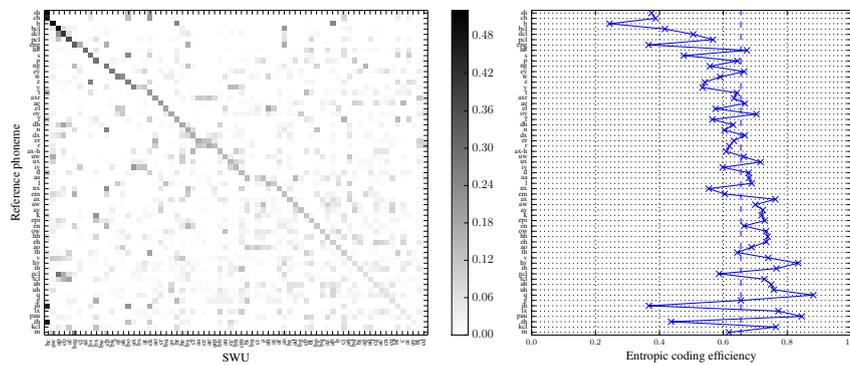
## 4.9 Summary and conclusion

We proposed two novel lattice-constrained Viterbi training strategies for the refinement of automatically-induced SWU inventories and transcriptions. The first of these strategies attempts to jointly learn a bigram SWU language model

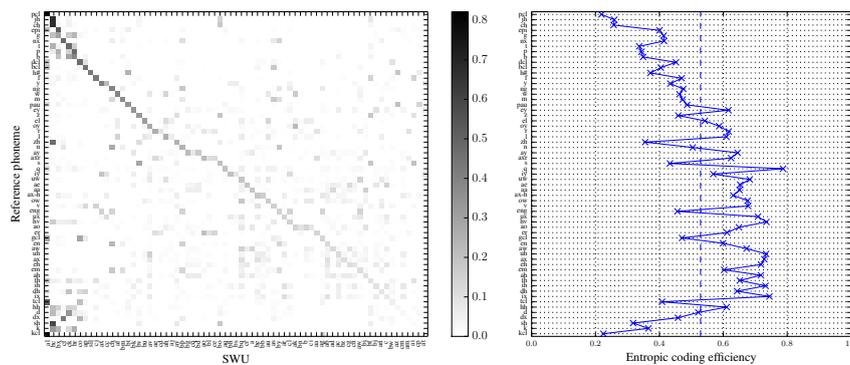
along with the evolving SWU inventory, while the second approach attempts to jointly infer an SWU pronunciation model for each word in the vocabulary, and to constrain transcription using these models. We found that both approaches yielded substantial increases in correspondence with reference phonemes and we were able to extract more consistent pronunciations from the transcriptions.

We also investigated the application of these refined units to the tasks of pronunciation generation and continuous speech recognition. We have presented a novel pronunciation pruning and fixing strategy that makes the most of our ability to confidently extract pronunciations for the most frequent words in a language. We have also shown that, by mapping our SWU's to grapheme sequences rather than words, we can achieve good recognition results, compared to a monophone system.

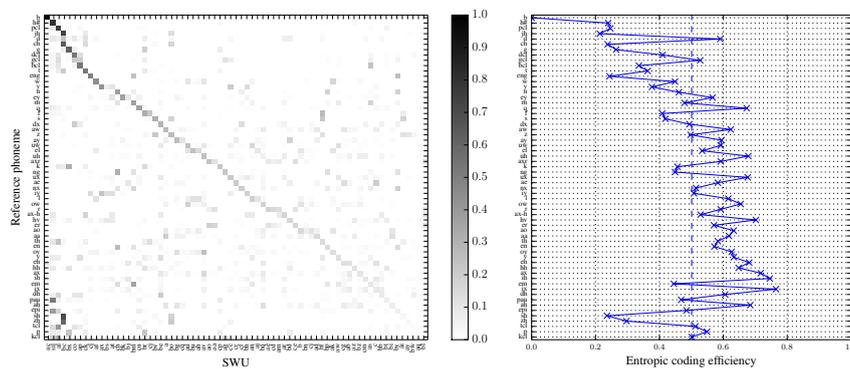
In the next chapter we explore a more sophisticated pronunciation modeling approach, in order to achieve ASR performance that is closer to what can be obtained using systems trained using hand-crafted lexicons.



(a) Baseline sparse coding SWU inventory



(b) After 40 iterations of sequence-level bigram constrained Viterbi training



(c) After 40 iterations of word-level constrained Viterbi training

Figure 46: Comparison of phoneme-SWU coincidence matrices and their corresponding entropic coding efficiencies for experiments 1, 2, and 3 in Table 53. The dashed lines show the weighted mean coding efficiencies.

# Chapter 5

## Lexicon induction for under-resourced languages

### 5.1 Introduction

The objective of this chapter is to demonstrate the feasibility of performing ASR with an automatically induced pronunciation lexicon in a truly under-resourced setting. Many studies have addressed parts of this problem (as detailed in Section 1.4), but these typically focus only on unit segmentation and discovery, while neglecting the subsequent development of pronunciation lexicons that are needed to perform ASR. Other studies have focused on designing lexicons, but have typically adopted pre-existing phonemes or alphabetic graphemes for use as sub-word units (SWU's). In the studies that have applied automated pronunciation lexicon design with automatically discovered SWU's to ASR, the datasets used are low-perplexity (e.g. connected digits or weather query corpora) or represent ideal scenarios such as read speech that may not be representative of the results that would be obtained in an under-resourced large-vocabulary continuous speech task. In contrast, we aim to show that both fully automatic SWU discovery and lexicon induction of a sufficiently high quality are feasible even in the challenging scenario in which the data consists of low-quality recordings of large-vocabulary spontaneous speech and where no graphemic information is used. To this end, we present a new SWU discovery approach based on self-organising HMM-GMM states that are agglomeratively tied across

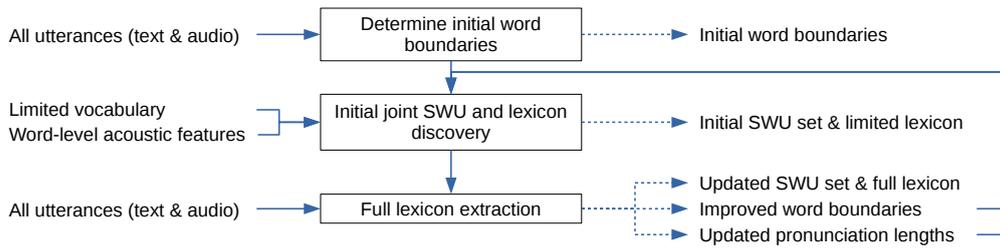


Figure 51: Overview of the proposed SWU discovery and lexicon induction process, as described in Section 5.2.

words. In addition, we present a novel pronunciation lexicon induction method that models variation using a HMM that generates discrete SWU sequences and that iteratively reduces pronunciation variation by pruning the model’s state emission distributions.

## 5.2 Proposed approach to SWU discovery and lexicon induction

In this section, we describe the new approach to discover sets of sub-word units and induce associated pronunciation lexicons using training corpora that are limited to recorded speech and the associated orthographic transcriptions. Since we do not assume the availability of any word boundary information, an initial automatic word-level segmentation is performed. Then, we jointly induce an initial lexicon and associated SWU inventory, using a limited vocabulary and associated isolated word-level acoustic samples that have been extracted from the utterance-level acoustic features using the most recent alignment. Subsequently, a full lexicon induction step is performed, where the initial lexicon is expanded to cover the full dataset by means of a more rigorous scoring and pruning process. The resulting lexicon is then used to determine new pronunciation lengths and more accurate word boundaries, which are in turn used to perform the initial lexicon and SWU inventory discovery. This procedure is illustrated in Figure 51, and is further described in the following sections.

### 5.2.1 Determining initial word boundaries

In previous chapters, attempting to accurately determine word boundaries at the level of SWU sequences has proven to be a key stumbling block to developing high quality lexicons. In this chapter, we will address this by determining the initial word boundaries using the acoustic features and orthography as input. This task usually requires a pronunciation dictionary and a set of acoustic models with which to perform a forced Viterbi alignment. Since we do not initially have a lexicon, and are constraining ourselves to developing a set of approaches that would be usable with non-alphabetic writing systems, we use whole-word acoustic modeling to determine these initial word boundaries. To do this, we model each word as an  $n$ -state left-to-right HMM-GMM with no skips. Each HMM state models the acoustic realisation in terms of 39-dimensional MFCC feature vectors of a particular section of a word. We found that, for the three languages considered in our later experiments,  $n = 8$  states per word gave a good overall compromise between providing sufficient temporal resolution while not over-fitting shorter words. We also found that the most accurate word boundaries were obtained when a single Gaussian distribution was used to model the emission probabilities of each state, and when covariance and transition matrices were tied across all states of all words. The whole-word HMM-GMMs were trained using Baum-Welch re-estimation, and word boundaries were extracted using a forced Viterbi alignment with the trained models.

### 5.2.2 Initial joint SWU and lexicon discovery

In this Chapter we take the view that, in order for the resulting lexicon to be useful for ASR, sub-word units should be based on both linguistic identity as well as on within-class acoustic similarity. We therefore use a similar approach to that proposed by Bacchiani and Ostendorf, where the initial unit inventory and lexicon are jointly discovered [12]. The main assumption used is that a sub-word unit occurring at the same position in each instance of a word represents the same sub-word linguistic element, even if acoustic realizations vary, due to e.g. speaker characteristics. This implies that the segmentation of words needs to be constrained to produce the same number of segments for each instance of a

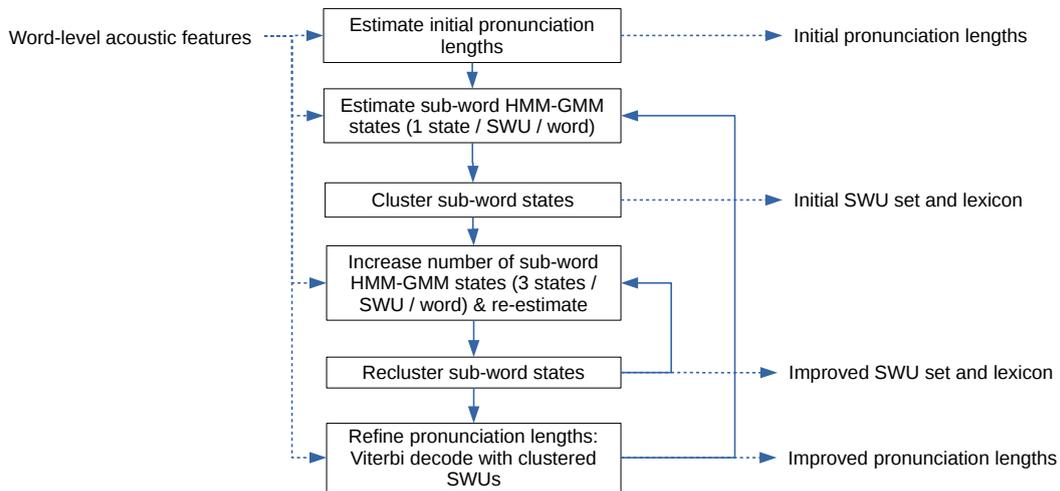


Figure 52: Overview of the initial SWU and lexicon discovery procedure, as described in Section 5.2.2.

word.

Most approaches to unsupervised SWU discovery, including the one taken by Bacchiani and Ostendorf, first attempt a sub-word level acoustic segmentation, before clustering the resulting segments. In contrast, we forgo this explicit initial segmentation. Instead, we view sub-word units as a collection of shared HMM states, which are allowed to self-organise at the level of individual word contexts, i.e. the particular position in a word the unit occurs, before being globally tied into a compact set. We use iterated agglomerative tying to arrive at the final set of units, which differs from the top-down divisive approach taken by Bacchiani and Ostendorf. Furthermore, since we use the lexicon induced during this stage only to seed a more rigorous subsequent lexicon induction process, we can limit its vocabulary to words that occur frequently enough for a robust segmentation and clustering to be achieved. This procedure is shown in Figure 52 and described in detail in the following sections.

### 5.2.2.1 Estimate initial pronunciation lengths

We will consider the desired average SWU duration (in terms of number of feature vectors)  $R_p$  as a tunable meta-parameter. Given the set of word lengths in terms of feature vectors  $\{N_{i,k}\}$ , we select the initial pronunciation length  $L_i$  of word  $w_i$  to be the median of  $\{N_{i,k}/R_p\}$ . Hence  $L_i$  is a first guess of

the number of SWUs in the pronunciation of  $w_i$ . An alternative approach would be to first perform a segmentation of the feature vectors based on signal dynamics (using for example the techniques described in [7–10; 12; 13]) of each word instance and then use the median segmentation length. However, during preliminary testing we found that such a segmentation approach often produced unacceptably poor results when applied to the low quality recordings used in this study. Nevertheless, it is worth investigating in future what the effect might be of more rigorously determining initial pronunciations lengths. In the case where a lexicon has previously been established, which is true when the overall lexicon induction process is being iterated for example, the pronunciation lengths provided by that lexicon may be used instead.

### 5.2.2.2 Estimate sub-word HMM-GMM states

For each word type  $w_i$ , given the initial pronunciation length  $L_i$ , we assign a single mixture HMM-GMM with  $L_i$  states, that models the acoustic likelihood of the word type. All Gaussians are initialized using the global mean and variance, which is known as a *flat start*. The parameters of these states are then trained using Baum-Welch re-estimation, leading to a probabilistic segmentation of the word into  $L_i$  segments that align best when considered over all instances of a word type in the training corpus.

After the re-estimation is complete, the following statistics calculated during Baum-Welch re-estimation are determined for each  $s_{i,j}$ , the  $j$ th state of the HMM representing word type  $w_i$ :

1.  $S_{i,j} = E[\sum\{y_{i,j}\}]$  and  $S_{i,j}^2 = E[\sum\{y_{i,j}^2\}]$ : the expected sum and sum of squares of all the feature vectors  $\{y_{i,j}\}$  that align with the state  $s_{i,j}$ , and
2.  $n_{\text{tr}}(i, j)$ : the expected number of feature vectors aligning with state  $s_{i,j}$ .

