Unsupervised Learning for Text-to-Speech Synthesis

Oliver Watts

Thesis submitted for the degree of Doctor of Philosophy

The University of Edinburgh

2012

Declaration

I have composed this thesis. The work in it is my own unless stated.

Oliver Watts

Acknowledgements

I would like to thank the following people:

- Simon King and Junichi Yamagishi, for all of their supervision, guidance and advice
- All those who provided useful feedback after earlier presentations of this work, in particular Steve Renals and Rob Clark
- Everybody at the Centre for Speech Technology Research, for creating such a friendly and stimulating place to work
- Karl Isaac for helping set up an experiment using Mechanical Turk, and Cássia Valentini-Botinhão for helping to recruit local listeners for experiments
- Adriana Stan at the University of Cluj-Napoca for providing the raw version of the newstext data used in Chapter 9 (and indeed for having made the whole of her Romanian speech corpus generally available), and for recruiting participants for the evaluation of Romanian systems
- Martti Vainio and Antti Suni at the University of Helsinki for making the Finnish database available for the experiments described in Chapter 8
- Antti Suni (again) and Heini Kallio at the University of Helsinki and Tuomo Raitio at Aalto University for recruiting participants for the evaluation of Finnish systems.

This work was made possible by a Doctoral Training Award from the Engineering and Physical Sciences Research Council (EPSRC: `http://www.epsrc.ac.uk`), and has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF: `http://www.ecdf.ed.ac.uk`). The ECDF is partially supported by the eDIKT initiative (`http://www.edikt.org.uk`).

Abstract

This thesis introduces a general method for incorporating the distributional analysis of textual and linguistic objects into text-to-speech (TTS) conversion systems. Conventional TTS conversion uses intermediate layers of representation to bridge the gap between text and speech. Collecting the annotated data needed to produce these intermediate layers is a far from trivial task, possibly prohibitively so for languages in which no such resources are in existence. Distributional analysis, in contrast, proceeds in an unsupervised manner, and so enables the creation of systems using textual data that are not annotated. The method therefore aids the building of systems for languages in which conventional linguistic resources are scarce, but is not restricted to these languages.

The distributional analysis proposed here places the textual objects analysed in a continuous-valued space, rather than specifying a hard categorisation of those objects. This space is then partitioned during the training of acoustic models for synthesis, so that the models generalise over objects' surface forms in a way that is acoustically relevant.

The method is applied to three levels of textual analysis: to the characterisation of sub-syllabic units, word units and utterances. Entire systems for three languages (English, Finnish and Romanian) are built with no reliance on manually labelled data or language-specific expertise. Results of a subjective evaluation are presented.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Goals and Approach

The social value of speech synthesis technology is obvious: it is an indispensible ingredient, for example, of devices that allow the blind to interpret written documents and allow the mute to 'speak'. Systems for performing the automatic conversion of text to speech are making steady improvements year after year, and speaker-adaptive methods now allow the swift creation of personalised speaking aids for those that are losing the ability to speak, so long as the target language belongs to the small group for which synthesisers already exist. Moving to a new language, however, is more problematic, as typically the linguistically-labelled data necessary to build a synthesis system in that language (in the conventional supervised way) will not exist. Collecting such resources for a new language is time-consuming and represents the major bottleneck in developing systems for new languages. Collecting speech and unlabelled text data, and transcribing a small quantity of speech using standard orthography, in contrast, are not difficult tasks. The goal of this thesis is to formulate and test a framework for exploiting this data directly, side-stepping to some extent the need for expensive labelled data.

The central claim which this thesis examines is the following:

> A corpus of speech transcribed in standard orthography provides the basis for the construction of a high-quality automatic text-to-speech (TTS) conversion system, without the explicit representation of intermediate layers derived from linguistic knowledge, provided that:
>
> - Orthographic units are not too sparsely represented in the cor-

pus;

- An appropriate method of finding representations that generalise over these orthographic units is specified.

  The representations should rely on speech and text data along with general notions of context and similarity, and not on some manually-specified intermediate layers of representation.

The rest of this introduction will give a brief overview of the contents of this thesis, with particular reference to how individual chapters relate to the individual points of this central claim.

1.2 Organisation of this thesis

Chapter 2 provides background information on which the rest of the thesis builds. It starts with an overview of the speech synthesis framework which forms the basis of the experiments presented in this thesis. Central to learning a model for TTS in this framework is what is termed the *primary corpus*: a corpus of speech recordings associated at the utterance level with plain orthography transcriptions. This is the *corpus of speech transcribed in standard orthography* in the claim stated above. The collection of a reasonably sized primary corpus presents little difficulty. However, the conventional approach to TTS conversion that is described relies on intermediate layers of representation to bridge the gap between the two halves (text and speech) of this primary corpus. These intermediate layers are founded on linguistic knowledge and made up of elements such as phonemes, part of speech tags and intonational phrases. To be able to predict these representations from text, statistical models are generally used. To train these models in the conventional way requires expert-annotated data (for example, text annotated with phrase-breaks or with part of speech tags). These data are here termed *secondary data*, and their collection represents the major difficulty and expense in building TTS systems for new languages. The difficulty of obtaining these labelled data is the problem addressed by this thesis. After describing the conventional approach, therefore, some possible alternatives are described. Then the approach adopted in this thesis is briefly sketched: this approach makes no use of secondary data, relying on a corpus of relatively simple-to-collect primary data, and on a resource even simpler to collect which will be called *tertiary data*: unlabelled text data. Some of the assumptions made in this thesis about the target language script are stated. For example, languages with ideographic scripts

are beyond the scope of this thesis, as they do not meet the first condition made in the claim stated above for a speech corpus of reasonable size: *orthographic units are not too sparsely represented in the corpus.*

Before seeking to replace the hard-to-obtain features of a conventional synthesiser, however, it is clearly desirable to know what they contribute to system performance in cases where they are available. **Chapter 3** therefore seeks to determine what the different elements of a conventional TTS system add to the quality of the speech it generates. This is done through a series of three experiments, each of which focuses on a different area of the representation used by a conventional TTS system. The first focuses on the relatively high-level features derived from part of speech, phrase-break and symbolic intonation annotation. The second and third focus on the sub-syllabic level, looking at the impact of a phonemic transcription and of phonetic categories on the quality of synthetic speech, respectively. Experiments 2 and 3 also include initial attempts at inducing richer representations automatically from the primary corpus, of letters-in-context and phoneme types respectively. These initial attempts meet with some success, but some of their shortcomings are also discussed.

An unsupervised approach to learning representations for orthographic objects and incorporating them into a TTS conversion system is introduced in **Chapter 4**. This *distributional–acoustic method* is what is called a *method of finding representations that generalise over orthographic units* in the claim made above. It overcomes problems with the methods presented in Chapter 3 and underpins the work presented in the rest of the thesis. Importantly, the representations output by distributional analysis are points in a continuous *vector space* rather than category labels. These continuous features leave hard categorisation of textual objects open until later stages, when decision tree-based clustering can find divisions of the space of objects that are acoustically relevant and therefore pertinent to TTS.

The following two chapters describe the implementation of vector space models (VSMs) on two different levels of textual analysis, their incorporation into TTS systems, and their evaluation. Sub-syllabic units (letters and phonemes) are the focus of **Chapter 5** where an experiment using vector space model features for acoustic state-tying is described. Word-level units are the focus of two experiments in **Chapter 6**, where VSM-derived features are used for phrase-break prediction and for acoustic state-tying. The experiment of chapter 5 and the first one of chapter 6 both produce promising results, where automatically-generated features close most of the performance gap between baseline systems

and systems using expert-encoded linguistic knowledge.

Chapter 7 considers developments to the techniques already described. An obvious attraction of the distributional–acoustic method is that the quantity of data used in the distribuational phase is unrestricted by the need for manual annotation. First, therefore, the ability of the VSM of word types described in Chapter 6 to exploit larger quantities of training data is examined. In the second experiment using word-level features in Chapter 6, results were mixed and did not conform to predictions. It is hypothesised that a more robust method of feature selection than that provided by conventional decision-tree building algorithms is needed to exploit the numerous, noisy, and often irrelevant features provided by distributional analysis. This hypothesis is tested in Chapter 7: a method of feature selection using ensembles of trees is developed, and applied to the same task attempted in the second experiment of chapter 6, with good results. Chapter 7 concludes with the extension of the distributional method to the level of the utterance. This part of the thesis is particularly relevant given current trends in TTS research, where large, continuously-recorded databases are starting to be tackled, in which utterances occur in some genuine discourse context. However, a full evaluation of the incorporation of utterance-level features into actual TTS systems is not undertaken. Instead, the ability of a space representing utterances to separate known types of utterance is evaluated.

Throughout the first 7 chapters of this thesis, the techniques described are tested with reference to a single language, English. Furthermore, evaluations are done in a piecemeal fashion, where the systems tested combine experimental modules with ones from conventional systems in order to make controlled comparisons. **Chapter 8**, in contrast, applies the distributional–acoustic approach developed in this thesis to all modules of systems for three languages. These systems use the distributional–acoustic approach in an end-to-end fashion: this technique is used to obtain features on the three levels of analysis convered in Chapters 5–7 (letter, word and utterance), and only minimal human intervention is made in the construction of entire systems. Listening tests are conducted in three languages, and results presented.

Chapter 2

Background

2.1 Conventional Systems

In essence, the challenge of text-to-speech (TTS) conversion can be expressed as the following problem: given a sample of text, what is an appropriate waveform? In the most general terms, a TTS conversion system can be viewed as specifying a mapping from any arbitrary text to an appropriate waveform. The learning sample, from which this mapping can be learned during the training of a corpus-based TTS system, takes the form of a collection of recordings of spoken utterances, associated at the utterance level with transcriptions in plain orthography of the contents of those recordings. This is the type of database fundamental to conventional approaches to corpus-based speech synthesis, and will here be called the *primary corpus*.

However, in conventional systems, this mapping is not attempted directly: the formidable gap between text and speech is bridged by what will here be called a *linguistic specification*. This specification is given in terms of features based on linguistic knowledge, such as phonemes, syllables, intonational phrases, etc. TTS systems are therefore made up of two components: a textual–linguistic analyser (which will here be called a *front-end*) and a waveform generator (here called a *back-end*). The front-end accepts text as input and returns a linguistic specification of the utterance to be synthesised (a sequence of feature vectors representing prosodically enriched phonemes). The back-end takes this specification and converts it into an appropriate acoustic waveform. In modern corpus-based systems, both these components are typically trained on data. The purpose of this section is to give an overview of some methods commonly used to build these two major components of conventional speech synthesis systems. Emphasis will be placed

upon the type and quantity of data that are required for each component, as it is the limitations imposed by the need to collect this data that motivates the work presented in this thesis.

2.1.1 Synthesiser Front-end: Linguistic modelling

The term *topline systems* will here be used to denote the conventional benchmark systems for English that are used in the experiments presented in the rest of this thesis. The front-end used in a topline system accepts the text of an utterance that is to be synthesised and returns a list of context-dependent representations of speech segments to be synthesised. The representation of each segment consists of a set of Boolean values, that can be considered answers to yes–no questions about the environment in which it occurs.[1] Take for example, the second instance of *a* in the word *Batavia*, as it occurrs in the utterance (from Twain, 1880):

> We came in the Batavia – Cunard, you know.

The front-end used in these experiments associates a phonetic–prosodic segment with this letter, which can be expressed as a set of almost 2000 Boolean values. These values can be thought of as the system's answers to questions such as:

1. Is the phoneme to the left /t/?
2. Is the phoneme to the left alveolar?
3. Is the left-hand neighbouring word a pronoun?
4. Is it less than 3 syllables till the end of the prosodic phrase?
5. Is the syllable in which this segment belongs pitch-accented?

A summary of these features is given in Table 2.1. To provide answers to these questions, the front-end must obviously perform some analysis of the text input, using some established sets of linguistic features. To answer each of the five

[1] It should be noted that this description of the division of systems built into front- and back-end abstracts away from some details of implementation for clarity. However, this may be initially confusing to readers who have worked with similar systems. To further clarify therefore, it should be pointed out that each of the context-dependent representations of phonetic segments mentioned here is the set of yes–no answers to each of the questions in a context clustering question set that are true of the segment in question. Devising this question set is here considered a front-end task. The context labels used to interface between the output of the front-end (in the form of a Heterogeneous Relation Graph, for example (Taylor et al., 2001)) and this question set are simply an arbitrary set of strings, whose role is to compactly encode the set of Boolean values that constitutes this context-dependent representation.

Table 2.1: *Standard set of linguistic contexts used in English topline systems. Features are divided into classes with identifiers for ease of reference elsewhere in the thesis.*

F	**Features relating to phonemes**
F1	Monophone (identity of current phoneme)
F2	Triphone (identities of immediate neighbours)
F3	Quinphone (identities of neighbours-but-one)
FC	**Features relating to classes of phoneme**
FC1	Monophone (phonetic categories of current phoneme)
FC2	Triphone (phonetic categories of immediate neighbours)
FC3	Quinphone (phonetic categories of neighbours-but-one)
S	**Features relating to syllables**
S1	Number of segments {since, until} syllable boundary
S2	Lexical stress of syllable
S3	Size of syllable in segments
S4	Position of syllable in word
S5	Vowel of syllable
S6	Number of syllables {since, until} stress
S7	Size of word in syllables
W	**Features relating to words**
W1	Guess parts of speech of the current word and its neighbours
W2	Distance (in words) {since, until} a content word
P	**Features relating to phrase**
P1	Number of syllables {since, until} phrase boundary
P2	Number of stressed syllables {since, until} phrase boundary
P3	Number of words {since, until} phrase boundary
P4	Number of content words till phrase boundary
P5	Number of syllables in phrase
P6	Number of words in phrase
P7	Number of phrases {since, until} utterance boundary
P8	Number of phrases inutterance
T	**Features relating to tone and accent**
T1	Boundary tone of phrase
T2	{Previous, current, next} syllable has a pitch accent
T3	Number of syllables {since, until} pitch accent
T4	Number of accented syllables {since, until} phrase boundary
U	**Features relating to utterance**
U1	Number of syllables in utterance
U2	Number of words in utterance

questions given above, for example, the following is a non-exhaustive list of some of the tasks the system must perform (matched point-for-point). Following each task in parentheses is the module that a system might make use of to perform that task:

1. Letter-to-phoneme conversion (using lexicon and letter-to-sound rules)
2. Associate phonemes with phonetic categories (phoneme set)
3. Associate surface forms of words with their part of speech (POS tagger)
4. Segment utterances into prosodic phrases (phrase-break predictor)
5. Associate syllables with pitch-accents (pitch-accent predictor)

More details will now be given about the modules mentioned in connection with each of these tasks.

Typical modules of a front-end

Lexicon and letter-to-sound rules For languages using scripts with an opaque orthography–pronunciation relationship such as Mandarin and English, look-up in a dictionary of pronunciations is generally performed. The construction of a quality lexicon of a decent size involves considerable work. The Unilex lexicon used for some topline systems consists of 150,000 entries derived from an expert-compiled database (Fitt and Isard, 1999). Even where a quality lexicon is available, the inventory of a living language's words is ever-changing, and new words are constantly appearing. For example, relatively recently coined words like *polyamory* and *cryonaut* and names like *Ahmadinejad* and *Torquay* are absent from the latest version of the Carnegie Mellon University Pronouncing Dictionary used in other experiments presented here (CMU). Even if a lexicon is used, therefore, a system which is to perform TTS conversion on arbitrary input text must supplement its pronunciations with a method for predicting the pronunciations of unseen words. These can be hand-written, but can also be determined automatically by using the spelling–pronunciation correspondences in the lexicon as a learning sample.

Phoneme set A phoneme set that associates phonemes in the lexicon with phonetic categories is another expert-compiled resource. The list used for topline

systems in experiments presented in this thesis includes approximately 90 categories, many of which are compounds such as *voiced consonant*, *labiodental fricative*, *long back vowel*, etc.