The quantities  $S_{i,j}$ ,  $S_{i,j}^2$  and  $n_{\text{tr}}$  are used in the subsequent clustering step.

### 5.2.2.3 Cluster sub-word HMM states

The previous step produces a large number of HMM-GMM states, each modelling the acoustic variation at a particular position within a word across all

instances of that word type in the training corpus. These word-level context-dependent units are subsequently clustered to produce a compact set of sub-word acoustic units. This clustering is achieved by successively tying the pairs of state emission distributions  $a = (i, j)$  and  $b = (i', j')$  whose combination  $ab$  yields the smallest decrease in overall log-likelihood, as given by

$$\Delta LL(a, b) = -\frac{1}{2} \sum_{d=1}^D \left[ n_{\text{fr}}(a) \ln \frac{\sigma_{ab}^2(d)}{\sigma_a^2(d)} + n_{\text{fr}}(b) \ln \frac{\sigma_{ab}^2(d)}{\sigma_b^2(d)} \right], \quad (5.1)$$

where  $d$  indexes the parameter value at the  $d^{\text{th}}$  feature dimension and  $D$  is the dimensionality of the feature vector. This method of state-tying uses the same log-likelihood loss objective that is used during conventional triphone clustering, but the clustering proceeds by bottom-up merging of word-level sub-states rather than by splitting monophone states from the top down using rule-based decision trees [62].

Note that the variance  $\sigma_{ab}^2$ , resulting from the tying of states  $a$  and  $b$ , can be expressed as:

$$\sigma_{ab}^2 = \frac{1}{n_{\text{fr}}(ab)} [S_{ab}^2 - 2\mu_{ab}S_{ab} + n_{\text{fr}}(ab)\mu_{ab}^2], \quad (5.2)$$

where  $S_{ab} = S_a + S_b$  and  $S_{ab}^2 = S_a^2 + S_b^2$ , and  $\mu_{ab} = S_{ab}/n_{\text{fr}}(ab)$  refers to the mean of the feature vectors aligning to the tied state  $ab$ . Hence, once the initial statistics of the individual word-level sub-states have been calculated, the statistics of tied states can be efficiently aggregated. This makes clustering of large datasets tractable. In order to consider only word-level sub-states for which robust statistics can be estimated, we disregard infrequent words in this step. This also allows us to limit the number of initial word sub-states, while still using a large portion of the training dataset (see Table 52). Doing so is advantageous, because the number of comparisons needed to be made at each clustering step grows quadratically with the number of initial states.

#### 5.2.2.4 Increase the number HMM-GMM states and recluster

The previous step leaves us with an initial lexicon and SWU inventory, which we can refine by increasing the temporal resolution of the sub-word states. This is done by expanding the clustered set of single-state HMM-GMMs to the more conventional 3-state models used for ASR. These are first estimated as tied units

across all words, before being cloned for each unique sub-word position. A single forward-backward pass is subsequently used to calculate word position specific statistics, after which clustering proceeds as in Section 5.2.2.3, but with a loss criterium that now includes a summation over the three newly introduced sub-states  $s$  as well as over the  $D$  feature dimensions, as shown in Equation (5.3).

$$\Delta LL(a, b) = -\frac{1}{2} \sum_{s=1}^3 \sum_{d=1}^D \left[ n_{\text{fr}}(a) \ln \frac{\sigma_{ab}^2(s, d)}{\sigma_a^2(s, d)} + n_{\text{fr}}(b) \ln \frac{\sigma_{ab}^2(s, d)}{\sigma_b^2(s, d)} \right] \quad (5.3)$$

### 5.2.2.5 Refine pronunciation lengths

Using these more complex acoustic models, the length in terms of SWUs of the pronunciation of each word in the initial vocabulary can be refined. This is achieved by using a simple unweighted phone-loop to perform Viterbi decoding for each individual word token extracted using the current set of word boundaries (see Section 5.2.3.2 for a bit more detail on how such a Viterbi decode is performed). A revised estimate for each word's pronunciation length is subsequently obtained by calculating the median of the decoded SWU sequence lengths over all instances of that particular word type.

## 5.2.3 Full lexicon extraction

Once an initial SWU inventory has been discovered, a lexicon that best fits the set of corresponding acoustic models and which covers the full training set vocabulary must be derived. The procedure developed to achieve this is indicated in Figure 53. We have adopted the general framework proposed by Singh et al. [26], which proposes a divide-and-conquer strategy of iteratively and successively updating the lexicon, the word segmentations or the SWU acoustic models, while holding the others fixed. However, we diverge from Singh et al. in a number of respects. Firstly, we achieve lexicon initialisation in a fully unsupervised fashion as described in the previous section, and do not rely on graphemes for initial sub-word units. Secondly, instead of using a graphical representation of the observed pronunciations to synthesise new alternative pronunciations, we use these models to bias and constrain subsequent acoustic decoding. This effectively blends SWU sequence likelihoods with acoustic scores. Finally, instead of immediately identifying and retaining only the single

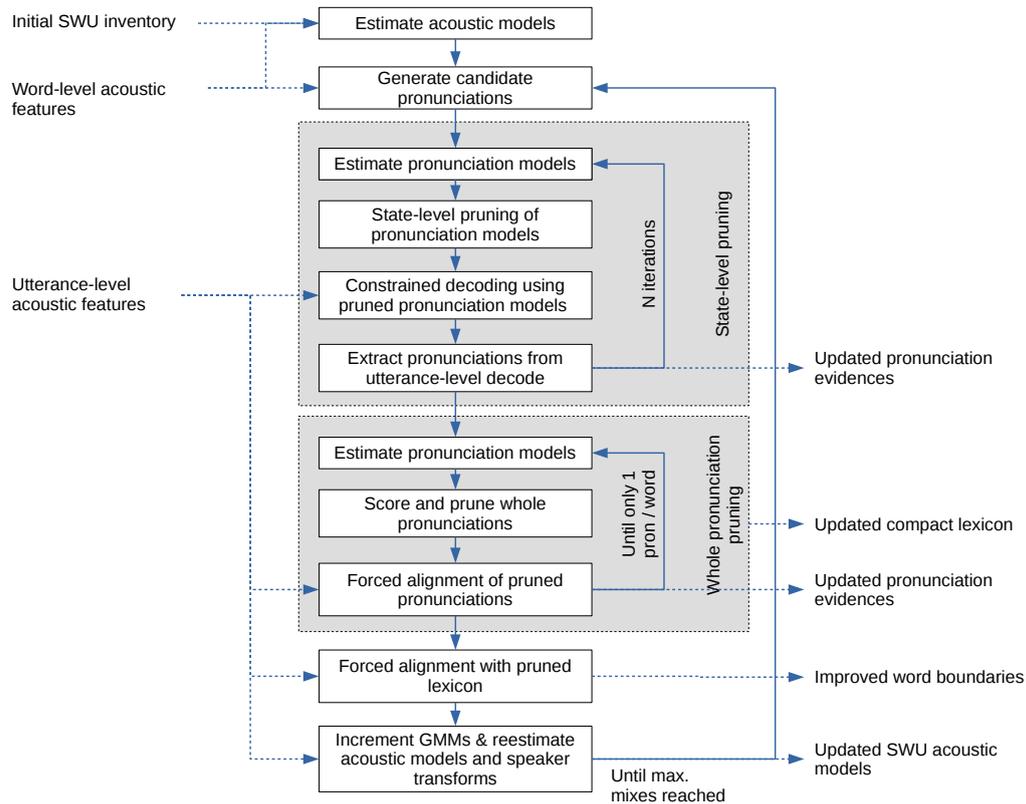


Figure 53: Overview of steps performed during full lexicon extraction, as described in Section 5.2.3.

best pronunciation per word, we allow multiple competing hypotheses that are gradually and iteratively pruned.

Our procedure uses acoustic models estimated using the initial limited lexicon (Section 5.2.2) to produce candidate pronunciations for the full training set vocabulary (as will be described in Section 5.2.3.2). We then perform several pruning passes, using the pronunciation modelling approach described in Section 5.2.3.3 as their basis, to arrive at a single pronunciation per word (described in Sections 5.2.3.4–5.2.3.5). Subsequently, an utterance-level forced alignment is performed with the resulting compact lexicon to produce a new set of word segmentations. The resulting SWU transcriptions are also used to update the SWU acoustic models. As part of acoustic model updates, the number of GMM mixture components is incremented, and the speaker transforms are re-estimated. These steps are repeated until a maximum number of mixture components is reached, as depicted in Figure 53.

### 5.2.3.1 Initialise acoustic models

The initial lexicon obtained using the method described in Section 5.2.2 is used to estimate speaker and context independent single-mixture 3-state HMM-GMMs with CMLLR speaker transforms. A regression class tree is used to tailor the number of transforms generated per speaker to the amount of per-speaker data available. In the case where speaker identity is not available, a single transform per utterance is generated.

### 5.2.3.2 Candidate pronunciation generation

The initial set of candidate pronunciations is generated by performing a forced Viterbi decode of utterances pre-segmented at word-level using an unweighted phone-loop that prohibits immediate repetitions. Decoding pre-segmented samples obviates the need for further processing to extract pronunciations from utterance-level SWU transcriptions. Isolated word decoding also facilitates the efficient generation of multiple pronunciation hypotheses per word instance by means of N-best lists, which is especially helpful in establishing consensus among pronunciations for words that occur less frequently.

Unconstrained or lightly-constrained (using for example a phone-loop that disallows repetitions) acoustic decoding, as described above, searches purely for the most acoustically likely model sequence. This has been observed to often produce sequences containing many spurious insertions. In order to mitigate insertion errors in ASR, it is common to introduce a symbol *insertion penalty* that is added after the acoustic score of each decoded symbol. This insertion penalty can be manipulated to balance the relative rate of insertions and deletions. In this work, before performing acoustic decoding, we automatically tune the insertion penalty on a small held-out set so that the measured average phone duration is equal to the already-specified desired average SWU duration metaparameter  $R_p$ .

### 5.2.3.3 Pronunciation modeling

The purpose of performing pronunciation modelling is to score and prune the many hypothesized pronunciation sequences that are generated for each word type by the procedure described in Section 5.2.3.2. We model each word type  $w_i$

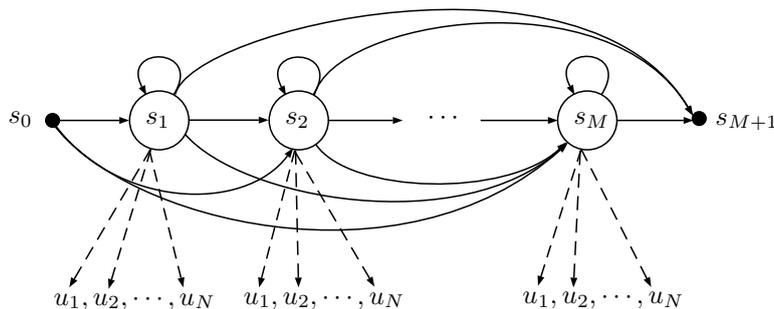


Figure 54: Topology of the word pronunciation HMM.

as an  $M_i$ -state left-to-right HMM with forward skips, with a discrete emission distribution  $p(u_n|s_{ij})$  for each state  $j$  governing the probability of the SWU  $u_n \in [u_1, u_N]$  being associated with that particular position in the word. The benefit of this topology is that it can be embedded as a generative model in a lattice structure to allow for constrained acoustic decoding [63]. We exploit this during the the first pruning stage (described in Section 5.2.3.4), by supplying the acoustic decoder with pruned pronunciation models that bias it towards more likely pronunciation sequences.

The pronunciation models are trained using Baum-Welch reestimation. First, the model topologies for each word  $w_i$  are initialised by setting the number of states  $M_i$  to the median pronunciation length observed for that word during the preceding candidate generation or pronunciation pruning pass. The model parameters are initialised with a uniform emission distribution for each state and a transition matrix set to approximate a predetermined duration distribution with an expected value equal to  $M_i$ . The calculation of the transition matrix is described in more detail below (see Equations 5.6 and 5.7). Then, several passes of forward-backward reestimation are performed on the most recently decoded SWU sequences (resulting from candidate generation or pruning), during which only the state emission distributions are updated:

$$p(u_n|s_{ij}) = \frac{\gamma_{nij}}{\sum_n \gamma_{nij}}, \quad (5.4)$$

where  $\gamma_{nij}$  is the expected number of times that the SWU  $u_n$  is aligned with word type  $w_i$  at state  $j$ , as calculated by the forward-backward algorithm. Note that we use the raw expected occurrence counts to update the state emission distributions. Since the objective of pronunciation modelling in this context is to synthesise pronunciations biased by the predominant evidence, or to score

hypothesised pronunciations, it is counterproductive to smooth the emission distributions by for example introducing pseudocounts for unseen SWUs, since these would be immediately pruned. Note also that the pronunciation models used in this step do not directly take acoustic scores into account. Rather, the acoustic SWU models determine how likely it is that certain SWUs occur at certain locations in each word, and the raw unit occurrence counts act as the “acoustic evidence” used to train the pronunciation models.

The paucity of candidate pronunciations for the majority of words in the datasets (see Table 52) means that it is not feasible to directly estimate the full transition matrix for each word pronunciation model. Rather, we calculate the expected duration of each state

$$\lambda_{ij} = \frac{\sum_n \gamma_{nij}}{N_i}, \quad (5.5)$$

where  $N_i$  is the number of times a pronunciation is observed for word  $w_i$ . We then use the state duration estimates to generate a full transition matrix  $A_i[j, k]$  for word  $w_i$  from state  $j$  to  $k$ :

$$A_i[0, k] = \begin{cases} P(l(s_{ik}) > 0) \prod_{k'=1}^{k-1} P(l(s_{ik'}) = 0) & \text{if } k \in [1, M_i - 1] \\ \prod_{k'=1}^{k-1} P(l(s_{ik'}) = 0) & \text{if } k = M_i \end{cases} \quad (5.6)$$

$$A_i[j, k] = \begin{cases} 1 - \frac{P(l(s_{ij}) > 0)}{\lambda_{ij}} & \text{if } k = j \\ (1 - A_i[j, j]) P(l(s_{ik}) > 0) \prod_{k'=j+1}^{k-1} P(l(s_{ik'}) = 0) & \text{if } k \in [j + 1, M_i] \\ (1 - A_i[j, j]) \prod_{k'=j+1}^{k-1} P(l(s_{ik'}) = 0) & \text{if } k = M_i + 1 \end{cases} \quad (5.7)$$

Equations 5.6 and 5.7 require us to choose the state duration distribution  $P(l(s_{ij}))$ . In this work we assume that the state duration  $l(s_{ij})$  follows a Poisson distribution with parameter  $\lambda_{ij}$ . This was an a preliminary choice, made on the basis that Poisson distributions are often suitable for modeling such durations, and is quite possibly not the best to accurately model state durations. However, HMMs are in any case not able to achieve flexible duration modelling. In order to achieve robust duration modelling, it would be necessary to modify the HMM architecture or to incorporate an external mechanism. In this work, we address

this shortcoming differently depending on whether the pronunciation models are used for decoding or for scoring. When the models are used to generate new pronunciation hypotheses, we bias the lattice transition weights to favour a linear transition through the word states and deemphasize skips and self-transitions. When the models are used for scoring, we model the expected pronunciation length separately and combine the pronunciation model score and the duration score.

The pruning of candidate pronunciations is achieved in two stages, both of which aim to gradually reduce the number of alternative pronunciations per word. The first stage (described next in Section 5.2.3.4) aims to reduce the number of competing pronunciations for each word by reducing the number of SWUs that can be emitted by each word state. The second stage (described in Section 5.2.3.5) discards entire pronunciations until only a small dominant number remain. In the case of this study this is just one pronunciation per word type, although future work should investigate the impact of allowing more variants.

#### 5.2.3.4 State-level pronunciation model pruning and lattice-constrained decoding

The objective of state-level pruning is to gradually reduce the number of alternative pronunciations per word by discarding sub-word units that are uncommon at particular positions in a word. Furthermore, by encoding the pruned pronunciation models as lattices that constrain subsequent acoustic decoding, we can generate acoustically relevant ‘consensus’ candidates that are biased to use the dominant SWU at each sub-word position, even if that variant itself is not observed during the initial candidate generation.

First, pronunciation models as described in Section 5.2.3.3 are trained using the candidates generated using the procedure described in Section 5.2.3.2, or by the previous state-level pruning iteration. These newly estimated emission distributions  $p(u_n|s_{ij})$  for each word sub-state are then pruned. First, the SWU’s associated with each sub-state are sorted from most to least frequently observed. Their occurrence probabilities (as calculated by Equation 5.4) are subsequently accumulated in the variable  $P_{\text{cum}}$  until  $P_{\text{cum}} \geq \delta$ , where  $\delta_s$  is the pre-specified

state-level pruning threshold. The last probability that was added to  $P_{\text{cum}}$  before  $\delta_s$  was reached is recorded as  $P_{\text{thres}}$ . A pruned emission distribution is then calculated according to Equation 5.8.

$$p_{\text{pruned}}(u_n|s_{ij}) = \begin{cases} \frac{p(u_n|s_{ij})}{P_{\text{cum}}} & \text{if } p(u_n|s_{ij}) \geq P_{\text{thres}} \\ 0 & \text{otherwise} \end{cases} \quad (5.8)$$

After the pruned state emission distributions have been calculated, it is necessary to generate new pronunciation candidates that are constrained by the pruned models. For each utterance, a decoding lattice is constructed by chaining the lattice structure provided by the pronunciation models for each word in the utterance, incorporating optional silence models between words. These decoding lattices, along with the current set of acoustic models, are presented to an acoustic decoder to produce new utterance-level SWU transcriptions. From these we extract an updated set of candidate pronunciations by performing a Viterbi alignment with the same set of pronunciation models used for decoding.

This procedure of estimating pronunciation models, pruning them and then using the pruned models to generate new pronunciations is repeated several times until the number of competing pronunciation variants is reduced sufficiently to allow whole pronunciations to be pruned.

### 5.2.3.5 Whole pronunciation pruning

Speech recognizers have been observed to perform best when presented with fewer lexicon entries per word. However, after the state-level pruning discussed in the previous section, it is not uncommon for a handful of pronunciations to remain for some words. During whole-pronunciation pruning, the objective is to further reduce the number of competing pronunciation variants down to an acceptable number for ASR. This is achieved by iteratively discarding a fraction of the poorest scoring variants until a suitably compact lexicon remains. The remainder of this section describes this scoring and pruning procedure in more detail.

The pronunciations that remain after each such pruning iteration are re-aligned with the acoustic features in order to update the counts for each candidate pronunciation. These pruned pronunciations and associated counts are subsequently used to estimate new pronunciation models (described in

Section 5.2.3.3) for each word  $w_i$ . Using these pronunciation models, a score is calculated for each pronunciation  $D_{ik}$ :

$$S(D_{ik}) = \frac{1}{Z_i} \exp \left( \frac{\log P(D_{ik}|\theta_i)}{t_{ik}} + \alpha \log P(t_{ik}) \right), \quad (5.9)$$

where  $P(D_{ik}|\theta_i)$  represents the probability of the word pronunciation HMM generating the SWU sequence associated with that particular pronunciation, as calculated by the forward algorithm. Further,  $t_{ik}$  is the length of the pronunciation in terms of SWUs and  $P(t_{ik})$  is the probability of this pronunciation length given the expected distribution of lengths for that word, which we again assume to follow a Poisson distribution. The normalising factor  $Z_i$  ensures that  $S(D_{ik})$  is a properly normalised probability distribution.  $S(D_{ik})$  can be seen to be the exponent of a weighted sum of the length-normalised HMM sequence log-probability and the pronunciation length log-probability. The parameter  $\alpha$  controls the weight of the second term relative to the first, and is chosen to be relatively small ( $\leq 0.5$ ).

We retain variants  $D_{ik}$  with highest associated  $S(D_{ik})$  and accumulate their scores until this reaches a predetermined threshold  $\delta_w$ . If any words still have an unacceptable number of variants, then the pruning process is repeated. Following this, an utterance-level forced alignment of the newly pruned lexicon is performed, which produces updated alignment counts. These are used to re-estimate the pronunciation models and length distributions used for scoring and pruning.

### 5.2.3.6 Update word boundaries and acoustic models

Once the pruning steps are complete, we use the resulting compact lexicon to perform a forced alignment with the training set utterances. This yields an improved set of word boundaries. The forced alignment also produces a new SWU transcription for each utterance which is used to update the SWU acoustic models. The acoustic model update involves incrementing the number of mixture components for the GMM associated with each state of the SWU acoustic HMM and subsequently re-estimating the model parameters. The CMLLR transforms associated with each speaker in the dataset are also updated.

Table 51: Summary of datasets used for experimental evaluation. #Words indicates the number of word tokens.

Dataset	Train			Test		
	Hours	#Utts	#Words	Hours	#Utts	#Words
Acholi	9.19h	4863	8719	0.30h	156	882
Ugandan English	5.75h	4403	6737	0.39h	164	988
Luganda	9.59h	8774	18305	0.29h	164	1150
Librispeech	5.31h	1519	7230	2.03h	1089	4391

## 5.3 Experiments and results

### 5.3.1 Datasets

The datasets used in the experiments performed in this chapter are summarised in Table 51. They have been compiled from recordings of Ugandan community radio stations broadcasting in Ugandan English, Luganda and Acholi, and have been orthographically annotated by mother-tongue speakers [64; 65]. Luganda and Acholi are both severely under-resourced Ugandan languages, with practically no resources available other than the datasets used in this study. This is despite these languages having a combined total of 8 million native speakers. The recordings are taken from radio talk shows and consist of fast, conversational speech, with poor overall audio quality due to noise in the received signal and a high incidence of background sounds. The datasets also included pronunciation lexicons compiled by phonetic experts, which enabled the establishment of baselines indicating what can be achieved using the conventional ASR system development approach.

In addition to the under-resourced datasets discussed above, we also performed some system evaluation using the standard “mini” subset of the Librispeech ASR corpus [66]. This corpus consists of read speech in English, which is a well resourced language. However, in contrast to the Ugandan datasets, it is freely available, which would enable other researchers to compare their results with ours. Further, the use of a small and standardised subset of the Librispeech corpus effectively simulates an under-resourced task.

Table 52 summarises the out-of-vocabulary (OOV) rates of the datasets, with

Table 52: Summary of the vocabulary distribution and OOV rates for the three datasets introduced in Table 61. Tokens in the training set occurring more than three and nine times are indicated by  $n > 3$  and  $n > 9$  respectively.

Dataset	#Words	Train				#Words	Test			
		$n > 3$		$n > 9$			OOV ( $n = 0$ )		OOV ( $n < 4$ )	
		Types	Tokens	Types	Tokens		Types	Tokens	Types	Tokens
Acholi	8719	27.9%	91.9%	13.3%	85.3%	882	10.8%	3.5%	24.8%	8.2%
Ugandan English	6737	26.9%	88.2%	11.1%	78.3%	988	19.4%	7.3%	40.0%	15.6%
Luganda	18305	14.4%	75.0%	5.6%	63.8%	1150	31.3%	16.2%	49.7%	26.6%
Librispeech	7230	22.9%	85.0%	8.5%	73.9%	4391	43.5%	12.3%	72.8%	22.7%

the entries under *Types* indicating the fraction of unique words, and *Tokens* the fraction of all word occurrences. The table shows that, in comparison with Acholi and Ugandan English, Luganda is characterised by a large proportion of rarely occurring words and associated higher OOV rate. This can be ascribed to its agglutinative character. Librispeech also has a comparatively high OOV rate, but this can be ascribed more to the relatively large size of its test set. Preliminary experimentation revealed that a training set word occurrence frequency of approximately 4 is the minimum with which a useful pronunciation can be extracted, and an occurrence frequency of at least 10 is required for a relatively robust lexicon entry to be deduced. This occurrence count threshold of 10 is the cut-off that was used in the following experiments during the initial SWU discovery described in Section 5.2.2. Table 52 shows that, although this threshold means that we only use between 5.6% and 13.5% of the vocabulary types, this covers between 64% and 85% of the corresponding training data tokens.

For the test sets, we list the fraction of the test set vocabulary that is not seen during training ( $n = 0$ ), or is seen so infrequently that the extracted pronunciation is likely to be poor ( $n < 4$ ). It is clear from the table that a major challenge in performing large vocabulary unsupervised lexicon extraction lies in determining good pronunciations for the majority of word types in each dataset that are used relatively infrequently. A higher proportion of infrequent words also significantly increases the difficulty of automatically inferring word boundaries. It can be seen from the table that the challenge of developing a lexicon is especially severe for Luganda, which, due to its highly-agglutinating nature, has a large vocabulary with many words that are seen only once.

### 5.3.2 Experimental setup

The speech files in each dataset described in Table 61 are parameterised as 39-dimensional feature vectors, consisting of twelve MFCCs, with the addition of log energy, and first and second order differential coefficients. A first set of word-boundaries is then obtained using the approach described in Section 5.2.1. In order to refine these word boundaries, a complete lexicon induction pass is performed for each of the four datasets, where each such pass comprises all the steps described in Sections 5.2.2–5.2.3 (Figures 52 and 53). This leaves us with a more accurate set of word boundaries resulting from a forced alignment with this first set of lexicons.

We then investigate the effect of the two most important meta-parameters, which are the number of units in the SWU inventory  $N_p$  and the average SWU duration  $R_p$ . The other important meta-parameters, the state level pruning threshold  $\delta_s$  and the word-level pruning threshold  $\delta_w$ , were set to values that represented a reasonable initial trade-off between gradual pruning and limiting the number of pruning iterations. Although there is the potential for future system improvement by also optimising these parameters, time and resource constraints did not permit doing so for this study. In the case of Librispeech, we used the optimal metaparameters determined for Ugandan English.

Each lexicon induction pass is repeated a number of times, with each iteration yielding an updated set of word boundaries, as well as an updated lexicon which can be used to seed the initial SWU clustering (as described in Section 5.2.2.4) in the subsequent iteration.

For the acoustic model estimation, decoding and alignment necessary during lexicon extraction, we used the HTK tools [1]. However, for the evaluation of ASR performance, we used the Kaldi toolkit, culminating in a system consisting of a combination of SGMM-MMI and DNN/HMM acoustic models (SGMM+DNN) with cross-word dependency [67]. We also evaluated the ASR performance of the best-performing lexicons using a TDNN/HMM and SGMM-MMI system combination (SGMM+TDNN). Since we generally found that the SGMM+TDNN system reflected the same trends shown by the SGMM+DNN system, so we utilised the latter system for meta-parameter optimisation since it was quicker to train. In addition to the acoustic models, we trained bigram language models

Table 53: Summary of ASR word error rates for baseline systems as well as systems using automatically induced sub-word units and associated lexicons.