Part of speech taggers The part of speech (POS) taggers used for the topline systems of this thesis are statistical taggers trained in a supervised way on c. 1 million words of POS-annotated newstext. Collection of such resources involves obvious effort and expense, and also means that a POS inventory must be designed if none exists for the target language. Note that reference is made to POS indirectly also in question 4 above, because POS tagging is generally used as a first step towards phrase-break prediction. Approximations of full POS tagging can be made, by compiling a list (or lists) of function words, labelling these function words in incoming text as such and labelling everything not in the list simply as a content word. However, a direct comparison of these approaches on the phrase-break prediction task in Section 6.3 of this thesis shows that full POS tagging gives better results.

Phrase-break predictor The phrase-break predictor used in the topline systems of this thesis is trained on data from 40 BBC radio news stories hand-annotated with phrase-breaks by two linguists (SEC/MARSEC: Knowles et al., 1996a,b; Roach et al., 1993). Again, the collection of such resources involves obvious expense.

Pitch-accent and boundary tone predictors The pitch-accent and boundary tone predictors used in the topline systems are trained on news stories from the Boston Radio News Corpus (Ostendorf et al., 1995) which are hand-annotated using ToBI (Silverman et al., 1992). Again, this is an expensive resource to collect, particularly as it is likely that a ToBI-style annotation system for a given target language has not been devised, and that annotating prosodic structure symbolically in the way that ToBI dictates would involve first devising such a system. ToBI has formed the basis for systems in languages other than English (for example, Greek: Arvaniti and Baltazani (2000), Japanese: Maekawa et al. (2002), Russian: Odé (2008)), but the creation of such new systems is obviously an expensive activity.

Supervised Learning for Prediction from Text

The major components of the front-end used in the topline systems in this thesis have now been outlined, along with the resources needed to obtain them. With the exception of the phoneme set, all of these components incorporate modules that are trained using supervised machine learning, but nothing has been said so far about how this training is actually done. The Classification and Regression Tree method will be outlined as a method representative of the sort of supervised learning used to train these modules. Note that tree-based techniques were actually used to train the letter-to-phoneme converter and tone and accent predictors used in the topline systems. Decision trees could potentially also be applied to the other tasks that must be performed: phrase-break prediction (see Section 6.3) and POS tagging (see e.g. the baseline system in Ushioda, 1996).

Decision Trees

The Classification and Regression Tree (CART: Breiman et al., 1993) technique is commonly applied to the task of automatically inducing a set of rules for performing *letter-to-sound conversion* for unseen words. The use of the classification variety of this technique will be described with reference to this task, although the areas of a TTS front-end where it can be applied are numerous. The pronouncing dictionary is treated as a *learning sample*: the goal of the learning algorithm is to capture general patterns of correspondence between the two sides (standard orthography and phonemic transcription) of the learning sample which generalise to unseen words. This allows the resulting predictor to produce likely pronunciations for novel sequences of letters in the language.

The stimulus and response of the learning sample in this case are aligned at the word level, but not at the level on which we wish to model correspondences – that of letters and phonemes. A finer-grained letter–phoneme alignment must therefore be created. This can be done in an automatic or semi-automatic way: Black et al. (1998) give examples of both types of method.

Take the following toy vocabulary:

lisp, flick, file, lice, excite

This toy data-set set gives us five examples of the letter i in context, which our dictionary, once aligned, tells us correspond either to the phoneme /ih/ or /ay/:

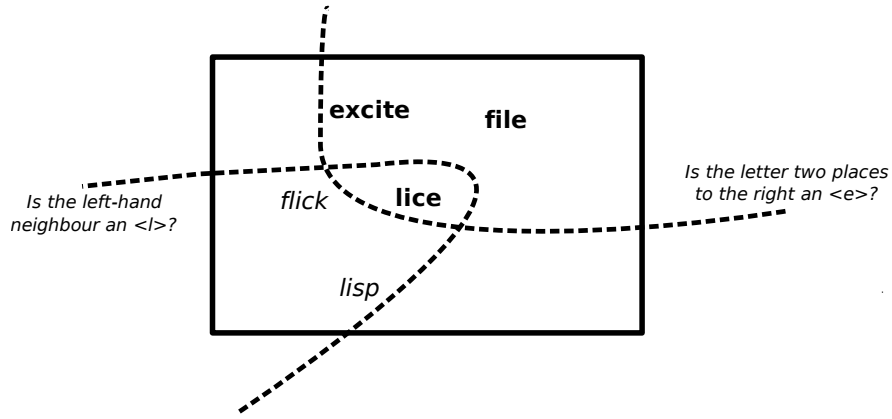


Figure 2.1: *Examples of two questions to split a toy dataset for constructing a letter-to-sound conversion tree. Words in which <i> is labelled /ay/ are shown in bold type; words in which <i> is labelled /ih/ are shown in italics.*

Word	*Realisation of <i>*
lisp	ih
flick	ih
file	ay
lice	ay
excite	ay

From such examples, a model to predict how the letter i is realised in unseen words can be induced, based on the reasonable assumption that a little context of an occurrence of this letter will give useful cues as to its realisation. In the training set, we will specify a set of features that are answers to questions like: *Is the preceding letter an <f>?*, *Is the letter two places to the right a <k>?*. There are 12 such possible questions about four neighbouring letter contexts in our examples. To start with, all examples are placed at the root node of a decision tree. From the set of questions we have defined, we choose one and branch the tree so that it has two new nodes. Examples for which the answer to the question is false end up in one new node, examples where the answer is true end up in the other. The question used to split the examples in this way is chosen so that it puts examples with similar labels together. This is achieved by defining a measure of the impurity of a node. A common measure is the Gini index, although others can be used to similar effect. The Gini impurity of node t, $i(t)$, is defined as follows, where the labels can take m classes, and t_i is the fraction of examples in the node labelled with class i:

$$i(t) = \sum_{i=1}^{m} t_i(1 - t_i) \tag{2.1}$$

This measure is 0 where all items in a node belong to the same class. The root node formed by the toy dataset will here be denoted the 'parent node' t_P (as splitting it will create child nodes). The Gini index for this t_P (where 2 i's are labelled /ih/ and 3 are labelled /ay/) is:

$$i(t_P) = \tfrac{2}{5}(1 - \tfrac{2}{5}) + \tfrac{3}{5}(1 - \tfrac{3}{5}) = 0.48 \tag{2.2}$$

The best question to split a node is the one that minimises: $i(t_L) + i(t_R) - i(t_P)$ where t_L and t_R are the left and right child nodes respectively of t_P. The question shown on the left of Figure 2.1 achieves a score of -0.036; the one on the right, -0.48 – this is in fact the best score associated with any of the 12 questions, and the associated question is therefore chosen. The impurity of the resulting tree is 0, because it is now 0 in either leaf node – all the examples labelled /ih/ are together, and all those labelled /ay/. The tree captures the generalisation that an <i> with an <e> two places to its right is pronounced /ay/, but that otherwise the pronunciation /ih/ is used. The tree will therefore predict the vowels of words like *tin* and *tine* correctly.

Node-splitting is performed recursively on the child-nodes resulting from partitions. Trees with more than two leaves generally result from training sets of a proper size. A fully-grown tree characterises the learning sample well, but may not generalise well to new data. To see why this might be so, let us imagine that the example for the word *lice* is absent from the data set depicted in Figure 2.1. In this case, the two questions shown would split the data in identical ways, and the tree-building algorithm might choose either of them. If the question *Is the left-hand neighbour an <l>?* were chosen, a 2-leaf tree would result, with no impurity in the nodes. However, the feature chosen for the split is spurious, and won't generalise properly to new data like the <i> in *line*. Splitting the data set on spurious features is particularly a problem in the more leafward nodes of large trees, where splits are chosen on the basis of only a few data points.

To remedy this, a process called *cost complexity pruning* is used. The idea is to find a tree which is large enough to fit the learning sample well, but not so big that it overfits it. Fit to the learning sample for a tree T is quantified by $R(T)$, the fraction of training samples that are misclassified by the tree. A tree's size is measured by the number of its leaf nodes, $|T|$. The cost–complexity of a

tree $R_\alpha(T)$ is found as an average of these two terms weighted by a complexity parameter α:

$$R_\alpha(T) = R(T) + \alpha|T| \tag{2.3}$$

The fully grown tree is progressively pruned (leaf nodes are removed). Each successive subtree minimises $R_\alpha(T)$ for progressively larger values of α. A good value of α to select the final pruned tree can be found by cross-validation. To do this, multiple auxiliary trees are built from the learning sample, but data are held out from each to allow the auxiliary trees to be evaluated when pruned with different values of α. The value of α that gives the lowest cross-validated error can be used to select the final pruned subtree trained on the whole learning sample. Alternatively, the largest value of α giving a cross-validated error within 1 standard deviation of the lowest error score can be used.

Note that the decision tree methodology explained here in the context of LTS conversion could be applied to any supervised learning task that needs to be performed in a synthesiser front-end. For example, tree classifiers are built for the task of phrase-break prediction in Chapter 6. The only difference is that instead of a training set which matches features of letters with the appropriate phoneme, features over words – like POS or distance to punctuation – are associated with labels indicating whether a phrase-break is present. Given the data, exactly the same CART-building procedure can be followed.

2.1.2 Synthesiser Back-end: Acoustic Modelling

It has been shown that although the front-ends of synthesis systems can be trained on what is here called *secondary data*, this data is hand-annotated and so incurs considerable expense in the building of a system. The collection of primary data in contrast presents fewer difficulties. As in the case of secondary data, the collection of this data can be considered as a type of annotation: either existing audio is annotated with a plain orthography transcription, or else a text prompt is 'annotated' by a speaker with an appropriate acoustic realisation. Unlike in the case of secondary data, however, this annotation is non-specialist: neither the annotation of audio or the reading-out of text require skills more specialised than literacy. Although the use of specialised facilities (such as purpose-built recording booths) and specialised personnel (such as professional voice talents) would be expected to improve the quality of the data collected, a laptop in a quiet room and non-professional voice talent can give adequate results (Kominek and Black,

2004).

Given the text and speech of a primary corpus and a synthesiser front-end, the training of a back-end can be done automatically in a statistical parametric framework. Linguistic specifications can be made for utterances in the speech database by passing the corresponding text through the front-end, and using the front-end's predictions about how the speaker has realised the text as annotation for the audio signal. The sequences of context-dependent phonemic segments output by the front-end can be aligned in time with the signal using well-established techniques associated with *hidden Markov models* (HMMs), as will be explained here. Also, extensions of HMMs allow models whose parameters can be automatically and robustly estimated from the data to generate appropriate speech for novel pieces of text. Some details of the application of HMMs and variants of them to speech synthesis are given here.

In statistical parametric speech synthesis, the parameters of a statistical model of speech units are inferred automatically from some suitably coded speech corpus. This model can then be used to generate novel speech. It should be noted that this is not the only approach that can be taken to data-driven speech synthesis. In concatenative synthesis, for example, sections of speech are lifted out of the training corpus and joined together to form new utterances. A statistical parametric approach has been adopted in the work presented here because of its robustness and ability to build reasonable systems on small quantities of data. There is also less manual tuning of parameters necessary with this approach compared with unit selection synthesis, which is an attractive quality in work that seeks to lessen the dependence of synthesis system training on manual intervention. It should also be noted that the statistical parametric approach outlined here, based on HMMs, is not the only one that is possible. For example, techniques for acoustic modelling using decision trees (Black, 2006) and neural networks (Karaali et al., 1998; Raghavendra and Prahallad, 2010) have been proposed. The HMM-based approach is presented here as it is the one used for the work presented in this thesis.

Hidden Markov Models for Speech Modelling

Hidden Markov models (HMMs: Gales and Young, 2007) are the most widely-used statistical models for mapping between acoustic and phonetic representations of speech, both in speech recognition and speech synthesis. HMMs of speech operate on some representation of speech that is more compact and smoothly-evolving

than the original waveform. The requirement of this representation for speech recognition is that it must allow a system to discriminate between phonetically different segments. An additional requirement of this representation in speech synthesis is that it contains enough information that a waveform acceptable to listeners can be reconstructed from it. In either case, utterances are represented as a sequence of vectors representing the acoustics of a short segment of speech. This sequence will be called the *observation sequence*, and denoted $\boldsymbol{O} = (\boldsymbol{o}_1, \boldsymbol{o}_2, ...\boldsymbol{o}_T)$ where T is the length of the sequence and $\boldsymbol{o}_t$ is a vector of values (or *frame of speech*) observed at time t.

In HMM-based modelling of speech, the observed sequences are considered to have been emitted by a statistical model. N states are defined for a model; when an observation sequence of length T (e.g. a piece of speech) is generated by the model, the model passes through a sequence of these states. Such a state sequence will be denoted $\boldsymbol{Q} = (q(1), q(2) \dots q(T))$ – where $q(t)$ is the identity of the state which the model is in at time t. A transition matrix $\boldsymbol{A}$ is defined for a model (which will be called $\boldsymbol{\lambda}$), where a_{ij} is a *transition probability*: the probability that the model will make the transition to state j given that it is in state i. The placement of 0's in this matrix determines the model topology, and thus the allowable state sequences for a model. A common topology for speech recognition and synthesis can be seen in the following transition matrix:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The model is forced to start in the first state, then progress from left to right with no skips to the final state, where it ends. The process of an N-state HMM emitting an observed sequence of T frames can be thought of as a path through an $N \times T$ lattice. The model starts in the top left corner of this lattice (in state 1 and time 1), and takes a path to the bottom right corner (state N at time T). At each time frame from 2 till $T-2$, the model has two possibilities: remaining in its current state (a rightward horizontal move in the lattice) or moving to the next state (a diagonal downwards move). Every path through the lattice is a different state sequence, and – ignoring for a moment the probability of the model emitting

the observations – a probability can be given for each one using $\boldsymbol{A}$. Note that many state sequences can generate the same observation sequence, but different sequences can have different probabilities attached to them. A probability of some state-sequence $\boldsymbol{Q}$ given a model $\boldsymbol{\lambda}$ is computed as the product of the transition probabilities between each pair of consecutive states:

$$\begin{aligned} P(\boldsymbol{Q}|\boldsymbol{\lambda}) &= a_{q(1)q(2)} a_{q(2)q(3)} \dots a_{q(T-1)q(T)} \\ &= \prod_{t=2}^{T} a_{q(t-1)q(t)} \end{aligned} \tag{2.4}$$

As it passes through a sequence of states $\boldsymbol{Q}$, the model emits observations stochastically. For modelling continuously-varying values like speech features, continuous emission probability density functions are usually used to determine the probability of a given state generating a given observation vector. The probability density function for state j will here be denoted $b_j(.)$. In the present work, Gaussian density functions will be used: each $b_j(.)$ has a vector of mean and a matrix of covariance parameters that can be used to determine the probability of an observed vector having been emitted by state j. These are denoted here as $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$.

Given a state sequence $\boldsymbol{Q}$ and a model $\boldsymbol{\lambda}$, the probability of the model emitting the observations is computed as the product of state emission probabilities for each state in the sequence:

$$\begin{aligned} P(\boldsymbol{O}|\boldsymbol{Q},\boldsymbol{\lambda}) &= b_{q(1)}(\boldsymbol{o}_1) b_{q(2)}(\boldsymbol{o}_2) \dots b_{q(T)}(\boldsymbol{o}_T) \\ &= \prod_{t=1}^{T} b_{q(t)}(\boldsymbol{o}_t) \end{aligned} \tag{2.5}$$

Note that any state can emit any observation; but some will do so with higher probability than others. By the same token, any of the many possible state sequences can emit a given observation sequence, but some will do so with higher probability than others.

The probability $P(\boldsymbol{O}|\boldsymbol{\lambda})$ of a sequence of observations given a model is computed by summing the probability of the observation given the model by all possible state sequences, weighted by the probability of those state sequences:

$$P(\boldsymbol{O}|\boldsymbol{\lambda}) = \sum_{\forall \boldsymbol{Q}} P(\boldsymbol{O}|\boldsymbol{Q},\boldsymbol{\lambda}) P(\boldsymbol{Q}|\boldsymbol{\lambda}) \tag{2.6}$$

Computing this by explicitly enumerating all state sequences that could account for the possible alignments of an N-state model with an observed sequence of T frames would be computationally infeasible. However, efficient methods for making this computation exist, and they will be introduced in the course of discussion on the estimation of HMM parameters.

HMM Training

Model Update Given an observation sequence (e.g. a sentence of speech on which we would like to train a synthesiser), how can we adjust the parameters of an HMM so that the model better describes that sequence? This can be done using an iterative procedure known as the Baum-Welch algorithm. This algorithm takes a (possibly poor) set of model parameters $\boldsymbol{\lambda}$, and estimates a new model $\hat{\boldsymbol{\lambda}}$ so that $P(\boldsymbol{O}|\hat{\boldsymbol{\lambda}}) \geq P(\boldsymbol{O}|\boldsymbol{\lambda})$.

In Baum-Welch training, the new model's parameters ($\hat{\boldsymbol{\lambda}}$) are estimated as weighted averages of values in the observation sequence. The weight used to update state j's parameters with frame $\boldsymbol{o}_t$ of the training data is the posterior probability of state j generating that frame under the old model parameters $\boldsymbol{\lambda}$. This posterior will be denoted $\gamma_j(t)$. For example, the formula for updating the mean vector of state j, $\boldsymbol{\mu}_j$, is:

$$\boldsymbol{\mu}_j = \frac{\sum_{t=1}^{T} \gamma_j(t)\boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_j(t)} \tag{2.7}$$

The update of state covariance matrices and transition probabilities is based on the same principle of expectation (calculation of posteriors) and maximisation of likelihood using those posteriors. (Equations for the covariance and transition probability updates can be found in e.g. Gales and Young (2007).)

Forward and Backward Probabilities How are the posterior probabilities used in Equation 2.7 obtained? A brute-force method would involve enumerating all possible state sequences explicitly. Due to the great number of possible state sequences, this is infeasible. A more efficient way of computing $\gamma_j(t)$ is based on the fact that this value can be represented as the normalised product of two other variables. The first is the probability of the model generating ($\boldsymbol{o}_1 ... \boldsymbol{o}_t$) and being in state j at time t; this is denoted $\alpha_j(t)$ and called the *forward probability*. The second is the probability of the model being in state j at time t and generating ($\boldsymbol{o}_{t+1} ... \boldsymbol{o}_T$); this is denoted $\beta_j(t)$ and called the *backward probability*. These values

account for all partial paths into the node of the state–time lattice representing state j at time t and emitting $\boldsymbol{o}_t$, and all partial paths out of that node of the lattice until the end of the sequence. Values of $\alpha_j(t)$ can be found efficiently, because once the values for all states at time 1 have been set, the following recursion can be used to compute values for each state at later times:

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij}\right] b_j(\boldsymbol{o}_t) \tag{2.8}$$

The recursion explains the efficiency of the procedure: indirectly or directly, calculation of values at time t reuses calculations made at all previous times. The values $\beta_j(t)$ for all states j at all times t are computed with a similar recursion, but working backwards from time T to time 1.

With $\alpha_j(t)$ and $\beta_j(t)$ computed in this way at every time t for every state j, the posterior probability $\gamma_j(t)$ of being in state j at time t given the observation sequence and the model can be computed as:

$$\gamma_j(t) = \frac{\alpha_j(t)\beta_j(t)}{\sum_{i=1}^{N} \alpha_i(t)\beta_i(t)} \tag{2.9}$$

The denominator normalises $\gamma_j(t)$ into a probability so that $\gamma_j(t)$ for all states j at time point t sums to one.

Given this tractable way of computing posteriors, model parameters can be adjusted as already mentioned. Iterating between these two steps – calculation of posteriors and adjustment of model parameters – yields a gradual refinement of model parameters. The two steps are iterated until convergence. Nothing has been said here so far about finding parameters for an initial model. Converging on a good set of model parameters depends in part on how this initialisation is done, as the algorithm only guarantees convergence to a locally maximum likelihood. All the experiments presented in the present work begin with a so-called *flat start* where the parameters of emission densities of a model are all set to the global mean and variance of the data on which it is to be trained.

Forced Alignment

The sequence $(\gamma_j(1), \gamma_j(2), ...\gamma_j(T))$ can be thought of as a soft alignment of state j with a sequence of training data. A hard segmentation – where the probability of each state j emitting each observation $\boldsymbol{o}_t$ is either 1 or 0 – can be obtained using a recursion similar to the one given in Equation 2.8, except that the summation

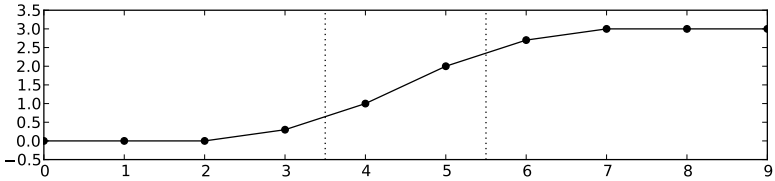
operation is replaced with taking the maximum. From this recursion, the single most likely state sequence of the model for some observation can be found. This is useful for aligning linguistic events such as words or phonemes (which are associated with states of the model) with an acoustic signal in a so-called *forced alignment*. The algorithm used to determine this optimal state sequence is called the Viterbi algorithm.

Duration Modelling

Using a transition matrix to model temporal structure is computationally efficient, but if this is done the probability of occupying a state is modelled as decreasing exponentially with time. This does not provide a good representation of the duration probability distibutions of real speech segments. A better (but still not ideal) representation is provided by modelling state duration explicity with a Gaussian distribution (Yoshimura et al., 1998). Forward, backward and Viterbi recursions are more computationally expensive when explicit durations are used as the HMMs no longer have the Markov property (i.e. that what they will do next depends only on which state they are in), and are more properly called *hidden semi-Markov models* (HSMMs). Explicit duration modelling was used for all synthesis models built for the experiments presented in this thesis.

Speech Parameter Generation

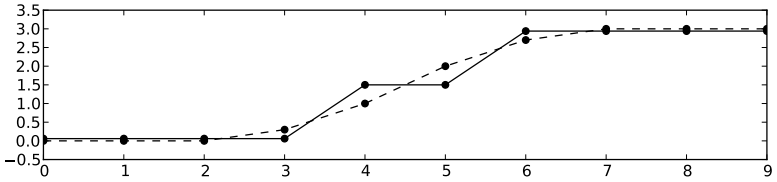
It has been shown how the parameters of an HMM can be estimated from an observation of speech. A question that must be answered for TTS is: given these learned parameters, how can trajectories of speech acoustic features be generated, from which in turn a waveform can be resynthesised? As a very simple example, imagine that the following is a sequence of values of some acoustic feature naturally occurring in speech, plotted against time:



This might represent any useful acoustic measurement such as F_0 or energy, in arbitrary units. For the sake of simplicity in the following explanation, 1-dimensional vectors of observations are used. Also, we here assume deterministic assignment of observations to three states, whose boundaries are shown by dotted

lines. There is therefore nothing *hidden* about the model in this example; we compute means and variances of single Gaussians for the segments directly: the means and variances of the emission distributions for the three states are 0.06, 1.5, 2.94 and 0.0144, 0.25, 0.0144, respectively.

Given the model represented by these six parameters computed from the data, what is the most likely sequence of observations? The state durations observed in the training data will be used in resynthesis for this toy example. As the model represents the speech using Gaussian distributions, the most likely value for every frame of a state will be the state mean. This resynthesised trajectory is plotted here with a solid line – the dashed line shows the natural trajectory for comparison:



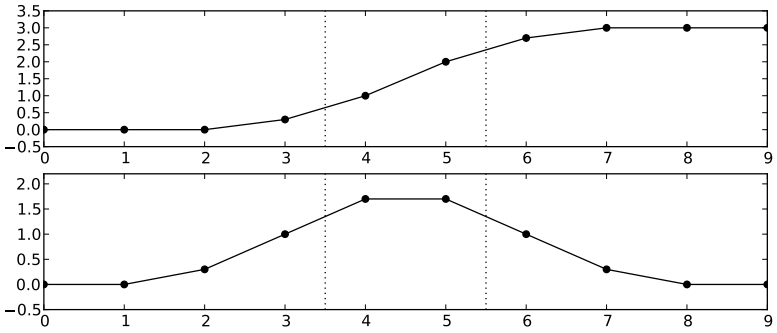
This is obviously a bad reconstruction of the data, as the acoustic values seen in speech tend to evolve much more steadily than the 'stepped' trajectory here. This stepped quality is due to the Markov property of the model, where observations are conditionally independent given the state. To circumvent this conditional independence, observed values are commonly supplemented with dynamic values, specifying not the current value of the parameter, but the rate at which that value is changing over a small window. The dynamic value at frame t will here be denoted $\Delta \boldsymbol{o}_t^{\boldsymbol{s}}$ and calculated very simply for the purposes of this example as the difference between left and right static values, $\boldsymbol{o}_{t+1}^{\boldsymbol{s}}$ and $\boldsymbol{o}_{t-1}^{\boldsymbol{s}}$.[2] The combined static and dynamic observation ($\boldsymbol{o}_t$) can therefore be obtained by linearly transforming static parameters:[3]

$$\boldsymbol{o}_t = \begin{pmatrix} \boldsymbol{o}_t^{\boldsymbol{s}} \\ \Delta \boldsymbol{o}_t^{\boldsymbol{s}} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{o}_{t-1}^{\boldsymbol{s}} \\ \boldsymbol{o}_t^{\boldsymbol{s}} \\ \boldsymbol{o}_{t+1}^{\boldsymbol{s}} \end{pmatrix} \tag{2.10}$$

The combined static and dynamic features $\boldsymbol{O}$ are shown here in the upper and lower panels respectively:

[2]The following explanation quite closely follows that given by Gales and Young (2007, §3.6)

[3]Note that $D \times D$ identity matrices should replace the 1's here for D-dimensional vectors of observations, and zero matrices should replace the 0's. From now on, 1-dimensional static observations will be assumed in this discussion.



These combined static and dynamic features will be modelled here using a 2-dimensional Gaussian with diagonal covariance. Assuming the state sequence $\boldsymbol{Q}$ is known (q(1)...q(T)), a single $2DT$-dimensional Gaussian distribution for the whole sequence can be composed from the state-level distributions, where T is the length of the state sequence, D is the dimensionality of the observations and the 2 accounts for the fact that the model has parameters for both static and dynamic features. For the 1-dimensional sequence of 10 elements plotted above, this gives a 20-dimensional distribution. Its mean vector and covariance matrix will be denoted $\boldsymbol{\mu_Q}$ and $\boldsymbol{\Sigma_Q}$ respectively, and they are composed as follows:

$$\boldsymbol{\mu_Q} = \begin{pmatrix} \boldsymbol{\mu}_{q(1)} \\ \vdots \\ \boldsymbol{\mu}_{q(T)} \end{pmatrix} \tag{2.11}$$

$$\boldsymbol{\Sigma_Q} = \begin{pmatrix} \boldsymbol{\Sigma}_{q(1)} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{\Sigma}_{q(T)} \end{pmatrix} \tag{2.12}$$

The most likely sample from this Gaussian is the same as its mean vector: a sequence of 20 values in which values corresponding to static and dynamic values are interleaved. Both the values for static and dynamic features will have the type of 'stepped' trajectory already shown.

Using the principle shown in Equation 2.10, a $2T \times T$ matrix $\boldsymbol{W}$ can be devised that expresses the combined static–dynamic features of augmented observation $\boldsymbol{O}$ (consisting of alternating static and dynamic values) as a linear transformation of the static observation (denoted $\boldsymbol{O^s}$):

$$\boldsymbol{O} = \boldsymbol{W}\boldsymbol{O^s} \tag{2.13}$$

For the toy example already shown, this is as follows:

$$\begin{pmatrix} 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.30 \\ 0.30 \\ 1.00 \\ 1.00 \\ 1.70 \\ 2.00 \\ 1.70 \\ 2.70 \\ 1.00 \\ 3.00 \\ 0.30 \\ 3.00 \\ 0.00 \\ 3.00 \\ -3.00 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 0.00 \\ 0.00 \\ 0.00 \\ 0.30 \\ 1.00 \\ 2.00 \\ 2.70 \\ 3.00 \\ 3.00 \\ 3.00 \end{pmatrix} \tag{2.14}$$

In the same way as it can relate static acoustic features to the corresponding static–dynamic features, this matrix $\boldsymbol{W}$ can relate the parameters of our 20-dimensional utterance-level Gaussian ($\boldsymbol{\mu_Q}$ and $\boldsymbol{\Sigma_Q}$) with the parameters of a 10-dimensional Gaussian distribution, also over the whole utterance. The mean vector and covariance matrix of this second distribution will be denoted $\boldsymbol{\mu_Q^s}$ and $\boldsymbol{\Sigma_Q^s}$; this is a 10-dimensional distribution over static features, but one which properly incorporates the constraints of the dynamic features. The following relationships exist between the parameters of the standard HMM utterance-level distribution and the parameters of this distribution over static parameters:

$$\boldsymbol{\Sigma_Q^s}^{-1} = \boldsymbol{W}^T \boldsymbol{\Sigma_Q}^{-1} \boldsymbol{W} \tag{2.15}$$

$$\boldsymbol{\Sigma_Q^s}^{-1} \boldsymbol{\mu_Q^s} = \boldsymbol{W}^T \boldsymbol{\Sigma_Q}^{-1} \boldsymbol{\mu_Q} \tag{2.16}$$

Note that the covariance of the standard HMM utterance-level distribution $\boldsymbol{\Sigma_Q}$ is block diagonal (in our case, diagonal, as we are using diagonal covariance for state distributions). The covariance matrix of this distribution over the utterance's static features – $\boldsymbol{\Sigma_Q^s}$ – on the other hand, is not diagonal or block diagonal because of the multiplication with $\boldsymbol{W}$, which is not block diagonal. The assumptions of conditional independence that are implicit in $\boldsymbol{\Sigma_Q}$ are not implicit in $\boldsymbol{\Sigma_Q^s}$. But $\boldsymbol{\Sigma_Q^s}$ and $\boldsymbol{\mu_Q^s}$ represent a Gaussian distribution, so the most likely sample from it will simply be the mean, $\boldsymbol{\mu_Q^s}$. This must be found to determine a static feature trajectory that respects both static and dynamic parameters of the original model. Equation 2.16 can be rearranged to:

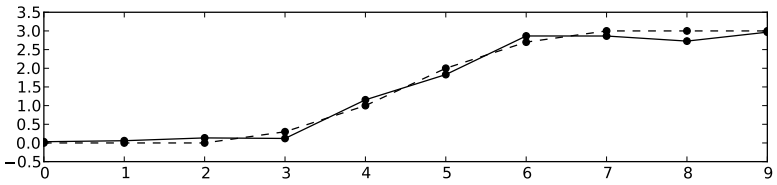
$$\boldsymbol{\mu_Q^s} = \boldsymbol{\Sigma_Q^s} \boldsymbol{W}^T \boldsymbol{\Sigma_Q}^{-1} \boldsymbol{\mu_Q} \tag{2.17}$$

Plugging Equation 2.15 in gives:

$$\boldsymbol{\mu}_{\boldsymbol{Q}}^{\boldsymbol{s}} = (\boldsymbol{W}^T \boldsymbol{\Sigma}_{\boldsymbol{Q}}^{-1} \boldsymbol{W})^{-1} \boldsymbol{W}^T \boldsymbol{\Sigma}_{\boldsymbol{Q}}^{-1} \boldsymbol{\mu}_{\boldsymbol{Q}} \tag{2.18}$$

All terms on the right hand side are known, both $\boldsymbol{W}$ and the utterance distribution parameters ($\boldsymbol{\mu}_{\boldsymbol{Q}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{Q}}$) whose elements were estimated from data.

Working through this parameter generation algorithm using the toy example already shown yields $\boldsymbol{\mu}_{\boldsymbol{Q}}^{\boldsymbol{s}}$ whose elements are plotted here:



As before, the values of the originally observed sequence are shown with dashed lines for comparison. Note that the generated trajectory is smooth like the original from timepoints 3–6, unlike the naive 'stepped' trajectory shown above. However, the inflection points at time 3 and 6 are sharper for the regenerated trajectory than the original. This is because only static and delta (speed) statistics are respected in this example; in practice the principle is extended to also respect delta-delta (acceleration) statistics to avoid such unnaturally sudden accelerations and decelerations.