Dataset	System	Grapheme	Expert	Automatic	Automatic (refined)
Acholi	SGMM+DNN	44.90%	46.50%	46.60%	46.47%
	SGMM+TDNN	42.20%	43.68%	43.77%	42.77%
Ugandan English	SGMM+DNN	50.02%	47.98%	55.03%	51.30%
	SGMM+TDNN	46.51%	45.00%	53.74%	49.86%
Luganda	SGMM+DNN	55.14%	54.95%	67.64%	61.63%
	SGMM+TDNN	52.55%	53.22%	64.99%	60.26%
Librispeech	SGMM+DNN	38.01%	36.15%	48.91%	44.87%
	SGMM+TDNN	36.57%	34.40%	47.94%	43.88%

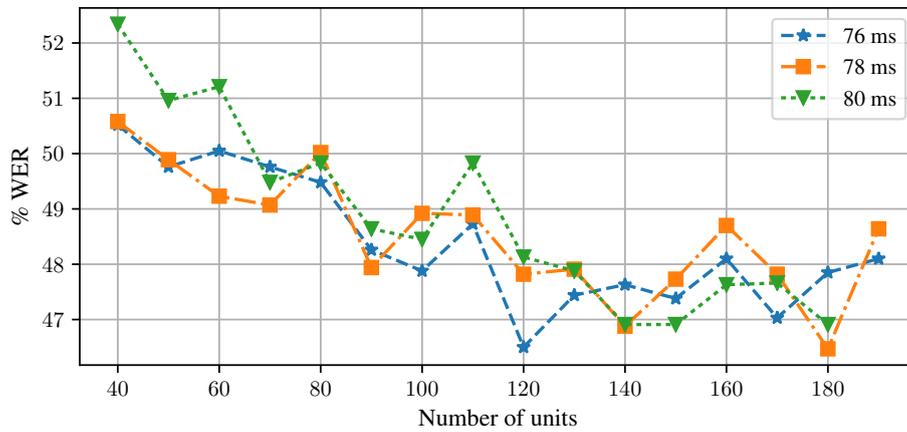
from corpora consisting of only the corresponding training prompts. These are relatively poor language models, due to the small corpus and low n-gram order, and also because they lack coverage of the test set OOV words. However, this also represents a realistic benchmark of what is achievable in a truly under-resourced setting.

All experiments were performed on 2.10GHz AMD Opteron 6172 processors. In the case of the larger datasets (Acholi and Luganda), a full lexicon extraction pass took around 165 hours (walltime) when executed on 8 CPU cores, resulting in a total consumption of 994 CPU-hours. The average peak memory usage per experiment was approximately 6GB of RAM.

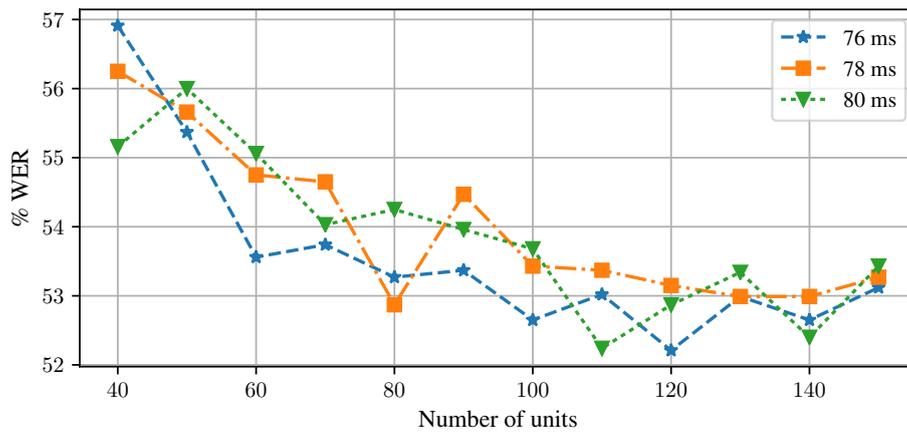
### 5.3.3 Automatic speech recognition

We include two baseline systems against which the performance of systems using our automatically-induced SWUs and lexicons were compared. The first baseline uses graphemes as SWUs. This is a relevant baseline in an under-resourced setting, since, in the absence of an expert lexicon, using context-dependent graphemes for ASR is often a viable strategy for languages that are written with an alphabetic orthography. The second baseline is the recognition performance achieved using the handcrafted (expert) lexicons provided with each dataset.

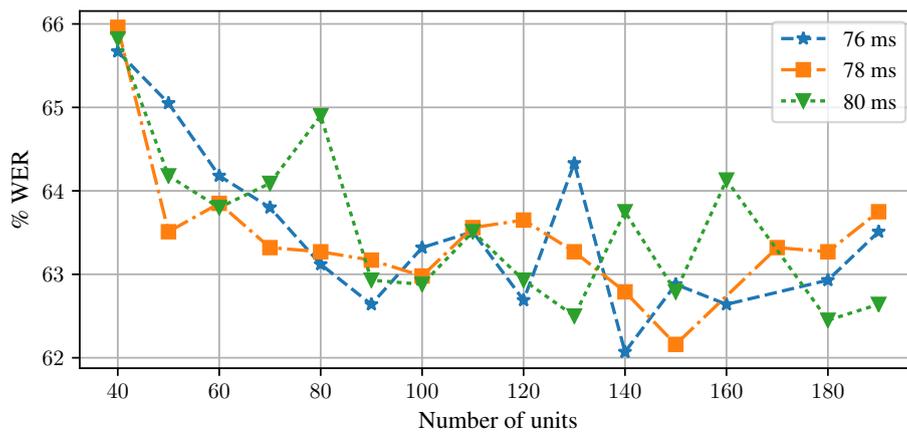
The baseline ASR performance for each of the four considered languages, as well as the corresponding performance for the best automatically discovered lexicon is shown in Table 53. We evaluate the performance of two variants of the



(a) Acholi



(b) Ugandan English



(c) Luganda

Figure 55: The impact of the meta-parameters number of units ( $N_p$ ) and average unit length ( $R_p$ ) on the word error rates of the resulting ASR systems.

same automatically derived lexicons. The first of these contains an automatically derived pronunciation for every word type found in the training set. This allows us to evaluate the end-to-end ASR performance of a lexicon derived in a fully unsupervised manner, using only recorded speech and symbolic word sequences.

In the second, we discard from the automatically discovered lexicon all pronunciations for words that occur fewer than 4 times in the training set, and replace these using a grapheme-to-SWU system [68] trained on the remaining lexicon. This demonstrates one way in which graphemic information can optionally be incorporated to improve system performance, should it be available, without necessarily relying on it during lexicon induction. The resulting refined lexicon somewhat ameliorates the performance penalty incurred by poorly induced pronunciations for infrequent words. It therefore allows us to better evaluate the intrinsic suitability of the automatically discovered SWUs in terms of ASR performance. Note that also in this case the resulting set of SWUs and associated lexicon has been obtained in a fully unsupervised manner.

It can be seen from the table that, for Acholi, we have been able to match the performance of a human expert, although a grapheme based system still achieved slightly better performance. The small performance margin between the automatic and refined automatic lexicons is ascribed to the relatively small proportion of test-set words affected by the refining step.

From the vocabulary analysis in Table 52, it was clear that Luganda would pose a challenge to robust automatic SWU discovery and lexicon induction, despite having the largest training set. This is reflected in the relatively poor performance of the automatically-induced lexicon relative to the baseline systems, and also in the larger margin between the automatic lexicon and the refined lexicon. In the next chapter, we will adapt the approaches presented in this work to be more amenable to application to highly-agglutinating languages like Luganda, by performing a morphemic chunking pre-processing step.

The results achieved for Ugandan English and Librispeech support the observed relationship between OOV rates and the margin between the automatic and refined lexicons. In this regard, Ugandan English presents a middle ground between the highly agglutinating Luganda and the more disjunctive Acholi, while Librispeech demonstrates the effect of a large vocabulary mismatch between

Table 54: Summary of ASR word error rates achieved when using ground truth word boundaries.

Dataset	System	Grapheme	Expert	Automatic	Automatic (refined)
Acholi	SGMM+DNN	44.90%	46.50%	46.75% (+0.15%)	45.28% (-1.19%)
	SGMM+TDNN	42.20%	43.68%	44.34% (+0.57%)	43.58% (+0.81%)
Ugandan English	SGMM+DNN	50.02%	47.98%	52.74% (-2.29%)	51.18% (-0.12%)
	SGMM+TDNN	46.51%	45.00%	51.71% (-2.03%)	48.42% (-1.44%)
Luganda	SGMM+DNN	55.14%	54.95%	62.98% (-4.66%)	60.91% (-0.72%)
	SGMM+TDNN	52.55%	53.22%	61.39% (-3.60%)	58.42% (-1.84%)

training and evaluation corpora. The performance of the Ugandan English system using the refined automatically induced lexicon compares reasonably well with the grapheme baseline, despite the smaller dataset.

### 5.3.4 Impact of word boundaries

In order to determine the performance penalty resulting from errors in the automatically discovered word boundaries, we repeated the experiments in Table 53 using more accurate word boundaries. These were obtained by means of a Viterbi alignment of the speech features with the expert lexicons, and are considered accurate enough to represent *ground truth*. In order to keep the results as directly comparable as possible, we used the final SWU acoustic models produced by the corresponding experiment in Table 53 to perform a single additional full lexicon extraction pass (described in Section 5.2.3) using the ground truth word boundaries. Note that the initial joint SWU discovery and lexicon induction step (Section 5.2.2) could also potentially have benefited from more accurate word boundaries, but that this is not explored here.

The ASR performance of the lexicons extracted in this way are shown in Table 54, with the figures in parentheses indicating the absolute performance change relative to Table 53 (with a negative difference indicating an improvement). It can be seen that a considerable proportion of the error margin between the automatic and refined automatic lexicons can be attributed to inaccurate word boundaries. However, even when using ground truth word boundaries, the refined lexicons still exhibited better performance than their unrefined counterparts. This indicates that it remains challenging to induce robust pronunciations for infrequent words due to the disproportionate effect that a

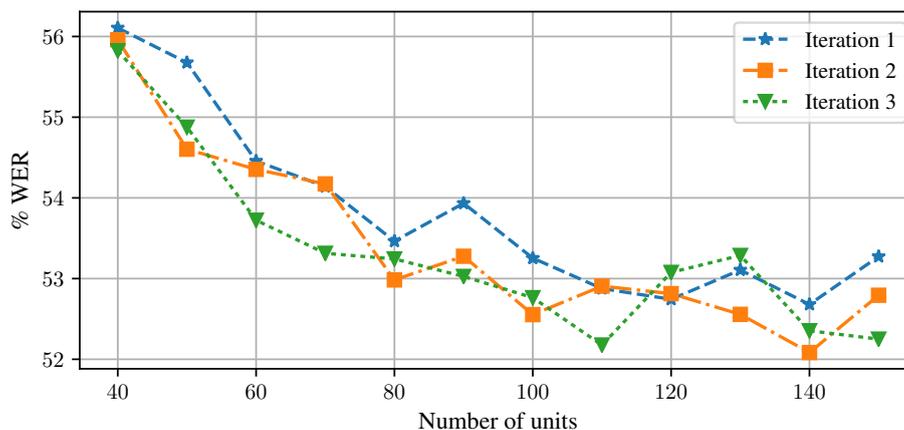


Figure 56: Word error rates for three consecutive lexicon extraction passes (Ugandan English).

few poor pronunciation realisations (due to e.g. unusual speaker characteristics or poor audio) can have on the induced pronunciations. The refined automatic lexicons induced using ground truth word boundaries also show modest performance gains relative to the refined lexicons induced using automatically discovered word boundaries. This indicates that also the pronunciations of more frequent words unaffected by the refinement step benefited from more accurate word boundaries.

### 5.3.5 Meta-parameter analysis

The impact of the meta-parameters of the SWU discovery and lexicon induction process on the resulting ASR performance is shown in Figure 55. In general, it appears that the optimum number of sub-word units tends to be considerably higher than the number of expert-determined phonemes. Furthermore, when comparing Ugandan English with its 5.75 hours of training data and support for 120 units with Acholi with its 9.19 hours of training data and 180 units, there appears to be a link between the amount of training data available and the optimal number of discovered units. This is likely due to the units we discover being inherently context dependent. Thus more training data allows more unique contexts to be supported.

The experiments shown in Figure 55 do not support any strong conclusions about the impact of the desired average SWU duration metaparameter  $R_p$ . It

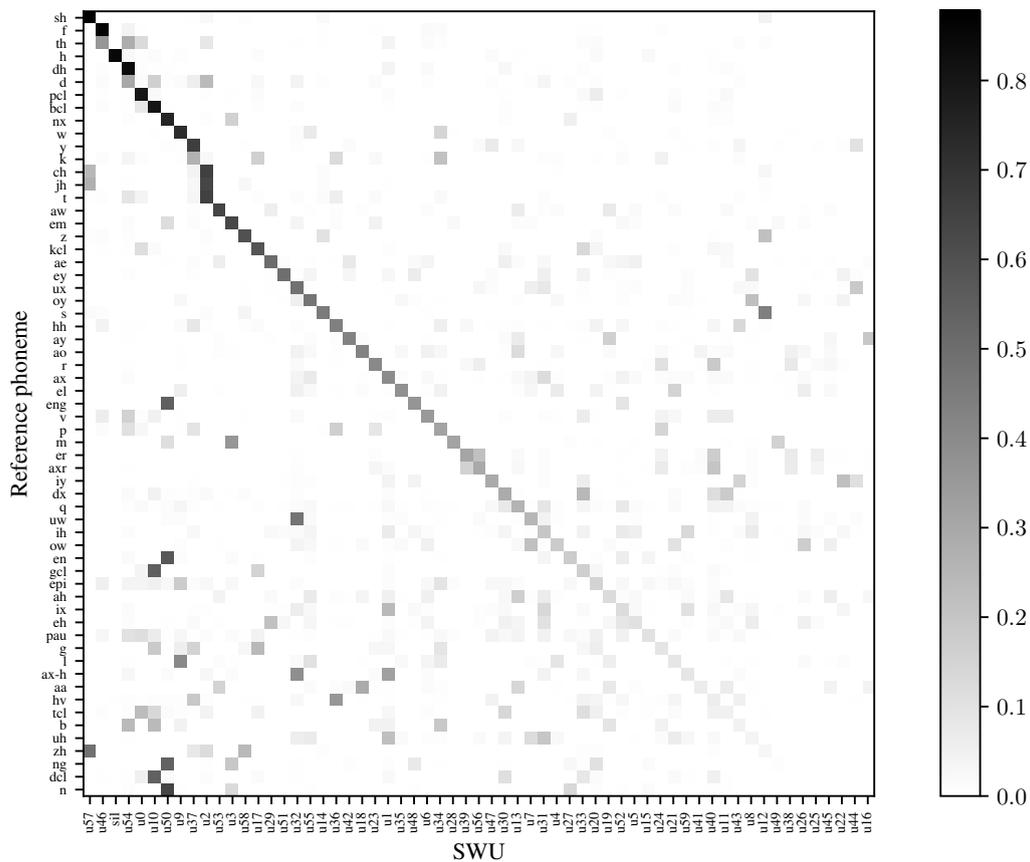


Figure 57: Coincidence of automatically discovered units with reference phonemes (TIMIT).

would appear that the shortest units perform slightly better on average, but a larger range of parameter values needs to be investigated to clarify whether that trend continues further. It is also not yet clear whether there exists a dependency between the number of units and the corresponding optimal average unit duration.