In this toy example, the state sequence was treated as visible. This is the approach taken for all speech generated for the experiments presented in this thesis, where the mean values of explicit state duration distributions are used to provide a state sequence (see Section 2.1.2). Note however that parameter generation algorithms that jointly optimise state sequence and state emissions have also been proposed (Tokuda et al., 2000).

Linguistic Context in Acoustic Modelling

Some of the basic techniques of acoustic modelling have been sketched, but as yet no mention has been made of the way that the rich contexts predicted for segments by the front-end as described in Section 2.1.1 are handled in acoustic model estimation. There it was mentioned that in the topline systems presented in this thesis, each segment associated with a phoneme is characterised by almost 2,000 contextual binary features. This context-dependency results in a vast number of possible units: nearly every unit in the training corpus will be of its own unique type and at synthesis time, the majority of models that are required to be synthesised will be of unseen types. Therefore, a method is needed to map from

the vast set of possible context-dependent models that could be predicted by the front-end to a set that is small enough that there are sufficient data to estimate model parameters during training, and general enough to represent unseen units at synthesis time. The technique employed for this purpose in all systems built for the experiments described in this thesis is decision-tree based clustering (Odell, 1995).

Acoustic models which are to be clustered consist of distributions over acoustic values associated with a set of linguistic–prosodic attribute values for c unique contexts. These distributions represent the data in the training set, and might generally be thought of as a synthesis model, but in the present context will be termed the *learning sample* for a decision tree. This learning sample, which will here be denoted $\boldsymbol{Z}$, is of the form $\{(\boldsymbol{x}_1, \gamma_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \ldots (\boldsymbol{x}_c, \gamma_c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)\}$, where $\boldsymbol{x}_j$ is a q-dimensional vector of binary values for the j^{th} context, whose k^{th} element specifies whether the k^{th} linguistic–prosodic attribute is true or false of context j. $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are the (possibly poorly estimated) parameters of distributions over acoustic values associated with the j^{th} context. γ_j is the quantity of frames of the training corpus attributed to context (context-dependent state) j in a forward–backward alignment of the model with the observations:

$$\gamma_j = \sum_{\forall t} \gamma_j(t) \tag{2.19}$$

where $\gamma_j(t)$ is defined in Equation 2.9.

As with the CART example given in Section 2.1.1, the aim of tree-building is to build a binary branching tree, down which linguistic units can be passed appropriately by answering questions about their linguistic features (values of $\boldsymbol{x}_j$). The model parameters in each of the leaves of this tree will be tied; these tied states $\boldsymbol{S}$ are then the model which is considered as having generated the acoustic observations.

The criterion by which splits of the data are chosen to build the tree is based on an approximation of the log-likelihood of the data given the tied model. The method presented in Odell (1995, pp. 37–38) works by assuming that the alignment of states with observations and model transition probabilities are unaffected by the clustering, and that the log-likelihood of observation $\boldsymbol{o}_t$ being generated from tied model $\boldsymbol{S}$ can be computed as the average of the log-likelihoods of it being generated by each tied state s in $\boldsymbol{S}$ weighted by the respective posterior $\gamma_s(t)$. Given these assumptions, an approximation of the true likelihood can be derived that relies only on knowing the covariance matrices and state occupancies of each

tied state s. These in turn can be calculated from the means, covariances and state occupancies of the unclustered model $\boldsymbol{Z}$. The best split of a node during tree building is the one that will increase this approximation of likelihood the most.

As with the classification tree example given in Section 2.1.1, the goodness of the model as measured by this objective function based on the learning sample will never worsen as tree size increases. As in Section 2.1.1, over-large trees will generalise badly to new data. Additionally in the case of state-tying, each node of the tree needs to represent enough frames of acoustic data that the parameters of its tied states can be well estimated after tree-building. Unlike in Section 2.1.1 where pruning an over-large tree structure was performed to obtain a suitable tree, early termination of tree-growing is used for the state-tying trees built for the work presented in this thesis. For these systems, the Minimum Description Length approach described in Shinoda and Watanabe (2000) is used to determine when tree-building should cease. Essentially, finding the description length of a model involves combining the likelihood approximation with a term that penalises large model size. When description length ceases to decrease, splitting is terminated.

2.1.3 Training Recipes Used

Details are given here of the actual recipes used to build the voices in the course of the work presented in this thesis. Two different training recipes were used, which will be called HTS-2005 and HTS-2010. The former is the procedure used for the HTS group's entry in the Blizzard Challenge 2005 (Zen et al., 2007), while the latter training procedure is similar (although not identical) to that used for the CSTR/EMIME entry to the Blizzard Challenge in 2010 (Yamagishi and Watts, 2010). Both are speaker-dependent training recipes which are very similar in outline, although HTS-2010 introduces numerous small improvements. The annotation of training data is identical for both recipes, and much of the procedure for acoustic model building is also shared. What is common to both recipes will first be described, followed by details of how the two recipes differ.

Annotation Recipe

The annotation for standard topline English voices is obtained in a similar way for all voices, using version 2.0 of the Festival Speech Synthesis System (Black et al., 1999), and in particular the voice-building tools associated with its Multisyn module (Clark et al., 2007).

A **phonemic transcription** is first produced from a plain orthography transcription of the data by performing lexical look-up from a pronouncing dictionary. For the experiments reported in this thesis, either the Carnegie Mellon University Pronouncing Dictionary (CMU) or the Unilex Received Pronunciation Lexicon (Fitt and Isard, 1999) was used.

Transcriptions of out-of-lexicon words in a training corpus are produced manually and the lexicon augmented. This initial transcription is then refined by forced alignment with the audio, allowing reduction of vowels and the insertion of pauses between words where supported by the audio data. Syllabification is taken from the lexicon and incorporated into the annotation, as are lexical stress and word boundaries.

Part of speech tags are assigned by the pre-trained model distributed with Festival, which had been trained on the Wall Street Journal data of the Penn Treebank (Marcus et al., 1993), using n-grams to assign tags to word sequences probabilistically.

Phrase breaks are assigned on the basis of Festival's predictions from text. Predictions are provided by the pre-trained probabilistic model distributed with Festival, trained on data from the the Lancaster/IBM Spoken English Corpus (SEC: Knowles et al., 1996a,b) and its machine-readable extension, MARSEC (Roach et al., 1993). The model uses n-grams over POS sequences to assign phrase breaks (Taylor and Black, 1998).

Pitch accents and boundary tones are assigned on the basis of Festival's predictions from text, provided by the pre-trained CART models distributed with Festival. These two CARTs had been trained on data from speaker *f2b* of the Boston University Radio News Corpus (Ostendorf et al., 1995), which is annotated in ToBI notation.

Acoustic Model Building Recipe

Speech units corresponding to phonemes are modelled with HMMs with 5 emitting states in the left-to-right topology described earlier. Single mixture component Gaussian distributions with diagonal covariance matrices are used to model state output probabilities. Model training begins with the estimation of monophone models (models of phones where all features except those regarding the phoneme's identity are ignored). These are then cloned and used as the basis for the full-context models. They are trained slightly to reflect their rich contexts before decision-tree based context clustering is applied. Clustering is done sepa-

rately for each state of the spectral envelope, F_0 and aperiodicity (see below) parts of the model and a single tree for duration is made for all five states, resulting in 16 trees for each voice built. The separate clustering of e.g. spectral envelope and F_0 parameters means that these can be modelled using different context features. This is important, as different aspects of context obviously affect spectral quality from those that affect F_0. The Minimum Description Length approach described above is used to determine suitable tree sizes. The parameters of states pooled in the leaf nodes of trees are then tied and re-estimated. This procedure can be repeated: parameters are untied and re-estimated a little, clustered, tied and re-estimated. For the HTS-2005 procedure, the number of iterations of this loop is fixed at 2. For HTS-2010, it is variable, and is specified at each point in the thesis where use of HTS-2010 is mentioned.

In the HTS-2005 procedure, 16kHz waveforms are parameterised as sequences of vectors consisting of of 40 Mel-cepstral coefficients, log F_0 and the energy of aperiodic components in 5 frequency bands, as well as the dynamic and acceleration features derived from all of these. Vectors are extracted from 25 ms windows of speech with a 5 ms frame-shift. F_0 is extracted robustly by using a committee of pitch trackers that vote on the value of each frame. Spectral analysis is performed using the high quality vocoder STRAIGHT (Kawahara et al., 1999), after which the STRAIGHT spectra are converted to the Mel-cepstral coefficients already mentioned.

For the HTS-2010 systems, the static acoustic features described in Yamagishi and Watts (2010) were used: Bark cepstrum instead of Mel cepstrum, auditory-scale motivated frequency-bands for aperiodicity features, and Mel F_0 instead of log F_0. In other respects, features were the same as for HTS-2005. That is, 16kHz waveforms were used and 40 cepstral coefficients derived from STRAIGHT spectra.

2.2 Types of Alternative Approach

Aspects of the conventional approach which can make it difficult and time-consuming to apply to a new target language have been mentioned in the course of its description. The principal difficulty is that collecting the resources necessary for building the synthesiser's front-end is expensive and hard to do. Previous work has addressed this problem, but mainly with the strategy of providing frameworks to ease the collection of the necessary resources. For example, Kominek

et al. (2007) describe SPICE, a project designed to 'reduce the difficulty of building and deploying speech technology systems' in any language. This is done by integrating tasks such as phoneme set definition and lexicon building in a user-friendly development environment. The related work of Kominek (2009) explores ways of optimally using non-expert but native speaker input for the iterative construction of a phoneme set. A related approach is taken in the work of the Local Language Speech Technology Initiative: a project with the goal of enabling people in the developing world to access information using voice technology (Tucker and Shalonova, 2005). The focus of the project is to provide:

> [...] training and on-going support to enable a non-expert to produce a good quality system in a reasonable time frame (Tucker and Shalonova, 2005).

Scarcity of acoustic resources is not considered in this thesis to be the main bottleneck to system development. As already explained, the 'labelling' of a text utterance with a waveform by a speaker is a very naive sort of annotation compared with that required to build a full front-end in the conventional supervised manner. It should be noted, however, that work has been done which could address the lack of a primary corpus. For example, Anumanchipalli and Black (2010) explore model adaptation as a means of reusing acoustic data between languages for TTS training. Latorre et al. (2006) and Zen (2010) present similar ideas, although the aim there is to enable polyglot synthesis rather than synthesis in scarcely-resourced languages. Another approach is the wholesale reuse of systems or system modules between languages, in what is usually presented as a stop-gap or prototyping approach (examples for several language pairs are Black and Lenzo (2004): Spanish and Basque; Dijkstra and Pols (2004): Dutch and Frisian; Somers et al. (2006): German and Somali).

The approach taken here, on the other hand, assumes that the collection of a reasonably sized primary corpus presents little difficulty. Even easier to collect are what are here called *tertiary corpora*: unannotated speech or text (i.e. not aligned with anything, even at the utterance level). Given this easy-to-collect data, the approach pursued in this thesis is to attempt to learn a synthesis model with minimal supervision. Such an approach depends on the assumption that similar structures to those which are labelled explicitly in a secondary corpus are inherent in language data. Three broad classes of approach that fit this description are here described: a *naive approach* that attempts to use textual units directly as speech modelling units, a *speech-driven approach* where structure is induced in

an unsupervised way from audio, and a *text-driven approach* where structure is induced from text. The conventional approach to HMM-based speech synthesis that has already been presented in this chapter will be briefly reviewed for ease of comparison with these three unsupervised approaches. This section will be concluded with an overview of the approach taken in the work presented in the rest of this thesis.

2.2.1 Conventional Approach

The conventional approach already described can be summarised as follows:

1. A front-end is trained in a supervised fashion: modules are trained to predict linguistically-motivated features which prior knowledge suggests will have an influence on speech segments' characteristics.

2. The predictions of this front-end are used to label the training data from its text part only.

3. The predictions obtained in the previous step are refined by means of forced alignment with the acoustic part of the database. For example, insertion of silent segments and reduction of vowels might be incorporated into the linguistic specification where evidence from the audio supports this.

4. The combinations of features assigned by the front-end that delimit the members of each modelled unit are decided during decision-tree building for state-tying. Features are selected and combinations of them are found in a data-driven fashion using distributions over the audio data. Features that do not affect the acoustic representation will not be used. Ideally, our representation of the signal will align closely enough with a perceptual representation that only factors relating to perceptually relevant differences will be used in the model.

A key point of the conventional approach just described is the following: a great number of features are predicted from text; linguistic knowledge suggests that they will be useful in acoustic (acoustic–perceptual) modelling, but if some are not it does not matter, as the 'chaff' will be 'winnowed' from the representation produced by the front-end during the building of acoustic models.

2.2.2 Naive Approach

The simplest approach imaginable is to naively use orthographic units in place of linguistically motivated ones. For example, letters might be used to stand in for phonemes, and surface word-forms for part of speech tags. The standard state-tying mechanism which the synthesis system uses for handling context is required to do extra work – handling letter-to-sound correspondences, for example, in addition to coarticulation effects. Such an approach does much of the work of synthesis in what is step 4 of the conventional approach; its success is dependent in particular on the ability of decision tree-based clustering to 'winnow away' irrelevant and useless features, and to find useful combinations of the relevant ones.

The sort of units that can be used in this approach depend on the language and its writing system. This approach has obvious application to languages that use alphabetic scripts, and some form of it might be made to work for languages using any type of script except an ideographic one (see Section 2.3). Where word boundaries are marked by a script, orthographic words can provide modelling contexts. The naive approach has been used in speech recognition as a means of managing without a phoneme set (e.g. Killer et al., 2003) and has also been attempted for speech synthesis, using both unit selection (Black and Font Llitjos, 2002; Anumanchipalli et al., 2008; Aylett et al., 2009) and HMM-based synthesis (Aylett et al., 2009). Kominek (2009) uses plain orthographic units as the starting point for the creation of a phoneset via the iterative application of split and merge operations. Whether such methods can be applied at levels other than the subword unit is not clear from previous work.

2.2.3 Speech-Driven Approach

One set of approaches to the unsupervised acquisition of a front-end is to place emphasis on step 3 of the conventional approach: to assign more (or all) linguistic features on the basis of extraction from acoustics rather than prediction from text. In the most extreme case, the audio part of the database is examined in isolation from any transcription, and language-independent constraints are used to segment audio into e.g. segment (Černocký, 1998; Aylett et al., 2009) or syllable-sized (Mermelstein, 1975; Xie and Niyogi, 2006) units. Units are then clustered in an unsupervised manner into e.g. phoneme-like classes (Černocký, 1998; Aylett et al., 2009). Similar approaches have been proposed also for levels

of analysis above the word such as intonation events (Hirst and Espesser, 1993; Hirst, 2005).

An obvious weakness of this type of approach for TTS is that it only produces an inventory of units, and no way of predicting these from text. A front-end must additionally be constructed to do this prediction. Furthermore, in discarding transcription when deriving an inventory of units, cues are discarded that could prove valuable in producing a useful inventory. Aylett et al. (2009) argue that acoustic parameterisations typically used for this work do not relate closely enough to a perceptually-relevant representation of speech for this type of approach to be successful.

2.2.4 Text-Driven Approach

Much work has been done on the unsupervised induction of linguistic classes and structures from data, for example, morphological segmentation (Creutz and Lagus, 2007), POS-like categories (Christodoulopoulos et al., 2010), and parse-trees (D'Ulizia et al., 2011). This work suggests a second possible type of approach to TTS where a conventional front-end is missing: induce some structure from the text part of the corpus, without reference to the acoustics, and then train a system in the conventional way, but using this induced representation in place of the conventional linguistic representation.

A very appealing characteristic of this approach is that it enables us to harness large untagged text corpora (*tertiary corpora*). The problem with following a strictly text-based approach is that (often categorical) representations are found without regard to the acoustic data available in the primary corpus. Although decision-tree-based context clustering allows us to some extent to select features from a large potential set that are relevant for the acoustic modelling task, it is desirable to delay the sort of hard decisions that are inherent in categorical representations until acoustic features have been observed.

2.2.5 Approach Taken in this Thesis

The approach taken here is to start with a naive system where textual units are used as acoustic modelling units. Such systems are evaluated against topline systems in Chapter 3. The remainder of the thesis seeks to strengthen this naive approach with representations of textual units that are derived using elements of the text-driven approach. Importantly, the representations used are continu-

ous rather than categorical. These continuous features leave hard categorisation of textual objects open to later stages, where for example, decision tree-based clustering can find divisions of the space of objects that are pertinent for TTS.

2.3 Naivety, language independence, and language typology

As explained in the previous paragraph, the approach on which the techniques developed and tested in this thesis are based is a *linguistically naive* one. Bender (2009) warns against making claims for the language-independence of NLP systems without testing on a range of typologically varied languages, observing that linguistically naive systems can make implicit assumptions about language structure and overfit the languages on which they are developed. In this thesis, techniques are developed mainly in one language and tested on a further two in Chapter 8: hardly a large and varied sample of the world's languages. A few words should be said, therefore, about the extent to which the techniques developed are anticipated to be more widely applicable.

Basic assumptions made about the target language in this thesis are that it uses an alphabetic script and that word boundaries are marked orthographically. For the Local Language Speech Technology Initiative project (see Section 2.2), a publically-accessible database containing script and language information for 107 languages especially relevant for TTS purposes was compiled (Shalonova and Tucker, 2003; Tucker and Shalonova, 2005).[4] The criterion for inclusion in the database is that 'official newspapers' are published in the language in question (Shalonova and Tucker, 2003). Table 2.2 shows some data extracted from this resource.[5] From this, it can be seen that over 65% of the languages in this database meet the assumptions mentioned, of alphabetic script and orthographically marked words. Although not tested here, it is supposed that letter-based synthesis would also be feasible with alphabetic-syllabic scripts, bringing the percentage of languages sampled by Shalonova and Tucker (2003) that meet the assumptions of the thesis to 79%.

[4]The database at `http://llsti.org/languages-database.htm` was accessed in January 2012 for the collection of the data presented here. At that time, data for 107 languages were included in the database, compared with the 105 described in the publications cited. Malay, Panjabi and Sindhi each have double entries for different script types, bringing the count of database entries to 110.

[5]A brief typology of the world's scripts is given by Comrie (2011), where the terms *alpha-syllabic* and *logographic* correspond to *alphabetic-syllabic* and *ideographic* used in Table 2.2.

Script type	*Orthographic words?*	
	Yes	*No*
Alphabetic	70	0
Alphabetic-syllabic	20	6
Consonantal	9	0
Syllabic-ideographic	0	1
Ideographic	0	1

Table 2.2: *Script characteristics of 107 languages. Entries in the table are counts of languages in the database described in Shalonova and Tucker (2003). Malay, Panjabi and Sindhi are counted only under their non-Arabic script entries.*

The assumptions mentioned here are those that must be met in order that the types of module described in Chapters 4 to 8 are *trainable* for a target language. Other typological characteristics will determine whether or not the induced representations will actually be *useful* for TTS in a given target language. For example, given a morphologically rich target language, orthographic word types may be too sparsely represented for word type representations such as those described in Section 4.2.4 to have any capacity for generalisation. For word representations to be of use in such cases, they would have to be representations of word tokens, using for example automatically derived morpheme-like units as internal or external contexts (see Section 4.2). However, the distributional–acoustic method presented in this thesis works on the assumption that features derived from vector space models in the distributional phase will be rejected by the system in the acoustic phase if they are irrelevant to the acoustics of the target language. This rejection of irrelevant features is the task of decision tree building modules, aided by the method of feature selection developed in Chapter 7. If a vector space model – such as one of word types in a morphologically rich language – provides features that are wholly irrelevant, the assumption is that it will at least not *harm* the quality of speech generated.

Finally, it should be noted that this thesis deliberately avoids broaching the issue of normalisation of *non-standard words* (Sproat et al., 2001), such as the normalisation of *£5* to *five pounds*. It is assumed that the text part of the primary corpus and run-time input to the systems built consist of fully-expanded standard orthographic forms.[6] Text normalisation is considered to be too language-specific to be handled in a wholly unsupervised way, but could be amenable to an *active*

[6]Note that no such normalisation is expected in the copious tertiary data, creating a slight mismatch when the results of distributional analysis of the tertiary data are used to characterise textual objects in the primary data.

learning type approach. As such, it is beyond the scope of this thesis, but will be the focus of future work.

Chapter 3

Benchmark Systems

3.1 Introduction

The central question that this chapter seeks to answer is: How necessary are the features output by various front-end modules listed in Table 2.1 on page 7 for the successful operation of a speech synthesis system? To this end, three experiments are carried out, each of which focuses on a different area of the knowledge encoded in a conventional linguistic specification. These different areas of knowledge will now briefly be outlined, along with an overview of the experiments designed to evaluate their contribution to the quality of speech synthesised.

Experiment 1: Contribution of High-Level Features Given on page 7 is a list of contextual features used for the conventional English systems built for this thesis (Table 2.1), and it is on what will be termed the *high-level* features given in this list that the first experiment focuses. The term *high-level* is used to denote features that cannot be obtained from a lexicon and letter-to-sound rules: features relating to part of speech, intonational phrase boundaries, pitch accents, and boundary tones (features classes W, P and T in Table 2.1). The first experiment is intended to determine what these features contribute to the quality of synthesised speech. This is done by building a range of voices, each of which excludes a subset of linguistic features, and then subjectively comparing synthetic speech generated by them for naturalness.

Experiment 2: Contribution of Phonetic Transcription The second and third experiments focus on the phoneme-related features of the linguistic specification (feature classes F and FC in Table 2.1 on page 7). Specifically, the second experiment examines the contribution of phonemic transcriptions to the quality of synthetic speech, addressing the situation in which such a transcription

is not available. To what extent is it possible to naively use textual objects (letters) in place of expertly-specified ones (phonemes and associated phonetic classes: feature classes F and FC in Table 2.1) as elements of the linguistic specification? As in Experiment 1, a range of systems is built, to each of which is made available a different level of annotation, and the intelligibility of resulting systems is measured in a listening test. The experiment considers letter-based systems for a language with an alphabetic writing system with relatively opaque letter–sound correspondence (English).

Experiment 3: Contribution of Phonetic Categories examines the contribution that knowledge of phonetic categories (encoded in features of class FC in Table 2.1) makes to the quality of synthetic speech. Again, a range of systems is built, to each of which is made available a different level of phonetic knowledge. Evaluation is carried out through objective comparison of held-out natural samples and the systems' attempts at synthesising those samples. The motivation for this experiment is the fact that in languages with more transparent alphabetic orthographies than the target language for these experiments (English), using letters directly provides a similar representation to one made using a phonemic transcription. In the case of letter-based systems, however, phonetic classes such as those that are implicit in the design of a phoneme set are not generally established, making evaluation of phoneme-based systems more straightforward.

Different methods of evaluation are used in each of the three experiments. In Experiments 1 and 2, human listeners are used. In the former, they are asked to rank synthetic samples for naturalness, and in the latter, to provide transcriptions of synthesised utterances which can then be scored against the true transcription to provide a measure of system intelligibility. In either case, the aspect of synthetic speech is evaluated that is expected to be most clearly affected by the way linguistic annotation is varied between experimental conditions. Where phonemes are replaced with letters in Experiment 2, intelligibility will most obviously be affected. Where high-level features are varied in Experiment 1, it is expected that naturalness of synthetic speech will be more obviously affected. Experiment 3 makes no use of human listeners due to the time-consuming nature of listening tests. Here, objective evaluation of spectral envelope parameters is performed; this aspect of speech was chosen over – for example – fundamental frequency, as it is spectral envelope that is expected to be affected most when knowledge of phonetic categories is varied.

Insights gained in the analysis section of Experiment 1 suggest improvements to the baseline systems in both Experiments 2 and 3. The methods of improve-

ment are presented in the relevant sections, along with experimental results. The systems incorporating improved techniques are in some respects successful. However, they have some shortcomings and limitations that are discussed in the concluding part of this chapter. This paves the way for introduction of the methodology that is central to this thesis in Chapters 4–7.

3.2 Experiment 1: Contribution of High-Level Features

3.2.1 Overview

This experiment takes the availability of a lexicon and features derived from it (what will be termed phoneme-level features: phonemes, lexical stress, syllable boundaries) as given, as well as utterance boundaries.[1] It is assumed that out-of-vocabulary words are handled well or perfectly by relevant modules. These assumptions are made in order to focus on *high-level* features as already mentioned (feature classes W, P and T in Table 2.1). It is assumed that there is no means of assigning these in the baseline case.

Twelve synthesis systems were assembled for this experiment. They are summarised and given identifying codes in Table 3.1. For all systems, a common training set was used, details of which are given below. The only aspect of training varied between systems was the annotation of the speech data used. This annotation was varied firstly with regard to the highest linguistic tier of annotation that was included, and secondly with regard to how that annotation was obtained. The first type of variation is represented by the rows of Table 3.1: the systems on the bottom row of the table make use only of features given by a lexicon and utterance segmentation (feature classes F, FC, S and U in Table 2.1). Moving up the table, the systems incorporate higher levels of analysis (part of speech, phrase-breaks and finally pitch accents: feature classes W, P and T, respectively). Systems in the uppermost row use the full feature set, and so constitute top-line systems.

The three columns of Table 3.1 represent different degrees of automation of annotation of the high-level features. In column 1 are what can be thought of as *ideal world* systems, employing manually-constructed or manually-checked annotation at both training and synthesis time. This type of high-quality annotation

[1] An earlier presentation of Experiment 1 was made in Watts et al. (2010a).

Training labels:		Gold	Gold	Auto
Synthesis labels:		Gold	Auto	Auto
		(**G**old	**M**ixed	**A**uto)
Feature descriptions:	*Feature classes*:			
Basic POS Phrase ToBI	F FC S U W P T	G1	M1	A1
Basic POS Phrase	F FC S U W P	G2	M2	A2
Basic POS	F FC S U W	G3	M3	A3
Basic	F FC S U	G4	M4	A4

Table 3.1: *Summary of systems built to analyse the impact of using linguistic features, and the impact of prediction noise on these features. The feature classes given refer to Table 2.1.*

can be thought of as a gold-standard, and will here be referred to as *Gold* annotation (G). In column 3 are systems that represent a more common real-world case, in that both training and synthesis are done using annotation automatically predicted from text; this annotation is here referred to as *Auto* (A). The middle column represents a mixed condition (M), where *Gold* annotation is used during training and *Auto* during synthesis.

The variation represented by the columns of this table is motivated by a secondary concern of this experiment. This is the impact of noise in the labelling of these higher-level features on their usefulness to a system. Extensive subjective comparison of the G, M and A conditions was not performed in this experiment, but the use of features by all three types of system is analysed in Section 3.2.5.

A more detailed account of the differences between different systems' features is given below in Section 3.2.3. Besides these different features, identical data and training procedure were used. All systems were built using the recipe termed HTS-2005 in Section 2.1.3, and using the data that are detailed in the following section.

3.2.2 Data Used

The Boston University Radio News Corpus (BURN: Ostendorf et al., 1995) was selected because of the high quality of the associated annotation, much of which is manually assigned, including ToBI labels (Silverman et al., 1992). The speaker *f2b* was chosen as target speaker, being the speaker with the largest amount of data hand-labelled with ToBI. Only the *radio news* part of the corpus was used, in order to achieve consistency of speaking style. The amount of data

used (55 minutes, not phonetically balanced) is the minimum needed for decent performance by a speaker-dependent system. Combined with the fact that there is also some variation between the acoustic quality of different sessions, this means that voices of very good segmental quality cannot be obtained from these data. However, the voices were built on the assumption that it should be possible to evaluate their global prosodic characteristics, i.e. those which are expected to be most affected by high-level features.

The BURN data are distributed in paragraph-sized files which were split up into smaller subjectively-unified utterances for the purpose of ease of processing in this experiment. Data that were judged too noisy for this experiment or of markedly different acoustic quality to the majority of the data or which lacked ToBI annotation were discarded. The result was a set of 425 utterance waveforms (55 minutes in total).

A testset was prepared from BURN using the audio and associated labelling of two different speakers (*m1b* and *m2b*), because the *f2b* data were considered too small to be partitioned into training and test sets of reasonable size. Two sets of annotation of this testset (*Gold* and *Auto*) were prepared in the same way as for the training set.

3.2.3 Annotation Used

Two sets of linguistic specifications were prepared for the training and test data, which will here be called *Auto* and *Gold*. The features specified in both sets of annotation are those summarised in Table 2.1 with two exceptions. First, part of speech of previous, current and following words (using utterance POS tags) were used instead of *guess POS*. Second, the types of pitch accent of previous, current and following syllables were used as features, not simply the true–false value used in the standard contexts (that is, whether or not a pitch accent is predicted for a given syllable). In both cases, manually-specified sets of POS tags and accents (such as *verb*, *function word*, *pitch-accent containing H*, etc.) were used.

Note that the differences between the *Gold* and *Auto* annotation account for the columns of Table 3.1. To create the conditions represented by the rows of that table, the relevant parts of these linguistic representations were suppressed for each voice built.

The lexicon and utterance-level features (features of classes F, FC, S and U in Table 2.1 on page 2.1) for both sets were derived in the same way from text transcriptions, using the automatic procedure outlined below. The high-level

features of the *Auto* annotation were also derived using automatic procedures. In the case of the *Gold* annotation, on the other hand, manual or manually-verified annotation was used for all these higher level features, as described below.

Auto: Automatic Annotation

The automatic annotation (including annotation of features relating to part of speech, phrase breaks, pitch accent, and boundary tones) was produced following the procedure outlined on page 25. The Carnegie Mellon University Pronouncing Dictionary was used, and out-of-lexicon words in both training and test data were added manually to be able to focus on the contribution of the high-level features.

It should be noted that the two CARTs used to assign pitch accents and boundary tones had been trained on data from the target speaker of the present experiment, *f2b*. Although these CARTs had been trained with a view to generalising to other speakers and not overfitting *f2b*'s characteristics, these trees' predictions are obviously expected to be much better on this – their training data – than on arbitrary data. Note that the two testset speakers, however, had made no contribution to the training data for Festival's ToBI-prediction trees.

Gold: Gold Standard Annotation

As mentioned above, the **phoneme-level features** (phoneme sequence, lexical stress, syllabification) were identical in the *Auto* and *Gold* annotation.

The part of speech (POS) tags supplied with the *radio news* sections of BURN are automatically assigned and not manually checked. For the *Gold* annotation, therefore, a new high-quality POS tagging was produced by running three high-quality taggers of different types over the c.10,000 tokens of the training corpus, accepting unanimous decisions of the taggers, and manually tagging the remaining 7% of the tokens in accordance with Penn Treebank tagging guidelines.[2]

Phrase-breaks: The annotation provided with BURN gives manually assigned phrase-break indices. Breaks with index 4 were used to provide phrase-breaks in the *Gold* annotation; other indices including those for intermediate phrase boundaries were discarded.

Pitch accents and boundary tone features in the *Gold* annotation were derived from the manual labelling in BURN. With the exception of intermediate

[2]The three taggers used were a trigram tagger (*TnT*: Brants, 2000), a maximum entropy tagger (*MXPOST*: Ratnaparkhi, 1996), and Brill's Transformation-Based Learning tagger (Brill, 1992). Models that had already been trained on the Wall Street Journal data from the Penn Treebank were used.

phrase tones and %H accents, which were discarded, accents were associated with syllables and boundary tones with phrases in the *Gold* annotation.

3.2.4 Subjective Evaluation

Procedure

An AB test was conducted in which a pairwise comparison was made between eight selected pairs of six of the systems built in terms of listeners' impression of the naturalness of the synthetic speech. Five comparisons were made among systems G1, G2, G3 and G4 to assess the impact of removing high-level features: the comparisons made were G1-G2, G1-G3, G1-G4, G2-G3, G3-G4. Three comparisons were made among systems G1, M1, and A1 to assess the impact of prediction noise while keeping the feature set constant.