Figure 56 shows how consecutive lexicon extraction passes improve word error rates. Each lexicon extraction pass comprised the steps described in Sections 5.2.2–5.2.3 (Figures 52 and 53). In this study, we stopped after three iterations due to the time and computational cost involved. It may be possible to achieve some further gains by allowing more iterations.

### 5.3.6 Nature of discovered units and lexicon

In order to investigate the nature of the discovered SWUs, and how they relate to conventional phonetic units, we applied our procedure to the TIMIT dataset for which manually produced phonetic transcriptions are available [69]. Figure 57 shows a 2D coincidence histogram between our automatically discovered SWU inventory and the TIMIT reference phonemes, computed by counting the number of times at least 50% of the span of an automatically discovered SWU occurs within the boundaries of a phoneme in the reference TIMIT transcriptions. Each row is normalised to show the fraction of occurrences of a phoneme represented by a particular SWU.

From the figure, it is clear that there is a strong correspondence between reference phonemes and the discovered units, suggesting that the automatically discovered SWUs are highly phonetically relevant. The mapping is not quite one-to-one, with some units corresponding to multiple phonemes and some phonemes being represented by multiple units. Furthermore, as the number of units increases, we expect also the number of units corresponding to a single phoneme to increase. This is not a concern if it better allows the ASR system distinguish between different words, which is indicated to be the case in Figure 55.

In general, we observed that the discovered units are highly context dependent, with a phoneme or class of phonemes being represented by several different units depending on which phoneme or class of phonemes precede or follow it.

For example, Table 55 shows how in Ugandan English the TIMIT phone *k* is represented by the automatically discovered units **u77**, **u126** and **u14** depending on whether it is followed by the TIMIT phone groups (*ao, aa, uh, em*), (*ae, ah, aw*) or (*eh, ey, iy, ih*) respectively. Table 56, on the other hand, shows how the TIMIT phones *b*, *d* and *g* are represented by sometimes overlapping automatically-discovered units **u94**, **u53**, **u105** and **u75** depending on the right-hand context. This observation is not surprising since we cluster based on acoustic similarity, and context influences acoustic realization of phonemes to such an extent that the same phoneme in different contexts can be more acoustically distinct than different but similar phonemes in the same context.

Table 55: Example of the representation of the TIMIT phone k exhibiting right-hand context-dependence in the automatically induced lexicon for Ugandan English ( $N_p = 140$ ,  $R_p = 7.8$ ). The first two columns indicate symbols used in the TIMIT phonetic alphabet, while the last column indicates automatically-discovered SWU labels.

Phoneme	Context	Word	Pronunciation
k	ao, aa, uh, em	CALL	<b>u77</b> u110 u72 u18
		CALLED	<b>u77</b> u110 u72 u69
		COMMISSION	<b>u77</b> u110 u87 u82 u109 u113
		COMMITTEE	<b>u77</b> u110 u87 u29 u33 u34
		COMMUNITY	<b>u77</b> u110 u87 u139 u23 u34 u23
		CONFERENCE	<b>u77</b> u110 u7 u57 u17 u125 u111 u2
		CONSCIENCE	<b>u77</b> u64 u1 u82 u0 u96
		CONTEST	<b>u77</b> u64 u28 u33 u21 u102 u1 u2
		COST	<b>u77</b> u64 u1 u2 u33
k	ae, ah, aw	CAMPAIGN	<b>u126</b> u114 u130 u67 u8 u131 u35
		CAN	<b>u126</b> u12 u28
		CANDIDATE	<b>u126</b> u114 u118 u90 u47 u51 u20
		CANDIDATES	<b>u126</b> u114 u118 u90 u63 u47 u23 u78
		CAN'T	<b>u126</b> u114 u130 u95
		COME	<b>u126</b> u114 u55 u83
		COMES	<b>u126</b> u114 u55 u83 u6
		COMING	<b>u126</b> u114 u5 u98 u35
		COUNCIL	<b>u126</b> u114 u55 u1 u9 u10
		COUNTRY	<b>u126</b> u114 u118 u24 u14 u93
		COUNTY	<b>u126</b> u114 u84 u79 u24 u34 u74
		KAMPALA	<b>u126</b> u114 u55 u67 u8 u114 u4 u102
k	eh, ey, iy, ih	CARE	<b>u14</b> u115 u44
		CASE	<b>u14</b> u22 u100 u59
		CASES	<b>u14</b> u97 u100 u43 u0 u96
		CAME	<b>u14</b> u13 u67
		K.	<b>u14</b> u22
		KEEP	<b>u14</b> u93 u20
		KILLED	<b>u14</b> u97 u60 u69
		KINGDOM	<b>u14</b> u13 u35 u53 u70 u83
		KIZZA	<b>u14</b> u122 u6 u76 u5

Table 56: Example of multiple TIMIT phones (b, d, g) represented with an overlapping pool of units depending on context in the automatically induced lexicon for Ugandan English ( $N_p = 140$ ,  $R_p = 7.8$ ). The first two columns indicate symbols used in the TIMIT phonetic alphabet, while the last column indicates automatically-discovered SWU labels.

Phoneme	Context	Word	Pronunciation
b, d	iy, ih, ax	BEAT	<b>u94</b> u37 u88 u20
		BECAUSE	<b>u94</b> u29 u77 u110 u79 u6
		BECOME	<b>u94</b> u29 u126 u114 u55 u83
		BEEN	<b>u94</b> u13 u35
		BEFORE	<b>u94</b> u93 u57 u56 u92
		BEHIND	<b>u94</b> u139 u107 u85 u50
		BETWEEN	<b>u94</b> u29 u33 u38 u13 u35
		BIG	<b>u94</b> u37 u88
		BILL	<b>u94</b> u37 u60 u15
		DECISION	<b>u94</b> u100 u43 u34 u23 u38 u28
b, d, g	eh, ao, aa, oy, uw, ah, er, ay	BEST	<b>u53</b> u49 u111 u59
		BOARD	<b>u53</b> u56 u64 u69
		BODY	<b>u53</b> u64 u44 u47
		BOTH	<b>u53</b> u64 u11 u20
		BOYS	<b>u53</b> u64 u138 u100 u96
		DEAD	<b>u53</b> u129 u23 u53
		DELEGATES	<b>u53</b> u115 u106 u47 u97 u111 u78
		DO	<b>u53</b> u95
		DOLLARS	<b>u53</b> u70 u79 u5 u6 u96
		DONE	<b>u53</b> u52 u118 u28
		DON'T	<b>u53</b> u70 u69
		GETTING	<b>u53</b> u97 u29 u33 u34 u35
		GIRLS	<b>u53</b> u52 u135 u79 u6
		GIVES	<b>u53</b> u128 u6 u96
		GUYS	<b>u53</b> u52 u49 u6
		GOVERNMENT	<b>u53</b> u52 u55 u54 u25 u24
b	ae, ay, r	BACK	<b>u105</b> u123 u4 u30
		BAD	<b>u105</b> u123 u81 u44
		BRIDE	<b>u105</b> u17 u125 u23 u73
		BY	<b>u105</b> u123 u85 u23
g	aa, ow, uh, r	GOD	<b>u75</b> u64 u44 u132
		GOING	<b>u75</b> u138 u35
		GONNA	<b>u75</b> u49 u5
		GOODS	<b>u75</b> u71 u69 u78
		GROUPS	<b>u75</b> u18 u88 u33 u59

Table 57: Sample of frequent n-grams found in the automatically induced lexicon ( $N_p = 50$ ,  $R_p = 7.8$ ) for Ugandan English.

N-Gram	Word	Pronunciation
u40 u13 u32 u38 u34 u25	UGANDA UGANDAN UGANDANS	<b>u40 u13 u32 u38 u34 u25</b> u38 <b>u40 u13 u32 u38 u34 u25</b> u34 u25 <b>u40 u13 u32 u38 u34 u25</b> u34 u25 u3
u4 u5 u2 u14 u22 u25	HAPPEN HAPPENS	<b>u4 u5 u2 u14 u22 u25</b> <b>u4 u5 u2 u14 u22 u25</b> u12
u43 u47 u22 u26 u16	MARRIAGE MARRIAGES	<b>u43 u47 u22 u26 u16</b> <b>u43 u47 u22 u26 u16</b> u45 u12
u7 u8 u28 u6	PUT COURT REPORT REPORTS SUPPORT SUPPORTERS	<b>u7 u8 u28 u6</b> <b>u7 u8 u28 u6</b> u22 <b>u7 u8 u28 u6</b> u22 u2 <b>u7 u8 u28 u6</b> u3 u9 u11 u2 <b>u7 u8 u28 u6</b> u9 u10 <b>u7 u8 u28 u6</b> u11 u12 u3
u7 u4 u34 u25	CAN AFRICAN CANDIDATE	<b>u7 u4 u34 u25</b> u5 u2 u39 u17 <b>u7 u4 u34 u25</b> <b>u7 u4 u34 u25</b> u21 u37 u20 u21 u1
u32 u38 u34 u25	DONE UGANDA	<b>u32 u38 u34 u25</b> u40 u13 <b>u32 u38 u34 u25</b> u38
u31 u17 u25	ACTION CREATION POSITION REGULATIONS	u4 u5 u6 <b>u31 u17 u25</b> u7 u17 u22 u20 <b>u31 u17 u25</b> u7 u8 u28 u12 u42 <b>u31 u17 u25</b> u22 u21 u42 u19 u20 <b>u31 u17 u25</b> u3
u16 u42 u26	T. G. AUTHORITY COUNTY SECURITY	<b>u16 u42 u26</b> <b>u16 u42 u26</b> u2 u39 u10 u28 u22 u26 <b>u16 u42 u26</b> u7 u4 u46 u25 <b>u16 u42 u26</b> u9 u36 u14 u17 u22 u26 <b>u16 u42 u26</b>

Finally, Table 57 lists some frequent SWU n-grams encountered in our automatically discovered lexicon for Ugandan English. These examples exhibit a reassuring degree of regularity between different forms of the same base word (e.g. Uganda/Ugandan/Ugandans) and between different base words with the same suffix (*action/creation/position*). The table also demonstrates some of the pitfalls of an acoustic SWU inventory developed with acoustic similarity as the primary optimisation objective. Several acoustically and linguistically distinct but somewhat similar sounding linguistic units (*put/court/report*) are assigned the same pronunciation sequences. In such cases, we would have to rely on a sophisticated language model to distinguish between words.

## 5.4 Summary and conclusions

This chapter has presented a novel sub-word unit discovery and lexicon induction approach that requires as input only recorded speech utterances and their associated orthographic transcriptions. This includes a new method of SWU discovery based on self-organising HMM-GMM states that are agglomeratively tied across words. In addition, we have presented a novel pronunciation lexicon induction method that models variation using an HMM that generates discrete SWU sequences and that iteratively reduces pronunciation variation by first pruning the state emission distributions and eventually by removing entire pronunciations. We used this method to implement DNN-HMM and TDNN-HMM ASR systems for three under-resourced African languages and we were able to produce lexicons that performed very well compared to baseline expert systems. The performance was observed to vary depending on the characteristics of the target language. For example, the lexicons developed for the Luganda dataset performed comparatively poorly, because that language's agglutinating nature leads to a large proportion of long, infrequent words. This will be addressed in the next chapter by incorporation of morphemic chunking. In the case of Acholi, however, the automatically induced lexicon was able to match the performance achieved using an expert-compiled lexicon, even where accurate word boundaries were not available and when no graphemic information was used. This demonstrates the potential of the proposed method to realise state-of-the-art ASR performance

using a pronunciation lexicon induced in an entirely unsupervised fashion.

## Chapter 6

# Lexicon induction for highly-agglutinating languages

### 6.1 Introduction

The previous chapter highlighted the challenges that highly agglutinating languages such as Luganda pose to automatic lexicon induction. Specifically, the agglutinating nature of these languages lead to a massive expansion of vocabulary types. It is usually very difficult to obtain sufficiently many observations of such word types to induce robust pronunciations. This contributes to an observed performance margin relative to expert lexicons which is strongest for agglutinating languages. In this chapter, we attempt to address this by inducing pronunciations for orthographic types that are shorter than words. Ideally, these would be morphemes designated by expert linguists. However, we desire to have fully automatic procedures that require minimal human input in order to operate in the under-resourced environment. We therefore propose and evaluate a data-driven technique that subdivides words into morpheme-like chunks (hereafter referred to as morphological segments) for which more reliable pronunciations can potentially be induced.

## 6.2 Background

### 6.2.1 Morphological segmentation

The process of morphological segmentation usually consists of two stages: *training* a segmentation model  $\theta$ , and subsequent *decoding* of a test set  $W$  using a tokenization function  $\phi(W; \theta)$  [70–72]. In our case, however, we tokenize the training corpus itself, so there is no secondary test set.

The training of the segmentation model  $\theta$  involves minimizing a cost function  $L(\mathbf{D}_W, \theta)$ , where  $\mathbf{D}_W$  is the corpus of training set words. The cost function consists of a model likelihood  $p(\mathbf{D}_W | \theta)$  and a model prior  $p(\theta)$ . The data likelihood assumes that the set of  $L_j$  morphological segments  $m_{ji}$  constituting a word  $w_j$  occur independently such that:

$$\log p(\mathbf{D}_W | \theta) = \sum_{j=1}^N \sum_{i=1}^{L_j} \log p(m_{ji} | \theta). \quad (6.1)$$

The maximum-likelihood estimate of the probability of a morphological segment is based on its usage count  $\tau_i$ :

$$p(m_i | \theta) = \frac{\tau_i}{N + \sum_i \tau_i}. \quad (6.2)$$

The prior probability of the model  $p(\theta)$  assigns higher probabilities to segmentation models that best manage the trade-off between having both fewer and shorter segments. This is achieved by an implicit exponential prior on morpheme lengths.

In order to control the model's tendency to oversegment or undersegment, a weight parameter  $\alpha$  is introduced into the cost function:

$$L(\mathbf{D}_W, \theta) = -\log p(\theta) - \alpha \log p(\mathbf{D}_W | \theta). \quad (6.3)$$

By reducing  $\alpha$ , it is possible to emphasise the cost function contribution due to the prior, which would favour a segmentation model consisting of shorter segments, and vice versa. We can therefore control the average segmentation length by tuning  $\alpha$ .

The training algorithm implemented by Morfessor 2.0 proceeds with a greedy, local search [73]. It does this by considering one morphological segment

at a time and evaluating each possible split of the segment into smaller segments to determine the one that minimizes the cost function with respect to the current model parameters. The model parameters themselves are then updated using the previously determined split as part of the model. This procedure of splitting and model updating is performed recursively on the segments resulting at each step. For the sake of efficiency, morphological segments are tied across all word types that contain them.

## 6.2.2 Lexicon induction using morphological segments

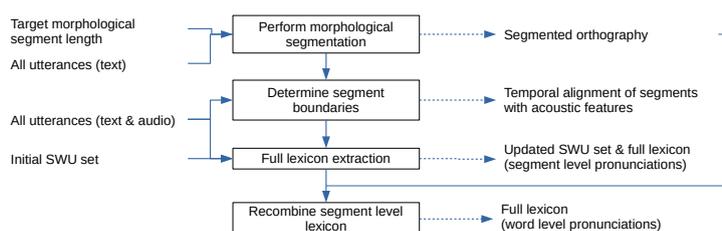


Figure 61: Overview of the steps performed during lexicon induction using morphological segments.

We now describe how we extended the approach to lexicon induction developed in the previous chapter (Section 5.2) to use morphological segments. An overview of the steps involved is given in Figure 61.

First, a morphological segmentation is performed using the approach described in Section 6.2.1. The training corpus for this segmentation is the set of unique vocabulary types in the training set orthographic transcriptions. A target average segment length is specified at this stage, and the parameter  $\alpha$  of the segmentation cost function (Equation 6.3) is tuned to achieve this.

Second, the resulting segmented orthographic transcriptions are aligned with the training set acoustic features, in order to extract pronunciations from the utterance-level SWU sequences that result from acoustic decoding. There are at least two ways to achieve this. The first is to regard the morphological segments as atomic, and train whole-segment acoustic models which can then be used for Viterbi alignment. The second option is to estimate acoustic models for grapheme SWUs and use those for acoustic alignment. The latter option is expected to yield more accurate alignments, since grapheme-level acoustic

modeling is both more robust and more granular than word level modeling. A further advantage of using graphemes for alignment is that it decouples the performance of the resulting lexicons from the accuracy of the temporal segment alignments, which is otherwise expected to be correlated with the average segment lengths.

Third, a full pronunciation lexicon for the morphological segments is extracted using the procedure described in Section 5.2, but using morphological segment-level transcriptions instead of word-level transcriptions.

Finally, in order to form a word-level pronunciation lexicon, we concatenate the newly induced pronunciations of the constituent morphological segments of each training set word.

## 6.3 Experiments and results

In this section we describe the dataset and experimental setup used to evaluate the approach to lexicon induction for agglutinating languages developed in this chapter.

### 6.3.1 Dataset and experimental setup

Table 61: Summary of the Luganda dataset used for experimental evaluation.

Dataset	Hours	#Utterances	#Word tokens
Train	9.59h	8774	18305
Test	0.29h	164	1150

The dataset used for experimentation is the Luganda corpus used in the previous chapter (see Section 5.3.1). It is summarised in Table 61. Luganda is a highly agglutinating language, as can be seen from the disproportionately large vocabulary relative to its recording time. It is also a severely under-resourced language, with practically no resources available other than the dataset used in this study. The dataset includes a pronunciation lexicon compiled by phonetic experts, which enabled the establishment of a baseline indicating what can be achieved using the conventional ASR system development approach.

In order to fairly compare lexicon induction with a morphologically segmented orthography to lexicon induction using the original orthography, we use the same initial SWU inventory and associated metaparameters (number of SWUs  $N_p = 150$  and average SWU duration  $R_p = 78\text{ms}$ ) in both cases. Further, in both cases we relied on accurate temporal alignment of the orthographic segments with the speech features.

We used the Morfessor 2.0 implementation of the approach to performing morphological segmentation described in Section 6.2.1 [73]. For the acoustic model estimation, decoding and alignment necessary during lexicon extraction, we used the HTK tools [1]. For the evaluation of ASR performance, we used the Kaldi toolkit, culminating in a system consisting of a combination of SGMM-MMI and DNN/HMM acoustic models (SGMM+DNN) with cross-word dependency [67]. Although newer and better models such as SGMM+TDNN are available, we found that they reflect similar performance trends with a relatively constant performance gain, while consuming significantly more processing time to train. In addition to the acoustic models, we trained bigram language models from corpora consisting of only the corresponding training prompts.

### 6.3.2 Context independent pronunciation induction

Table 62: Vocabulary distributions for various levels of morphological segmentation (expressed as the average segment length in graphemes), and the associated ASR performance for each system.

Lexicon	Average segment length	# Types	$n > 3$		$n > 9$		% WER
			Types	Tokens	Types	Tokens	
Phoneme							54.94%
Grapheme	1						55.14%
Auto (unsegmented)	5.7	18305	14.4%	75.0%	5.6%	63.8%	60.91%
Auto + segmentation	3.0	2135	70.12%	99.31%	49.51%	97.72%	56.35%
	2.5	836	93.42%	99.94%	82.30%	99.63%	56.35%
	2.0	230	99.57%	100.00%	99.13%	100.00%	54.95%
	1.5	70	98.57%	100.00%	97.14%	100.00%	<b>54.90%</b>
	1.0	33	96.97%	100.00%	93.94%	100.00%	55.14%

In the first set of experiments involving morphological segmentation, we investigated the effect of average segment length on the resulting vocabulary

distributions and resulting word error rates. The results of these experiments are shown in Table 62. The lexicons indicated by *Phoneme* and *Grapheme* represent the baseline results obtained using respectively an expert-compiled phonetic lexicon and a lexicon using graphemes as sub-word units. The lexicon indicated by *Auto* refers to an automatically induced lexicon with no morphological segmentation performed, i.e. using the unmodified orthography as described in Chapter 5. The lexicons indicated by *Auto+segmentation* refer to lexicons that were automatically induced using the morphologically segmented orthography.

We observe from Table 62 that even modest levels morphological segmentation profoundly reduce the number of unique types in the vocabulary, with an approximate halving of the average segment length leading to a nine-fold reduction in segment types. The frequency of occurrence of segment types also becomes much more favourable.

These promising trends are also reflected in the word error rates. Without morphological segmentation, the automatically induced lexicon is substantially outperformed by the expert and grapheme lexicons. However, even the least aggressive segmentation yields error rates that are close to (or slightly better than) the expert baseline, while still retaining a significant proportion of segment types. A further reduction in error rates is observed as the segments become shorter.

### 6.3.3 Context-dependent pronunciation induction

It was observed in the previous section that shorter morphological segments yielded better-performing lexicons. However, shorter segments also lead to a significant reduction in the number of segment types. At some point, the remaining segment types can be expected to become less able to capture the acoustically distinct behaviour that is useful for linguistic discrimination during speech recognition. In order to address this, we will attempt to increase the number of types associated with shorter morphological segments by taking the preceding and the following segments into account as contextual information. This should help to capture segment-level pronunciation variation relevant for distinguishing words.

When an inventory of context-dependent morphological segments is

established in such a way that each unique context becomes a new type, it is expected that there will be a large increase in types, many of which occur infrequently. This might counteract any performance gain achieved by introducing context-dependency. We address this by establishing a minimum occurrence threshold for each context-dependent instance of a segment. In cases where this threshold is not met, we pool infrequent contexts of a segment either by shared left or right context. If the specified threshold is still not met, then context is discarded altogether.

Table 63: The ASR performance of lexicons induced using context-dependant morphological segmentation for various pooling thresholds. A threshold of  $\infty$  indicates context independent segments.

Average segment length	Threshold	# Types	% WER
2	$\infty$	230	54.95%
	250	434	<b>54.33%</b>
	125	720	54.95%
	62	1358	55.05%
1.5	$\infty$	70	54.90%
	250	462	55.53%
	125	856	<b>54.81%</b>
	62	1573	56.83%
1	$\infty$	33	55.15%
	250	655	54.52%
	125	992	<b>54.28%</b>
	62	1347	55.87%

We applied this approach for various segment lengths and pooling thresholds, and show the resulting word error rates in Table 63. In general, we observe a performance improvement for the lexicons induced using context-dependant segments, relative to their context-independent counterparts, at least for some pooling thresholds. It is likely that there is also a trade-off. Reducing the pooling threshold increases the number of types for which independent pronunciations are obtained, but also reduces the number of observations per type and thus the robustness of the estimated pronunciations.

Another important observation is that we have been able to improve on both the expert (phoneme) baseline as well as the grapheme baseline for Luganda

(54.95% and 55.14% respectively), with the best-performing automatically induced system achieving an error rate of 54.28%. The system producing this error rate corresponds to a segment length of 1, which amounts to mapping the automatically discovered pool of acoustic SWUs to context-dependant graphemes. The improvement of such a system over a pure grapheme-based lexicon is interesting, especially because the ASR training pipeline used for performance evaluation already includes a context-dependant acoustic model expansion step (in the form of triphone creation). Since the ASR performance of the baseline grapheme lexicon already includes context-dependence, it can be deduced that the additional performance gain from using grapheme context during lexicon induction can be attributed to our SWU inventory.

## 6.4 Summary and conclusions

Table 64: Summary of the best ASR performance for each of the various systems evaluated in this study.

<b>System</b>	<b>% WER</b>
Phoneme	54.94%
Grapheme	55.14%
Auto	60.91%
Auto + segmentation	54.90%
Auto + segmentation + context	54.28%

In this chapter, we have presented an approach for improving the performance of automatic lexicon induction in the case of highly agglutinating languages. This involved performing data-driven morphological segmentation on the training set orthography before lexicon induction. Table 64 summarises the results obtain when applying this approach to Luganda. The morphological segmentation led to an approximately 10% relative reduction in word error rate compared to the best lexicon induced on an unsegmented orthography. We also demonstrated that further performance gains can be achieved by making the segments context dependent, resulting in a system that performs 1% better than one trained using an expert lexicon. This demonstrates that it is feasible to use

*CHAPTER 6. LEXICON INDUCTION FOR HIGHLY-AGGLUTINATING LANGUAGES***110**

automatically induced lexicons to facilitate ASR in an under-resourced setting even for highly agglutinating languages.

# Chapter 7

## Conclusion

### 7.1 Summary

In this thesis we have addressed the problem of automatic sub-word unit discovery and lexicon induction using only recorded speech and accompanying orthographic transcriptions. The ultimate goal of our work has been to enable automatic speech recognition for under-resourced languages by eliminating the need for a pronunciation lexicon crafted by language experts. Below follows a summary of our investigations and conclusions.

In Chapter 2 we investigated the application of convolutional sparse coding and coordinate descent to the task of discovering sub-word units in recorded speech. After applying this to a subset of the TIMIT corpus, we found it to yield units that had a degree of correspondence with the reference phonemes, as seen in Figure 71. This indicated at least some level of phonetic relevance. However, the transcriptions of the speech signals in terms of these units exhibited significant overlap between units, which rendered them unsuitable for lexicon generation. Furthermore, a high level of variation was observed across instances of the same orthographic transcription.

In Chapter 3 we improved the sparse coding model developed in Chapter 2 in such a way that the coding is constrained to use basis functions that do not overlap temporally. This constraint enabled us to develop a novel coding algorithm that produces the optimal sparse code for a given set of atoms. This sparse coding algorithm was incorporated into a novel local search algorithm

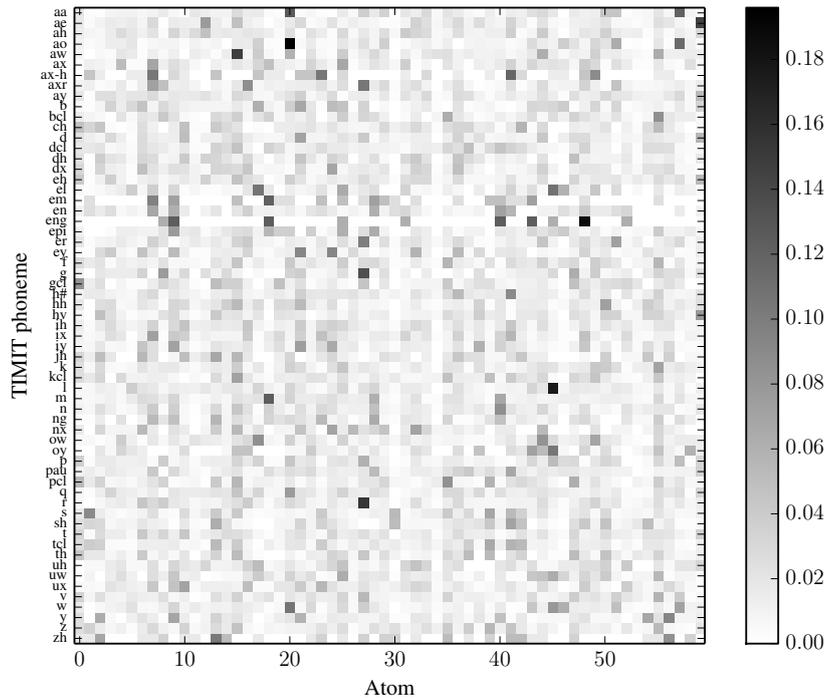


Figure 71: Coincidence matrix of the SWU inventory produced by the sparse coding approach of Chapter 2.

that iteratively improves the acoustic relevance of the automatically-determined sub-word units from a random initialization. We also attempted to find a more globally optimal set of sub-word units by developing a population-based metaheuristic optimisation procedure. The result of these interventions was automatically-determined sub-word units in our final inventories that exhibited a strong correlation with the reference phonetic transcriptions (see Figure 72). Unfortunately, the resulting transcriptions were still too variable to be directly useful for pronunciation modelling in ASR.