80 medium-length utterances (4–10 s.) were taken from the test set. 80 synthetic stimuli could therefore be used for each of the 6 systems to be evaluated. The utterances were randomly assigned to 8 *utterance sets*. Each listener was presented with each of the 8 system comparisons as ten (same-utterance) pairs from a single utterance set; no listener received the same system comparison from the same utterance set, and no listener heard the same utterance in more than one pair in the course of the entire evaluation. Within-pair ordering of systems was balanced within each utterance set. Finally, the presentation order of each listener's pairs was randomised, and the pairs presented in 4 blocks of 20. The listening test was conducted via a web browser and headphones in purpose-built listening booths, with a total of 8 paid listeners (ages 18–25, all native speakers of English). The listeners were asked to listen to the pairs and record their preference for the more natural-sounding utterance.

Results

Results of the paired comparisons are presented in Figure 3.1. There, error bars show 95% confidence intervals (with Bonferroni correction) from a binomial test. One preference was detected as significant (G1 vs. G4), and the overall trends are consistent with what is expected: systems perform worse the more tiers of high-level annotation are removed, and there is a trend of preference for systems using hand-labelling over ones using automatic labelling. It can also be noted that different types of feature seem to differ in the importance of their contribution to listeners' preference, in particular that the use of pitch accent, boundary tone

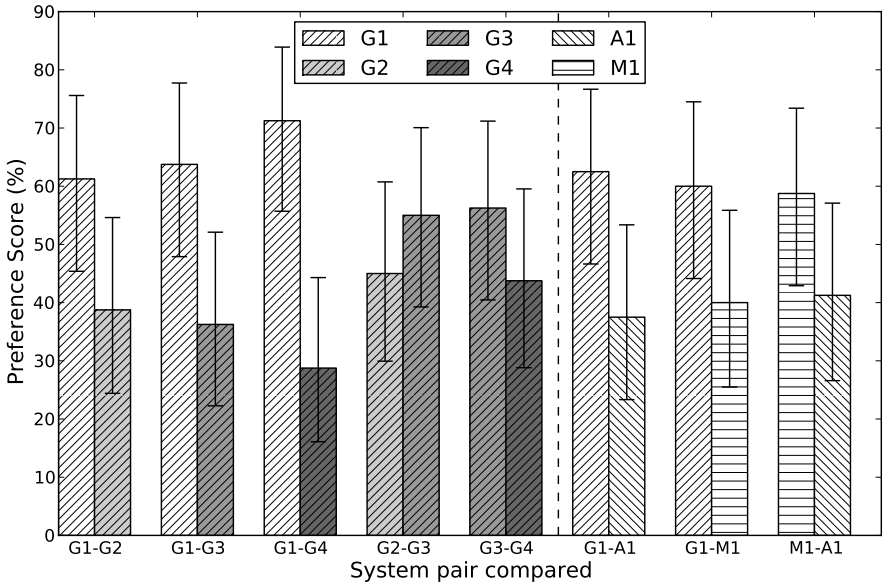


Figure 3.1: *Results of AB test for naturalness in Experiment 1. Error bars show 95% confidence intervals (with Bonferroni correction).*

and POS features seems to contribute more to preference scores than the use of phrase features. A more extensive evaluation with more listeners might be expected to detect more significant differences in support of these trends.

3.2.5 Systems' Use of Linguistic Features

To gain insight into the types of question most used by each system, and to see how this changes as conditions are varied, the data represented in Figure 3.2 were gathered. For each system the entire testset (consisting of 10,456 context-dependent phoneme tokens) was synthesised, and a count was made of the number of times each linguistic feature was queried as the trees for deciding log F_0 distribution were descended. Trees for F_0 were chosen here (rather than e.g. trees modelling spectral parameters) because this is the part of the model which is expected to be most affected by modifications to the higher-level linguistic features. The counts were categorised by the types of feature given in Table 2.1 on page 7, and normalised by the number of features queried for each system, resulting in the columns of Figure 3.2.

The overall distribution of features used over classes is much more similar between the G and M systems than between the M and A systems, from which it might be deduced that the type of annotation used in training is more important than the type used at synthesis time in determining which features define units

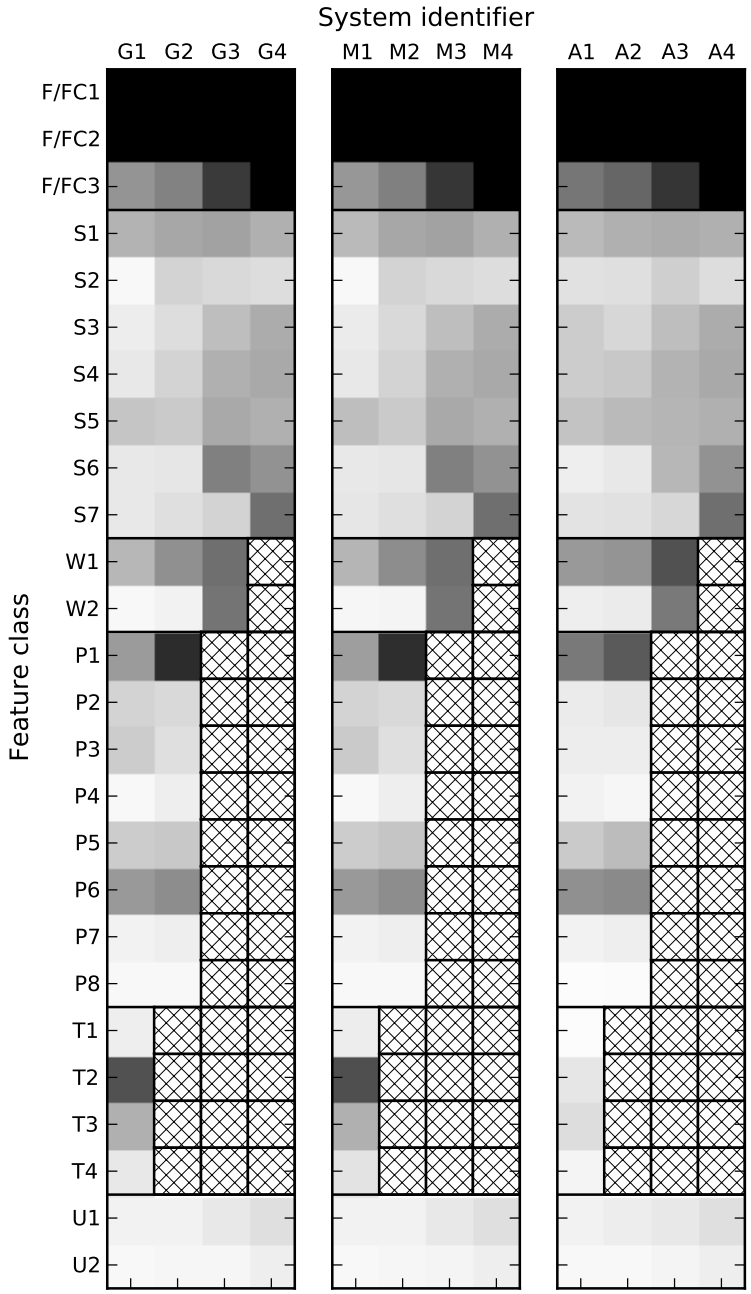


Figure 3.2: *Type of questions asked during synthesis of test set in Experiment 1. For each voice, the % of questions (tokens) asked during synthesis from each type is given as a grey-scale value (black = 11%, white = 0%: note the 11% ceiling obscures details of monophone and triphone values but enables small values to be presented with greater accuracy). Hatching indicates that a group of features was held out from a system. The identifiers for question classes are explained in detail in Table 2.1. Briefly, 'F' and 'FC' questions refer to phones, 'S' to syllables, 'W' to words (i.e. part-of-speech), 'P' to phrase, 'T' to tone and accent, and 'U' to utterance.*

in synthesis. The tendency for a greater proportion of questions to be asked about lower level features (e.g. quinphones) as higher-level features are removed is the sort of surrogacy effect that might be expected. But Figure 3.2 also reveals more specific surrogacy effects as higher-level features are removed. In all 3 conditions G, M and A, for example, when pitch accent and tone-related features are removed (categories T1–4), there is a sharp increase in use of questions from category P1 (*Number of syllables* {*since, until*} *phrase boundary*). Likewise, when questions from the phrase category (P1–8) are removed, questions in the POS class (W1–2) see a sharp increase in usage. These effects are what might be expected given that exactly these types of features are used as predictors of pitch accents and phrase-breaks in the front-end modules used. What seems to be happening is that similar combinations of e.g. phrase features as those used to assign ToBI events by a CART tree are being found directly in the decision tree for state-clustering when ToBI labels are removed from the system. It appears that acoustic model clustering trees are able find useful combinations of features from lower tiers of the linguistic specification directly. The methods introduced in the next two experiments are essentially ways of enhancing the trees' ability to find such combinations.

3.3 Experiment 2: Contribution of Phonemic Transcription

3.3.1 Letter-Based Speech Synthesis

The first experiment of this chapter confirmed that features used in a conventional TTS system that are obtained from modules other than the lexicon can significantly improve a system's perceived naturalness. The present experiment, on the other hand, examines the contribution made to the quality of synthetic speech by the lexicon through the phonemic transcription it provides.[3]

Similar work that attempts TTS based on letters is reported in Black and Font Llitjos (2002) in the context of cluster-based unit selection synthesis. The target language in that case was Spanish; the notoriously complex and irregular letter-to-sound correspondences of English make using it as our target language very ambitious. This is also shown by the findings of Killer et al. (2003), where the performance of grapheme- and phoneme-based systems on speech recogni-

[3]An earlier presentation of Experiment 2 was made in Watts et al. (2010b).

Identifier	Description	Modelling unit	Run-time lexicon and CART training data	Decision Tree Method
L-BAS	Letter-based baseline	Letter	*n/a*	1-pass
L-SER	Letter-based, serial tree-building	Letter	*n/a*	Serial
P-FUL	Phoneme-based with full lexicon	Phoneme	Full CMU lexicon	1-pass
P-LIM	Phoneme-based with limited lexicon	Phoneme	CMU lexicon entries for training set items	1-pass

Table 3.2: *Summary of systems built for Experiment 2*

tion tasks in German, English and Spanish are compared. Word error rates for grapheme systems are slightly higher than for phoneme systems in the case of German and Spanish, but significantly higher in the case of English.

Systems for this experiment are summarised in Table 3.2 and explained in the following paragraphs. All systems were trained using the audio data and text transcription for speaker *slt* in the ARCTIC database (Kominek and Black, 2004). The transcription was checked before use and manually preprocessed, and all numerals and abbreviations correctly expanded. The same procedure (denoted HTS-2005 and summarised on page 25) was used to build the acoustic models of all systems except L-SER, where this procedure was extended with a *serial tree-building* procedure at the final iteration of context clustering of spectral envelope parameters (motivated and described below). As in Experiment 1, the principal difference between systems is due to the different annotation that they employ, which is now described.

The systems that are given the identifiers L-BAS and P-FUL are a baseline and topline system, respectively. System P-FUL uses a phonemic transcription obtained from a pronouncing dictionary and derived letter-to-sound (LTS) rules. System L-BAS is identical, except that the lowercased letters of a standard orthography transcription are used in place of phonemes. This pair of systems allows the main prediction of the experiment to be tested: that using phonemic transcriptions derived from plain orthography gives better synthetic speech than using the plain orthography directly.

P-FUL Details

The annotation used to build the P-FUL system was obtained by a procedure similar to the one described on page 25. That is, an initial time-alignment and phoneme sequence was found by look-up in the CMU pronouncing dictionary (CMU) and forced alignment with HMMs. All out-of-vocabulary words found in the *training* data were added manually to the lexicon. The location of punctuation marks was used to initialise a silence model, and later the insertion of silence between words was allowed where supported by the audio. Selection of alternative pronunciations from the lexicon was also allowed during alignment. From the resulting phoneme sequences, simple features were derived, consisting of the identity of phonemes at each position of a 7-unit context window and membership of those phonemes to conventional phonetically-motivated categories. In addition, the number of phonemes since the start of the word, and the number of phonemes until the end of the word. No features above the word level (relating to e.g. position in phrase or utterance) were used. The phoneme inventory used consists of 54 units, including 15 stressed variants of vowels. To perform synthesis with this system, the CMU pronouncing dictionary was used to look up phoneme sequences from plain text. Out of vocabulary (OOV) words were converted using CART trees trained on the whole CMU lexicon.

L-BAS Details

The annotation used for system L-BAS was identical to that of P-FUL except for the lexicon used: instead of the CMU lexicon used by System P-FUL, L-BAS uses a *naive lexicon*, mapping tokens to sequences composed of the 26 lowercase letters of English. In all other respects the procedure used for obtaining the annotation for the two systems was identical. Note that the letter-based annotation of System L-BAS referred to no sets of units comparable with the phonetic classes used in System P-FUL's annotation. Informal visual comparison of the phoneme- and letter-based alignments shows that at the word level they are very similar, and that reasonable assignments of letters to acoustic segments are generally made when letters are used.

The use of a wider context window over sub-word units than elsewhere in this thesis – seven units rather than five – is inspired by features typically used in building CART trees for LTS conversion. Note that unlike in LTS trees, the context units in the window may also be taken from neighbouring words, as the features are expected to deal not only with LTS correspondences but also with

the type of co-articulatory effects for which decision-tree-based context clustering is conventionally used.

3.3.2 Phonemes: Modelling vs. Generalisation

Two other predictions are made in this experiment, and voices built to test them. The first prediction is that the expected superior performance of P-FUL over L-BAS is due to the capability for generalisation given by a pronouncing dictionary, rather than due to the inherent superiority of phonemes over letters as units for acoustic modelling. To allow this prediction to be tested, system P-LIM is built, which is identical to P-FUL with regard to training, but which at synthesis time is constrained to make predictions of phoneme sequences purely on the basis of the phonemic transcriptions observed in training. That is, lookup is performed in a lexicon limited to the forms encountered in the training set, and the pronunciations of out-of-vocabulary words are provided by a CART tree trained on this limited lexicon. P-LIM therefore provides a useful point of comparison with L-BAS, which must make all generalisation on the basis of items seen in training, but which also must map directly from surface forms of words to acoustics without the mediation of a phonemic transcription.

P-LIM Details

The annotation of training data and models build for system P-LIM were identical to those used for system P-FUL. The only difference was the LTS resources used at run time. Whereas P-FUL uses the full CMU lexicon, both for lookup of words and for building of decision trees for the letter-to-sound conversion of OOV words, the lexicon used for lookup and training LTS trees used for P-LIM was restricted to the entries in CMU lexicon of the 2333 word types seen during training.

3.3.3 Induced Compound Features

The final prediction made is that a *serial tree-building technique* can significantly improve the performance of a system. To test this, System L-SER was built, which starts with the same features as L-BAS but through a modified tree-building routine, acquires richer representations of letters in context by means of serial tree-building.

The use of serial tree-building is motivated by the well-known weakness of tree-based methods when multiple features must be queried simultaneously to

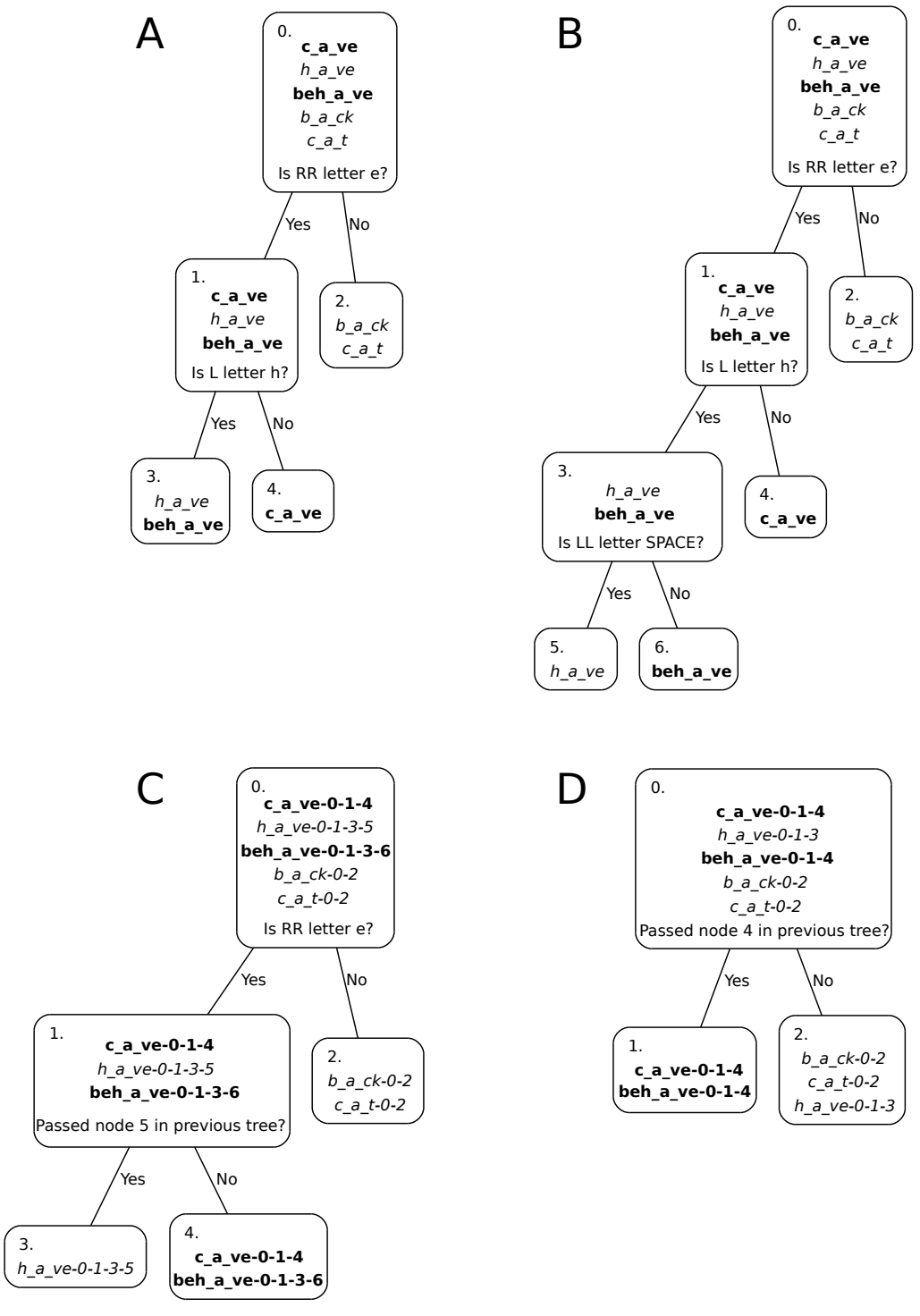


Figure 3.3: *A toy example of serial tree building for the characterisation of letter tokens.*

achieve useful partitions of a dataset (Breiman et al., 1993, pp. 136ff). Such is the case with sets of rules which capture English letter-to-phoneme correspondences. An example of a dataset for training a model to predict the realisation of ⟨a⟩ as either /a/ or /ei/ is the set of words shown in node 0 of the tree in Figure 3.3A. This diagram could represent either a CART tree for letter-to-phoneme rules or a tree for acoustic model clustering where letter-based features are used. Questions querying a single feature of these examples are not sufficient to split the set of words appropriately. For example, the question *Is the letter 2 places to the right an <e>?* fails because of the exceptional pronunciation of the ⟨a⟩ in *have*; this exception means that only a Boolean combination of features can split the set appropriately. Asking questions one at a time, as in standard tree-building procedures, leads either to impure nodes if splitting stops in the state depicted in Figure 3.3A, or over-fragmentation if splitting continues till the nodes are pure (as in Figure 3.3B, where items that would be best kept together are split apart, both in nodes 2 and 5 and nodes 4 and 6).

This phenomenon can be observed in real trees built for letter-based systems such as L-BAS. Over-fragmentation and under-fragmentation often occur together in different portions of the same tree. This is due to the Minimum Description Length criterion used to determine at which point tree-building should cease (see Section 2.1.2). This criterion is designed to balance the increasingly good fit of the model to the data and the concomitant increasing complexity of the model in an appropriate way. However, Description Length is computed globally over the tree as a whole. In effect, by creating many pure but fragmented clusters early in tree-building, we are getting bad value in terms of increased likelihood for the extra model parameters used. If free parameters are wasted through fragmentation in one part of the tree, it is understandable that splitting could stop in a locally premature way in another part of the tree. Qualitative evidence for this phenomenon is given in Table 3.7 in connection with the final experiment in this chapter. Note that empirical investigation shows that heavy fragmentation is not detrimental to the predictive performance of CART trees built for letter-to-sound conversion and that splitting till total node purity gives the best results (Black et al., 1998). The case of decision tree building for state-tying of acoustic models using letter-based units, however, is rather different, in that model complexity control is crucial to ensure adequate training data for each HMM state.

Various methods have been proposed to overcome this problem with tree-building. The tree-building algorithm can be reformulated so that each node is split on a combination of multiple independent variables. Many procedures for

finding useful variable combinations in trees work by considering some subset of possible combinations one node at a time, in a single pass of tree building. Cantu-Paz and Kamath (2003, §II) provide a useful review of approaches to the induction of such *oblique decision trees*. A different approach to finding combinations of variables is the serial method explained in Shafran and Ostendorf (2003) on which the technique used in the present experiment is very closely based.

In this method, trees are built iteratively. Starting with simple features, a tree is built that clusters some training data, and the names of nodes of this tree are added as features to the names of the models that enter them. The tree is then put to one side, but questions can now be asked about the new features it has provided in subsequent iterations. The tree produced in the final iteration is the tree that is finally used in the normal way.

As a toy example, take the tree in Figure 3.3C. We start by placing all model names in the root node (0), and appending the names with features indicating through which nodes they have passed on a previous iteration of tree-building (i.e. the tree in 3.3B). For example, to the *cat* model are appended the features 0 and 2, indicating that the model traversed those nodes of the previous tree (3.3B). Querying these features is equivalent to querying multiple original features simultaneously. At node 1 of 3.3C this is done, and results in a less fragmented tree than 3.3B. The procedure can be repeated, as in 3.3D: the models are renamed with the compound features found by traversing 3.3C, and reference to them leads in 3.3D to a final, perfect split of the data. In effect, serial tree-building allows questions to be asked (indirectly) about several linguistic attributes simultaneously: the new features represent Boolean combinations of the original features with the AND and NOT operators.

System L-SER was constructed in the same way as System L-BAS, except that the *serial tree-building technique* that has already been described is used. Five iterations of the procedure are used for the final clustering of spectral parameters of system L-SER; HMM parameters are not reestimated between iterations of serial tree building for this system (i.e. HMM parameters receive the same amount of training for systems L-BAS and L-SER).

3.3.4 Evaluation

It is predicted – uncontroversially – that use of the phonemic transcription will be beneficial to the intelligibility of synthetic speech. The principal hypothesis of this experiment therefore is that listeners will make significantly fewer transcription

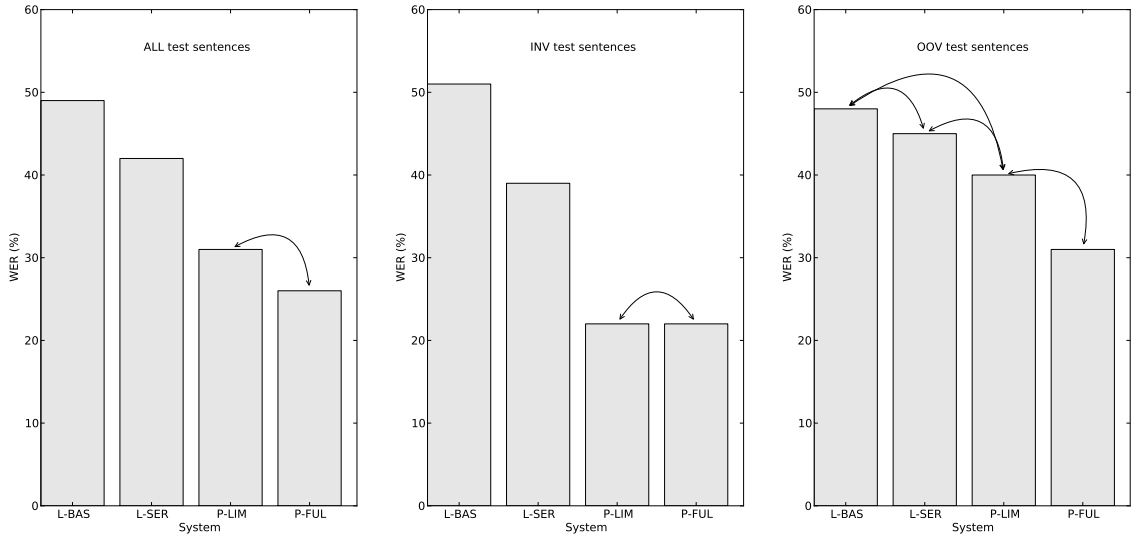


Figure 3.4: *Results of Experiment 2. WERs for all test sentences (ALL: left), sentences with in-training-vocabulary content words only (INV: middle), and sentences with out-of-training-vocabulary content words only (OOV: right) are shown. Arcs show pairs of systems where bootstrap-t confidence intervals over system differences show no significant difference (with $\alpha = 0.05$ and Bonferroni correction).*

errors when transcribing speech generated by P-FUL than by L-BAS.

A web-based evaluation of the intelligibility of the voices built was conducted on Amazon's Mechanical Turk.[4] This is a web-based platform that allows short tasks requiring human intelligence to be posted and completed on the web for payment. Many language experiments have been reported that use the service (e.g. Tietze et al., 2009). 40 listeners were obtained in this way to evaluate Semantically Unpredictable Sentences (SUS: Benoit et al., 1996) synthesised by the systems. 40 such sentences were produced using each system, 20 of which where the content words were not to be found in the systems' training vocabulary (the OOV portion of the test-set), and the other 20 so that all the content words had been 'seen' by systems during training (the INV portion). Listeners were assigned to one of 4 groups (each with 10 listeners); the groups were designed so that each group's listeners heard a different set of system–sentences, but so that the same sentences were heard for each system over the whole test. Stimuli were presented in random order to the listeners, and the listeners were asked to type what they heard. Word error rates (WERs) were then computed on the listeners' responses, with reference to the text that had been used to create the stimuli.

[4]https://www.mturk.com/mturk/

System	Description	Features
Q	Baseline	Quinphones only
QC	Topline	Quinphones and phonetic Categories
QH	Experimental	Quinphones and automatically Harvested categories

Table 3.3: *Summary of systems built for Experiment 3.*

3.3.5 Results

Results of the evaluation are summarised in Figure 3.4. WERs are given over all test sentences (left), sentences with in-training-vocabulary content words only (middle), and sentences with out-of-training-vocabulary content words only (right). Differences between system WERs were compared in a pairwise fashion using the bootstrap procedure outlined in Bisani and Ney (2004): bootstrap-t confidence intervals were calculated over system differences. Differences not found to be significant in this analysis (with α=0.05 and Bonferroni correction) are indicated with arcs in the figure.

On both the INV portion of the test set (centre plot of Figure 3.4) and on the OOV portion (right-hand plot of same figure), the phoneme-based systems achieve lower WERs than the letter-based ones, as expected. For the INV set, the two phoneme-based systems receive the same WER as we would expect, as they are essentially the same system when producing this 'seen' vocabulary. On the OOV set, the limited-lexicon phoneme-based voice P-LIM has a higher WER than counterpart P-FUL, although this difference between the P voices is not found to be significant.

The serial tree-building method produces a significant improvement to the baseline letter-based system in both the overall evaluation (left-hand plot of Figure 3.4) and evaluation on the INV portion of the test-set (middle plot in same figure). Also on the OOV portion of the test-set (right-hand plot of Figure 3.4), L-SER achieves a lower WER than L-BAS, although in this case it is not found to be significant. In no case does performance of the L systems approach that of the full phoneme-based system, P-FUL. On the OOV test-set, though, the addition of serial tree-building allows the letter-based system to close a part of the gap in performance between the baseline system L-BAS and the phoneme-based system with limited lexicon, P-LIM. Here, although there remains a gap between L-SER and P-LIM, it is not found to be significant (though as noted above, neither is the gap in performance between L-BAS and L-SER in this case).

3.4 Experiment 3: Contribution of Phonetic Categories

3.4.1 Overview

The present experiment examines the contribution that knowledge at the phoneme level makes to the quality of synthetic speech, specifically, knowledge of phonetic categories. As in Experiments 1 and 2, a range of systems is built, each incorporating a different level of phonetic knowledge. In this experiment, evaluation is carried out through objective comparison of held-out natural samples and the systems' attempts at synthesising those samples.

As observed in the introduction to this chapter, the motivation for this experiment is that in languages with transparent alphabetic orthographies, using letters directly provides a similar representation to one made using a phonemic transcription. In the case of letter-based systems, however, recognised classes such as those that are implicit in the design of a phoneme set are not generally established, which makes the construction of a benchmark system for comparison in evaluation problematic.

3.4.2 Systems Trained

Details of the systems built for this experiment are given in Table 3.3. These systems are identical with respect to the the data-set and training recipe used. The systems were trained on speech from the database released by Phonetic Arts for the 2010 Blizzard Challenge (King and Karaiskos, 2010). The whole database consists of the speech of 4014 isolated factual utterances read from a script by the speaker *rjs*, a professional 50-year old male RP-accented speaker. For training voices in this experiment, a subset of this data was used, consisting of its first 1000 sentences (approximately 1.6 hours of speech).[5] The training recipe denoted HTS-2010 on page 25 (with 10 iterations of context clustering) was used to train all three voices on this data-set.

The main comparison of the experiment is enabled by systems Q and QC, which are baseline and topline systems, respectively. These voices differ only with respect to the linguistic specification that they use. Both have access to a phonemic transcription of the training corpus and the quinphone features derived

[5]For convenience, this subset will be referred to as *RJS-1000* from now on in this thesis.

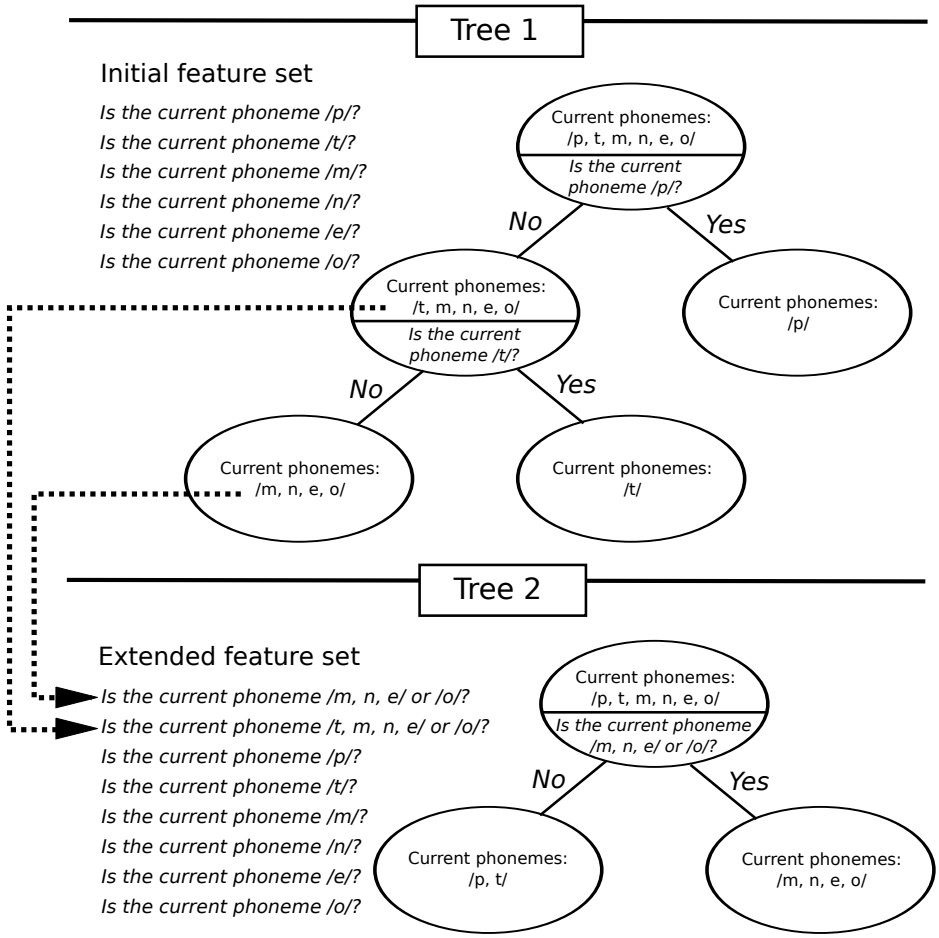


Figure 3.5: *A toy example of serial tree building for the characterisation of phoneme types*

from it (the features of class F on page 7), but only QC has access to phonetic category information (the features of class FC). It is predicted that the phonetic category information to which system QC has access will be beneficial to the quality of speech synthesised. The prediction made, therefore, is that the Bark cepstral distortion of speech generated by system QC compared with the natural reference speech will be lower than that of system Q.