In Chapter 4 we abandoned the approach of using sparse code signal reconstruction to tokenize speech into SWUs. Instead, we turned to using a more conventional HMM-GMM approach, in which the speech transcription is viewed as a maximum likelihood state sequence through the HMM-GMM model with SWUs as groups of states. However, the notion of a sparse transcription was retained through the use of an insertion penalty. Using this HMM-GMM approach, we investigated the application of two novel lattice-constrained Viterbi training strategies in an attempt to reduce transcription variability and

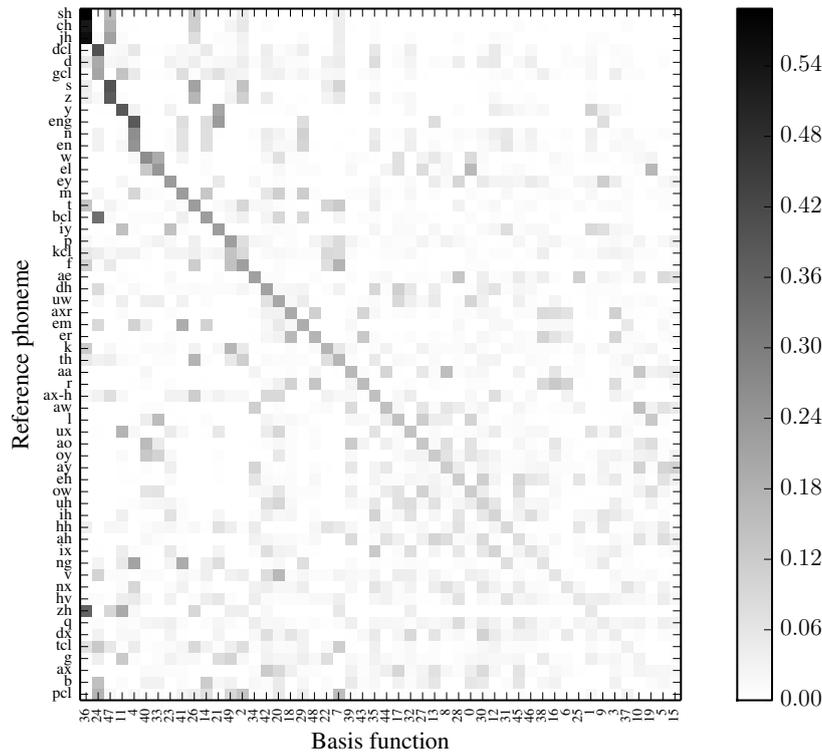


Figure 72: Coincidence matrix of the SWU inventory produced by the non-overlapping sparse coding model of Chapter 3.

improve the SWU inventories. The first of these attempted to jointly learn a bigram SWU language model along with the evolving SWU inventory. We found that this substantially increases correspondence of the SWUs with expert-defined reference phonemes on the TIMIT dataset, but did little to improve pronunciation consistency. In order to address this, we jointly inferred an SWU inventory alongside an SWU pronunciation model for each word in the training vocabulary, and then used these pronunciation models to constrain the transcriptions. This lightly supervised approach delivered similar increased correspondence with the reference phonemes (see Figure 73), but in this case also improved pronunciation consistency. Chapter 4 also included a first attempt to evaluate ASR performance using a automatically induced pronunciation lexicon. The result was reasonable SWU pronunciations for short, frequent words, but poor SWU transcriptions for longer and uncommon words, which was seen to negatively affect ASR performance.

In Chapter 5 we attempted to improve the quality of the induced lexicons

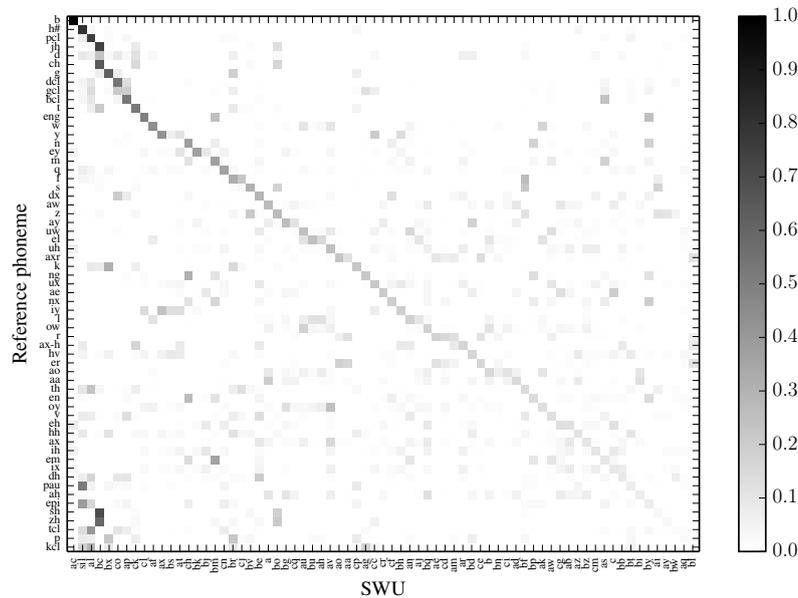


Figure 73: Coincidence matrix of the SWU inventory produced by the word level pronunciation model constrained Viterbi training introduced in Chapter 4.

using two further interventions. The first of these addressed the quality of the SWU inventory by the introducing a novel SWU discovery approach based on self-organising HMM-GMM states that incorporate orthographic knowledge during SWU discovery. This yielded an SWU inventory that had a very close correspondence with TIMIT’s reference phonemes, as can be seen in Figure 74. Secondly, we attempted to improve the quality of the discovered pronunciations by using more sophisticated pronunciation modeling and pruning approaches. We applied these methods to corpora of recorded radio broadcasts in Ugandan English, Luganda and Acholi, all under-resourced. We demonstrated that our proposed method is able to discover lexicons that perform as well as baseline expert systems for Acholi, and close to this level for the other two languages when used to train DNN-HMM ASR systems. The worst performing language was Luganda, which has a highly agglutinating vocabulary that makes automatic lexicon induction particularly challenging.

In Chapter 6 we sought to improve the performance of our automatic lexicon induction method specifically in the highly agglutinating setting, which was observed to be a key weak point of the approach developed in Chapter 5. We addressed the unfavorable vocabulary distributions of such languages by

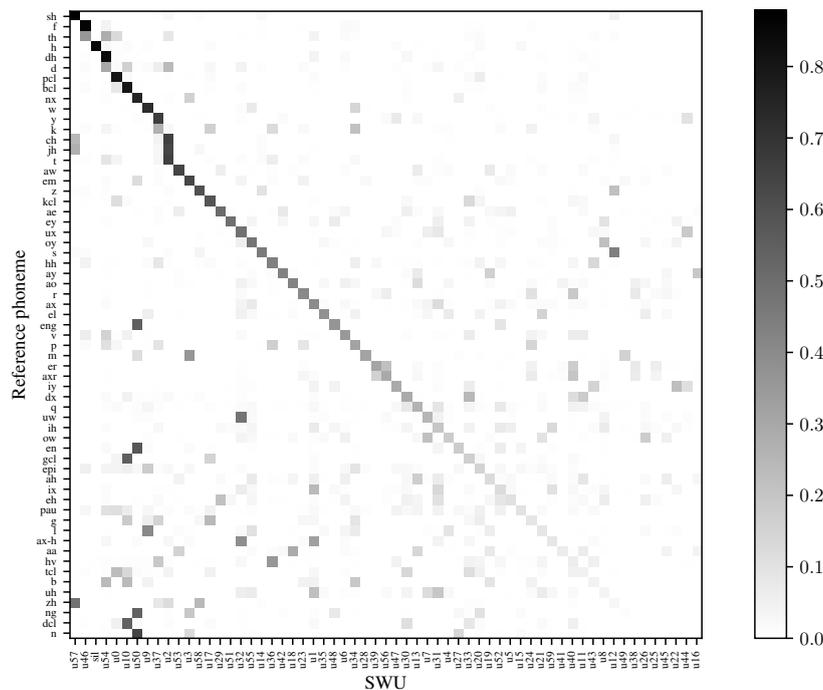


Figure 74: Coincidence matrix of the SWU inventory produced in Chapter 5 through the agglomerative clustering of self-organising word-level HMM-GMM states.

performing data-driven morphological segmentation of the orthography prior to lexicon induction. We applied this novel step to the same Luganda corpus used in Chapter 5. The intervention led to a 10% (relative) reduction in WER, putting the resulting ASR performance on par with an expert lexicon. When context was added to the morphological segments prior to lexicon induction, a further 1% WER (relative) reduction was achieved.

Taken together, the techniques developed in Chapters 2–6 demonstrate that it is feasible to perform ASR in an under-resourced setting using an automatically induced lexicon, even in the case of a highly agglutinating language.

## 7.2 Summary of contributions

In this work we have made the following key contributions:

1. **Chapter 2** — An investigation into the feasibility of using the sparse coding framework for discovering sub-word units in recorded speech.

2. **Chapter 3** — A novel shift and scale invariant sparse coding framework that is constrained such that no basis functions overlap in time. Associated with this we have contributed:
  - a) An algorithm that determines the optimal sparse code, given an utterance and a set of basis functions.
  - b) A search method that iteratively improves a sparse code and set of basis functions until it converges to a local optimum.
  - c) A metaheuristic search method to determine more globally optimal SWU inventories.
3. **Chapter 4** — Two lattice constrained Viterbi training methods to improve an existing SWU inventory (such as one determined through sparse coding):
  - a) A completely unsupervised training method that reinforces the frequently observed SWU patterns in the training set.
  - b) A lightly supervised method that uses knowledge of the orthographic transcriptions to constrain the SWU sequences that are generated for each word.
4. **Chapter 5** — An end-to-end approach to ASR training that requires only recorded speech and orthographic transcriptions. This involves the following contributions:
  - a) A novel SWU discovery method that relies on agglomerative clustering of tied self-organising HMM states.
  - b) A novel lexicon induction method that performs iterated state level pruning of an HMM pronunciation model as well as iterative realignment.
5. **Chapter 6** — An approach to developing higher quality lexicons for an agglutinating task language by incorporating data-driven morphological segmentation into the methods developed in Chapter 5.

## 7.3 Future research

In this section we discuss some potential avenues for future research that build on the work presented in this thesis.

### 7.3.1 Global optimisation of SWU inventory

In Chapter 5, sub-word units were discovered by clustering a large number of HMM-GMM states that were found by allowing word-level models to self-organise across all instances of a word type. In that chapter, the clustering itself was performed using a greedy, agglomerative approach, which may not be globally optimal. Future research could investigate the application of a more globally exhaustive search for an optimal clustering of the word-level substates into a compact SWU inventory. One candidate for this is the metaheuristic procedure that was applied to the sparse coding units in Chapter 3.

### 7.3.2 Optimising with respect to ASR performance

#### 7.3.2.1 Sub-word unit inventory

In this work, the optimisation of sub-word units was performed with respect to an objective function that essentially evaluated how well the units fit the training corpus audio features, for example by using the signal reconstruction residual in the case of sparse coding, or the likelihood of the training data being generated by the trained HMM-GMMs. This approach is standard in machine learning, and in a purely unsupervised setting it is both justified and inevitable. However, if the task is to develop lexicons that deliver the best possible ASR performance, and accompanying orthography is available to provide light supervision, then using ASR performance as a clustering objective from the outset would be beneficial. Future work could investigate SWU clustering approaches that either directly optimise ASR performance, or a more effective proxy for it than signal fit. An example of such a proxy may be a procedure that takes as input an existing lexicon and then applies a clustering objective that trades off producing a minimal number of identical pronunciations for different word types with the overall model likelihood of the clustered set.

### 7.3.2.2 Lexicon

In a similar vein, the lexicon induction performed in this work has relied on iterative pruning of a candidate set of pronunciations. This pruning made use of a strategy of retaining either the most frequently occurring components of candidate pronunciations or the most frequently occurring whole pronunciations. It is however possible that the most frequent pronunciation isn't necessarily the best overall compromise in terms of recognition accuracy. Although the use of an iterative pruning procedure that only removes a small number of the least frequent pronunciations at each iteration does to some extent allow compromise candidates to emerge, this does not guarantee producing the best pronunciation for ASR performance. Future work could investigate strategies for pronunciation pruning that more deliberately target ASR performance.

### 7.3.3 Leveraging multilingual datasets

Multilingual ASR training is a strategy for training ASR systems that take advantage of datasets comprising languages other than just the target language. Such a training approach has the potential to boost the performance of under-resourced languages by increasing the amount data available for training acoustic models. However, one key challenge in realising this lies in coordinating the disparate phone sets of multiple languages such that it becomes possible to share acoustic models for the SWUs they have in common. Future work could address this challenge by applying the automatic sub-word unit discovery and lexicon induction procedures developed in this work to multi-lingual training sets. This would be advantageous, since doing so would yield a single coherent SWU set that is shared between all languages in the training set, and which was derived in a fully data-driven fashion.

## List of References

- [1] Young, S.J., Evermann, G., Gales, M.J.F., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V. and Woodland, P.C.: The HTK book, version 3.4. 2006.
- [2] Zhang, X., Manohar, V., Povey, D. and Khudanpur, S.: Acoustic data-driven lexicon learning based on a greedy pronunciation selection framework. In: *Proceedings of Interspeech*. 2017.
- [3] Chen, G., Povey, D. and Khudanpur, S.: Acoustic data-driven pronunciation lexicon generation for logographic languages. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5350–5354. March 2016.
- [4] Goel, N., Thomas, S., Agarwal, M., Akyazi, P., Burget, L., Feng, K., Ghoshal, A., Glembek, O., Karafiát, M., Povey, D., Rastrow, A., Rose, R.C. and Schwarz, P.: Approaches to automatic lexicon learning with limited training examples. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5094–5097. March 2010. ISSN 1520-6149.
- [5] Lu, L., Ghoshal, A. and Renals, S.: Acoustic data-driven pronunciation lexicon for large vocabulary speech recognition. In: *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pp. 374–379. IEEE, 2013.
- [6] Ravishankar, M. and Eskenazi, M.: Automatic generation of context-dependent pronunciations. In: *Fifth European Conference on Speech Communication and Technology*. 1997.
- [7] ten Bosch, L. and Cranen, B.: A computational model for unsupervised word discovery. In: *Proceedings of Interspeech*, pp. 1481–1484. 2007.
- [8] Goussard, G. and Niesler, T.R.: Automatic discovery of subword units and pronunciations for automatic speech recognition using TIMIT. In: *Proceedings of*

- the Annual Symposium of the Pattern Recognition Society of South Africa (PRASA)*. 2010.
- [9] hung Siu, M., Gish, H., Chan, A., Belfield, W. and Lowe, S.: Unsupervised training of an HMM-based self-organizing unit recognizer with applications to topic classification and keyword discovery. *Computer Speech & Language*, vol. 28, no. 1, pp. 210 – 223, 2014. ISSN 0885-2308.
- [10] Svendsen, T. and Soong, E.: On the automatic segmentation of speech signals. In: *Proceedings of ICASSP '87. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 12, pp. 77–80. Apr 1987.
- [11] Sharma, M. and Mammone, R.: Blind speech segmentation: automatic segmentation of speech without linguistic knowledge. In: *Proceedings of the Fourth International Conference on Spoken Language (ICSLP 96)*, vol. 2, pp. 1237–1240. Oct 1996.
- [12] Bacchiani, M. and Ostendorf, M.: Joint lexicon, acoustic unit inventory and model design. *Speech Communication*, vol. 29, pp. 99 – 114, 1999. ISSN 0167-6393.
- [13] van Vuuren, V.Z., ten Bosch, L. and Niesler, T.: Automatic segmentation of TIMIT by dynamic programming. In: *Proceedings of the Annual Symposium of the Pattern Recognition Society of South Africa (PRASA)*. 2012.
- [14] Varadarajan, B. and Khudanpur, S.: Automatically learning speaker-independent acoustic subword units. In: *Proceedings of Interspeech*, pp. 1333–1336. 2008.
- [15] Jansen, A. and Church, K.: Towards unsupervised training of speaker independent acoustic models. In: *Proceedings of Interspeech*, pp. 1693–1692. 2011.
- [16] Wang, H., Lee, T., Leung, C.-C., Ma, B. and Li, H.: Unsupervised mining of acoustic subword units with segment-level Gaussian posteriorgrams. In: *Proceedings of Interspeech*, pp. 2297–2301. 2013.
- [17] Razavi, M.: An HMM-Based Formalism for Automatic Subword Unit Derivation and Pronunciation Generation. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015.
- [18] Razavi, M., Rasipuram, R. and Magimai.-Doss, M.: Towards weakly supervised acoustic subword unit discovery and lexicon development using hidden Markov models. *Speech Communication*, vol. 96, pp. 168 – 183, 2018. ISSN 0167-6393.

- Available at: <http://www.sciencedirect.com/science/article/pii/S016763931730105X>
- [19] Lerato, L. and Niesler, T.R.: Clustering acoustic segments using multi-stage agglomerative hierarchical clustering. *PLoS ONE*, vol. 10, no. 10, p. e0141756, 10 2015.
- [20] Lee, C.-y. and Glass, J.: A nonparametric Bayesian approach to acoustic model discovery. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pp. 40–49. Association for Computational Linguistics, 2012.
- [21] Ondel, L., Burget, L. and Černocký, J.: Variational inference for acoustic unit discovery. *Procedia Computer Science*, vol. 81, no. Supplement C, pp. 80 – 86, 2016. ISSN 1877-0509. SLTU-2016 5th Workshop on Spoken Language Technologies for Under-resourced languages 09-12 May 2016 Yogyakarta, Indonesia.
- [22] Liu, C., Yang, J., Sun, M., Kesiraju, S., Rott, A., Ondel, L., Ghahremani, P., Dehak, N., Burget, L. and Khudanpur, S.: An empirical evaluation of zero resource acoustic unit discovery. In: *Proceedings of Interspeech*. 2017.
- [23] Torbati, A.H.H.N., Picone, J. and Sobel, M.: Speech acoustic unit segmentation using hierarchical Dirichlet processes. In: *Proceedings of Interspeech*, pp. 637–641. 2013.
- [24] Lee, C.-y., Zhang, Y. and Glass, J.R.: Joint learning of phonetic units and word pronunciations for ASR. In: *Proceedings of Empirical Methods on Natural Language Processing (EMNLP)*, pp. 182–192. 2013.
- [25] Lee, C.-y., O'Donnell, T.J. and Glass, J.: Unsupervised lexicon discovery from acoustic input. *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 389–403, 2015.
- [26] Singh, R., Raj, B. and Stern, R.: Automatic generation of subword units for speech recognition systems. *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 2, pp. 89–99, Feb 2002. ISSN 1063-6676.
- [27] Harwath, D. and Glass, J.: Speech recognition without a lexicon - bridging the gap between graphemic and phonetic systems. In: *Proceedings of Interspeech*. 2014.

- [28] Graves, A. and Jaitly, N.: Towards end-to-end speech recognition with recurrent neural networks. In: *Proceedings of the 31st International Conference on International Conference on Machine Learning*, vol. 32.
- [29] Zweig, G., Yu, C., Droppo, J. and Stolcke, A.: Advances in all-neural speech recognition. In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 4805–4809. IEEE, 2017.
- [30] Soltau, H., Liao, H. and Sak, H.: Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition. pp. 3707–3711. 08 2017.
- [31] Graves, A., Fernández, S., Gomez, F. and Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376. ACM, 2006.
- [32] Graves, A.: Sequence transduction with recurrent neural networks. *arXiv*, vol. abs/1211.3711, 2012. 1211.3711.  
Available at: <http://arxiv.org/abs/1211.3711>
- [33] Graves, A., Mohamed, A. and Hinton, G.: Speech recognition with deep recurrent neural networks. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649. May 2013. ISSN 2379-190X.
- [34] Chan, W., Jaitly, N., Le, Q. and Vinyals, O.: Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4960–4964. March 2016. ISSN 2379-190X.
- [35] Chorowski, J.K., Bahdanau, D., Serdyuk, D., Cho, K. and Bengio, Y.: Attention-based models for speech recognition. In: *Advances in neural information processing systems*, pp. 577–585. 2015.
- [36] Zhang, Y., Chan, W. and Jaitly, N.: Very deep convolutional networks for end-to-end speech recognition. In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 4845–4849. IEEE, 2017.
- [37] Lu, L., Zhang, X. and Renais, S.: On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition. In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 5060–5064. IEEE, 2016.

- [38] Zweig, G., Yu, C., Droppo, J. and Stolcke, A.: Advances in all-neural speech recognition. In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 4805–4809. IEEE, 2017.
- [39] Prabhavalkar, R., Rao, K., Sainath, T., Li, B., Johnson, L. and Jaitly, N.: A comparison of sequence-to-sequence models for speech recognition. In: *Proceedings of Interspeech*. 2017.
- [40] Hyvarinen, A., Oja, E., Hoyer, P. and Hurri, J.: Image feature extraction by sparse coding and independent component analysis. In: *Proceedings. Fourteenth International Conference on Pattern Recognition*, vol. 2, pp. 1268–1273 vol.2. Aug 1998.
- [41] Kavukcuoglu, K., Sermanet, P., Boureau, Y.-L., Gregor, K., Mathieu, M. and LeCun, Y.: Learning convolutional feature hierarchies for visual recognition. In: *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1, NIPS10*, p. 10901098. Curran Associates Inc., Red Hook, NY, USA, 2010.
- [42] Olshausen, B.A.: Sparse codes and spikes. In: *Probabilistic models of the brain: Perception and neural function*, pp. 257–272. MIT Press, 2001.
- [43] Natarajan, B.K.: Sparse approximate solutions to linear systems. *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, April 1995. ISSN 0097-5397.  
Available at: <http://dx.doi.org/10.1137/S0097539792240406>
- [44] Smit, W.: Sparse coding of single spoken digits. In: *Proceedings of the Annual Symposium of the Pattern Recognition Society of South Africa (PRASA)*. 2013.
- [45] Sivaram, G.S.V.S., Nemala, S., Elhilali, M., Tran, T. and Hermansky, H.: Sparse coding for speech recognition. In: *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 4346–4349. March 2010. ISSN 1520-6149.
- [46] Smit, W. and Barnard, E.: Continuous speech recognition with sparse coding. *Computer Speech & Language*, vol. 23, no. 2, pp. 200–219, 2009. ISSN 0885-2308.
- [47] Grosse, R., Raina, R., Kwong, H. and Ng, A.Y.: Shift-invariant sparse coding for audio classification. In: *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, UAI07*, p. 149158. AUAI Press, Arlington, Virginia, USA, 2007. ISBN 0974903930.

- [48] Mørup, M., Schmidt, M.N. and Hansen, L.K.: Shift invariant sparse coding of image and music data. Tech. Rep., 2008.  
Available at: <http://www2.imm.dtu.dk/pubdb/p.php?4659>
- [49] Vinyals, O. and Deng, L.: Are sparse representations rich enough for acoustic modeling? In: *Proceedings of Interspeech*. 2012.
- [50] Aharon, M., Elad, M. and Bruckstein, A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, Nov 2006. ISSN 1053-587X.
- [51] Olshausen, B.A. and Field, D.J.: Sparse coding with an overcomplete basis set: a strategy employed by v1? *Vision Research*, vol. 37, pp. 3311–3325, 1997.
- [52] Mallat, S. and Zhang, Z.: Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, vol. 41, no. 12, pp. 3397–3415, Dec 1993. ISSN 1053-587X.
- [53] Tropp, J.: Greed is good: algorithmic results for sparse approximation. *Information Theory, IEEE Transactions on*, vol. 50, no. 10, pp. 2231–2242, Oct 2004. ISSN 0018-9448.
- [54] Jones, L.K.: On a conjecture of Huber concerning the convergence of projection pursuit regression. *The Annals of Statistics*, vol. 15, no. 2, pp. pp. 880–882, 1987. ISSN 00905364.  
Available at: <http://www.jstor.org/stable/2241347>
- [55] Pati, Y., Rezaiifar, R. and Krishnaprasad, P.S.: Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In: *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pp. 40–44 vol.1. Nov 1993. ISSN 1058-6393.
- [56] Li, Y. and Osher, S.: Coordinate descent optimization for  $l_1$  minimization with application to compressed sensing; a greedy algorithm. *Inverse Probl. Imaging*, vol. 3, no. 3, pp. 487–503, 2009.
- [57] Lewicki, M.S. and Sejnowski, T.J.: Coding time-varying signals using sparse, shift-invariant representations. 1998.
- [58] Plumbley, M.D., Abdallah, S.A., Blumensath, T. and Davies, M.E.: Sparse representations of polyphonic music. *Signal Processing*, vol. 86, pp. 417–431, 2006.

- [59] Baker, J.E.: Adaptive selection methods for genetic algorithms. In: *Proceedings of an International Conference on Genetic Algorithms and their applications*, pp. 101–111. Hillsdale, New Jersey, 1985.
- [60] Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: *Proceedings of the second international conference on genetic algorithms*, pp. 14–21. 1987.
- [61] Bisani, M. and Ney, H.: Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, vol. 50, no. 5, pp. 434 – 451, 2008. ISSN 0167-6393.
- [62] Young, S.J., Evermann, G., Gales, M.J.F., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V. and Woodland, P.C.: The HTK book, version 3.4. 2006.
- [63] Agenbag, W. and Niesler, T.R.: Refining sparse coding sub-word unit inventories with lattice-constrained Viterbi training. In: *Proceedings of the Workshop on Spoken Language Technologies for Under-resourced languages*. 2016.
- [64] Saeb, A., Menon, R., Cameron, H., Kibira, W., Quinn, J. and Niesler, T.: Very low resource radio browsing for agile developmental and humanitarian monitoring. In: *Proceedings of Interspeech*, pp. 2118–2122. August 2017.
- [65] Menon, R., Saeb, A., Cameron, H., Kibira, W., Quinn, J. and Niesler, T.: Radio-browsing for developmental monitoring in Uganda. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5795–5799. March 2017.
- [66] Panayotov, V., Chen, G., Povey, D. and Khudanpur, S.: Librispeech: An ASR corpus based on public domain audio books. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210. April 2015. ISSN 2379-190X.
- [67] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G. and Vesely, K.: The Kaldi speech recognition toolkit. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011.

- [68] Bisani, M. and Ney, H.: Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, vol. 50, no. 5, pp. 434–451, 2008. ISSN 0167-6393.  
Available at: <http://www.sciencedirect.com/science/article/pii/S0167639308000046>
- [69] S Garofolo, J., Lamel, L., M Fisher, W., Fiscus, J., S. Pallett, D., L. Dahlgren, N. and Zue, V.: TIMIT acoustic-phonetic continuous speech corpus. 11 1992.
- [70] Creutz, M. and Lagus, K.: Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Tech. Rep. A81, Publications in Computer and Information Science, Helsinki University of Technology Helsinki, 2005.
- [71] Kohonen, O., Virpioja, S. and Lagus, K.: Semi-supervised learning of concatenative morphology. In: *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pp. 78–86. Association for Computational Linguistics, 2010.
- [72] Virpioja, S., Kohonen, O. and Lagus, K.: Evaluating the effect of word frequencies in a probabilistic generative model of morphology. In: *Proceedings of the 18th Nordic conference of computational linguistics (NODALIDA 2011)*, pp. 230–237. 2011.
- [73] Virpioja, S., Smit, P., Grönroos, S.-A. and Kurimo, M.: Morfessor 2.0: Python implementation and extensions for Morfessor baseline. 2013.