The third system – QH – makes use of a modified training procedure based on the serial tree building technique described in Section 3.3.3. The method described there in effect iteratively finds combinations of simple features using the AND and NOT operators. The method developed for the present experiment is appropriate for finding sets of features similar to phonetic categories: compound features using the OR operator. The method is as follows: after each of the 10 iterations of context clustering, new features are harvested from the trees built. Features are harvested as follows: for each quinphone position in each node of a

tree, a category is formed from the phonemes that appear in the contexts that have reached the node. Figure 3.5 illustrates this with a toy example. Two trees are shown, and in each node of these trees, the phonemes in a single quinphone position (current phoneme) that have passed down to that node are shown. In the root node of both trees, this is a set of six phonemes: /p, t, m, n, e/ and /o/. Tree 1 uses the initial feature set for splitting the dataset, in which models are queried with regard to only single phoneme identities. That is, the system incorporates no knowledge of phonetic categories with which the dataset can be partitioned in other ways (e.g. unvoiced stop, nasal, vowel). It is assumed for the sake of demonstration that the shortest description length of the data is given by pooling the contexts with unvoiced stops as current phoneme in one leaf node, and all other contexts in a second leaf node. Given this, any tree built using the initial set of features will be suboptimal. Tree 1 is an example of this: the unvoiced stops /p/ and /t/ must be partitioned from the rest of the contexts one at a time, leading to over-fragmentation.

When harvested, however, tree 1 provides two categories of phonemes, which were produced when /p/ and /t/ are removed: /t, m, n, e, o/ and /m, n, e, o/. When described as /m, n, e, o/, this category does not seem to consist of phonetically homogeneous elements. However, the complement of this category – in the context of this example – is the category of unvoiced stops, /p, t/. A set and its complement obviously have the same status in decision tree-based state-tying. The use of these two categories by the second tree shown in the example means that it is capable of directly partitioning unvoiced stops from the other contexts by querying a single feature.

This is obviously a toy example, greatly simplified compared with a tree induced on a real dataset. When used to build system QH, the procedure was applied to features for all five quinphone positions, each of 10 iterations of context clustering to all trees for spectral envelope. Only the trees for spectral envelope were harvested for features in this experiment as it is these that are most related to segmental quality of speech, but in principle any of the trees built can be harvested in this way.

Note that the use of the HTS-2010 recipe in this experiment – with its many iterations of state-tying – means that the serial tree method can be incorporated very simply compared with the procedure used for System L-SER in Experiment 2, where no re-estimation of HMM parameters was conducted between trees in the series. It is also simple compared with techniques developed in speech recognition which also have the goal of automatically determining phonetic categories. These

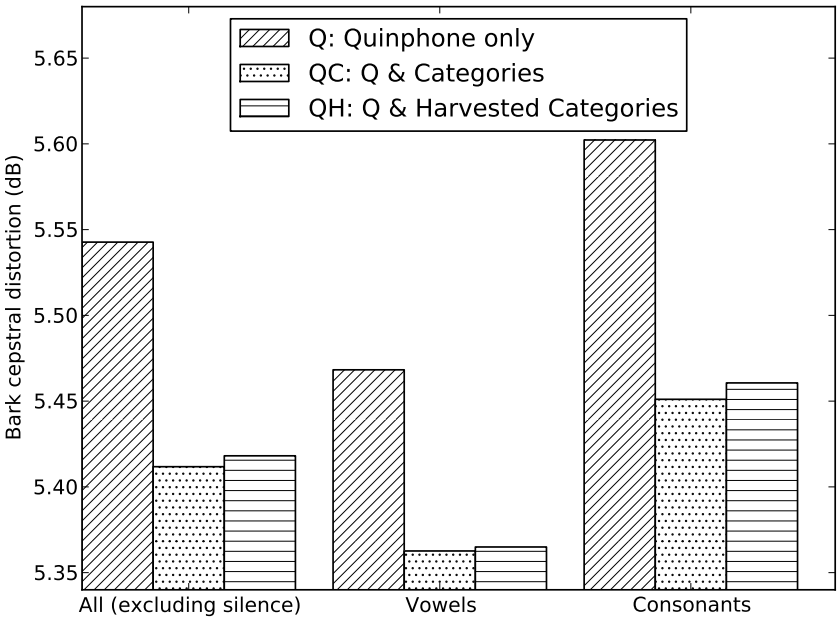


Figure 3.6: *Results of Experiment 3*

techniques (e.g. Beulen and Ney, 1998; Singh et al., 1999) typically involve a clustering stage additional to that used to build the final recognition models.

3.4.3 Evaluation

An objective evaluation of the three voices was carried out, by taking 100 utterances for which time-aligned annotation was available but which were held out of training, and synthesising them with natural segmental durations. The synthesised speech parameters could then be compared on a frame-by-frame basis with the parameterised natural samples. Mean Bark-cepstral distortion per non-silent frame was computed for this experiment, because spectral envelope is the attribute of speech that we expect to be most affected by the manipulation of the phonetic categories used by a system. Segments labelled as pause and silence were excluded throughout. As well as a general evaluation over all non-silent segments, measures were also computed separately over vowel and consonant segments.

Results are presented in Figure 3.6. It can be seen that encoding expert-specified phonetic categories in the annotation used leads to a reduction in Bark cepstral distortion. However, using the serial tree-building method to induce categories of phonemes leads to a comparable reduction in distortion without incurring any of the cost associated with manually compiling phonetic categories.

The serial tree building methods used in both this experiment and Experi-

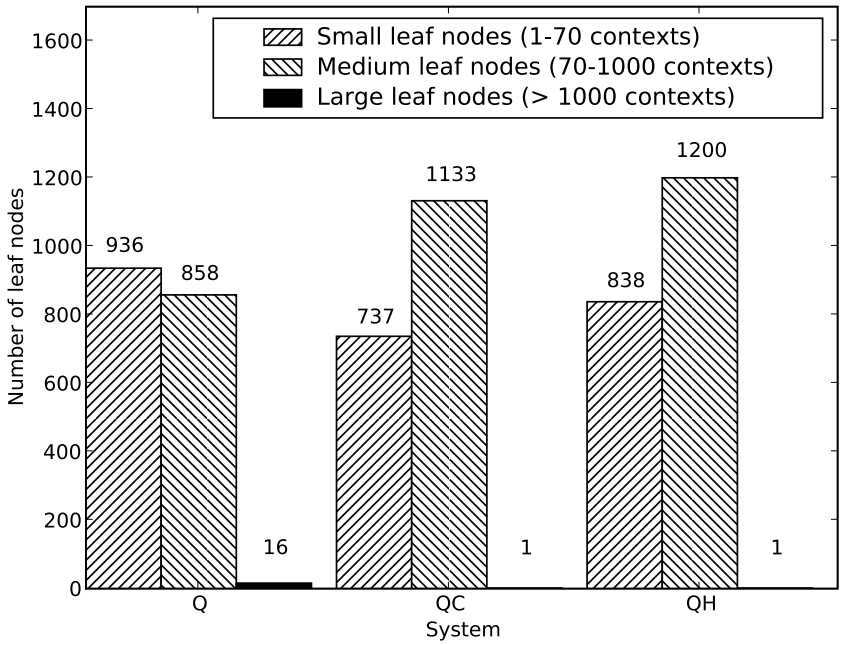


Figure 3.7: *Sizes of leaf nodes (number of context types from training data in each leaf-node)*

ment 2 are designed to overcome the problems of over- and under-fragmentation during treebuilding. Results of subjective and objective evaluation shown that quantitatively, systems incorporating these procedures behave more like toplines. Informal inspection of trees suggests that they do so also in qualitatively comparable ways. The data presented in Figure 3.7 demonstrate this. Here, all leaf nodes of all decision trees for Bark cepstrum state emission parameters are divided into three classes depending on the number of context-dependent phoneme types which that node represents. *Small* leaf nodes represent 70 or fewer types, *large* leaf nodes represent more than 1000 types, and the remaining leaf nodes are called *medium*. It can be seen from this figure that system Q contains more small and large leaf nodes than system QC, built using splits determined by phonetic category and not just phoneme identity. This greater number of small and large leaf nodes is a consequence of the combined over- and under-fragmentation mentioned above. Using the categories harvested with the serial tree method in system QH gives a distribution of leaf-node sizes much more similar to that of system QC than that of system Q. It seems therefore that the serial tree method not only allows a baseline system to approach the performance of a system using expert knowledge of phonetic categories quantitatively (i.e. as measured in cepstral distortion), but that it allows it to do so in a qualitatively comparable way.

3.5 Conclusions

In its analysis of the contribution of high-level features of a conventional linguistic specification to the quality of synthesised speech, Experiment 1 provides the foundation for the work presented later in this thesis, where less knowledge-rich alternatives to these features are sought. The subjective evaluation has established that these features do significantly improve listeners' reaction to synthetic speech, when features are assigned without error and combined. Perhaps more interesting are the findings from analysis of which features are used during synthesis, and the patterns of surrogacy between different tiers of features that this analysis reveals. Front-end modules provide linguistically-motivated combinations of input features which are then used by decision trees for state-tying. When those front-end modules are missing, however, it is clear that the state-tying trees are finding the same sorts of combinations directly. This inspires the adoption of serial tree methods in Experiments 2 and 3 as a means of facilitating this direct discovery of feature-combinations.

Experiments 2 and 3 show that – fairly obviously – it is beneficial to use phonemic representations and expert-specified phonetic categories when they are available. Failing that, however, two variants of a serial tree building were introduced that significantly improve the performance of a simple letter-based system when a phonemic transcription is absent, and close most of the performance gap as measured objectively in the case that knowledge of phonetic categories is unavailable.

The serial tree method is not without its shortcomings, however. In Experiment 3 it was used to characterise a *closed set*: the set of context-independent phonemes. It is reasonable to expect a system to hear all members of this set several times in a primary corpus of reasonable size. It makes no allowance, however, for items unheard during training in the case that it is used to characterise objects from an open set, such as word types. The variant used in Experiment 2 does this, as it characterises the large set of letters-in-context, all members of which cannot hope to be heard in training. It generalises to unheard cases, however, with limited success, providing a significant improvement over the baseline system only where the content words of the test-set are of heard types (central panel of Figure 3.4). Where the test-set includes content words of unheard type, WER is lower than for the baseline system when serial tree building is used, but not significantly so (right-hand panel of Figure 3.4). The following chapters develop

a method of *distributional–acoustic analysis* that allows a system to generalise to unheard (but not necessarily unseen) items.

Chapter 4

Vector Space Models for Speech Synthesis

The present chapter outlines the vector space model and its proposed application to text-to-speech (TTS) conversion. The vector space model is an unsupervised approach to learning representations for orthographic objects that underpins the work presented in the rest of the thesis. First, details of vector space models (VSMs) in general are given via a toy example. Then the way the VSM is applied to TTS in this thesis is outlined. The final two sections of the chapter are an overview of previous work on vector space modelling for the characterisation of linguistic and textual objects, and of some alternative approaches to this same task.

4.1 Vector Space Modelling: an Example

The *vector space model* (VSM) is well established in Information Retrieval (IR) and Natural Language Processing (NLP) as a way of representing objects such as documents and words as vectors of descriptors. The descriptors are typically selected or extracted from a database in an unsupervised manner. Selection or extraction of descriptors has the aim of placing items that are in some sense similar to one another close together in the vector space. The type of similarity that the vector space is intended to capture depends on the application, and influences the way in which descriptors are extracted. In its original incarnation in IR, documents are characterised by descriptors which indicate the presence of specific index terms attached to the document (possibly modified by some weighting scheme). For this application, it is similarity of subject matter that

is relevant, and the choice of index terms as descriptors achieves this. But the scheme has much wider application, as borne out by later work. In vector space models of word types, for example, the proximity of two words in the space might be required to reflect those words' syntactic or semantic similarity. Such spaces might be built by using words' neighbours as descriptors. A narrow context window in which immediate neighbours are counted is typically used when the space is required to capture syntactic similarity (as in e.g. Finch and Chater, 1992), and a wider one where semantic characterisation is the goal (as in e.g. Lund and Burgess, 1996). A toy example of a vector space model of word types designed to capture syntactic tendencies will now be given as a way of introducing some typical mechanisms of a VSM.

Take, for example, the following small vocabulary:

finds, *requires*, *makes*, *education*, *research*, *cooperation*

Such words which need to be characterised will be referred to *target words* in this thesis. Using knowledge of English we can divide these words into classes. Most obviously, the forms can be divided into two groups: 3rd person present forms of transitive verbs and abstract, uncountable nouns. Imagine, though, that we know nothing about such possible syntactic categorisations in this language. We do, however, have access to a large corpus of text, and can find words that directly co-occur with them (their left and right neighbours) in that corpus. This is the sort of narrow context that has already been mentioned as typically used when a space reflecting syntactic similarity is sought. Table 4.1 shows the first 20 word triplets for the tokens *finds* and *cooperation* in the news portion of the British National Corpus (BNC, 2007). For the purpose of this example we collect 100 such contexts for each of the six target words in our vocabulary. Then we construct a co-occurrence matrix by counting the number of occurrences of a target word with some group of *context words*. For this toy example we will consider only three left contexts, the frequent words *the*, *and*, and *it*.

The following matrix, which will be called $\boldsymbol{C}$, records counts of the times a target word co-occurs with a context word to its left in the 600 examples extracted from the corpus. For example, *the education* is observed seven times, and $\boldsymbol{c}_{1,4}$ records that:

$$\begin{pmatrix} 0 & 0 & 0 & 7 & 8 & 11 \\ 5 & 3 & 2 & 1 & 3 & 9 \\ 2 & 11 & 10 & 0 & 0 & 0 \end{pmatrix}$$

Each word has been turned into a vector of numerical features (i.e. a column

but finds it	and cooperation from
he finds friendly	for cooperation between
who finds himself	s cooperation .
racing finds cagney	friendly cooperation ,
hamilton finds that	the cooperation of
rice finds out	multi-agency cooperation in
davies finds out	' cooperation ,
he finds this	seeking cooperation and
delahunty finds particularly	gulf cooperation council
she finds our	close cooperation with
but finds the	gulf cooperation council
and finds it	close cooperation with
bradbury finds jandel	regional cooperation (
he finds himself	east-west cooperation and
and finds it	, cooperation on
technology finds a	community cooperation always
swinburn finds it	and cooperation in
torode finds a	the cooperation took
now finds itself	and cooperation in
he finds he	the cooperation of

Table 4.1: *First 20 word triplets for tokens 'finds' and 'cooperation' in the news portion of British National Corpus*

of $\boldsymbol{C}$) which characterises the way that word associates with neighbours in the corpus. It can already be seen that $\boldsymbol{C}$ contains useful information for distinguishing between the two groups already mentioned. Both the left neighbours *the* and *it* are useful for distinguishing the nouns and verbs (for obvious reasons). However, there are also drawbacks. For example, these 2 cues to the same partition of the vocabulary are not unified into a single feature, and so the representation is not as compact as it might be. Also, the other feature word (*and* as a left neighbour) is not useful for distinguishing the two classes. Neither are *the* and *it* perfect features. *It* is less frequent before *finds* than before *requires* and *makes*, and *the* occurs before *cooperation* more than before *education* and *research*.

These types of problem are typically overcome in vector space models by the use of a dimension-reducing projection. A well-established method is to use *singular value decomposition* (SVD) to extract lower dimensional latent structure. The co-occurrence matrix $\boldsymbol{C}$ is decomposed using SVD into three matrices, i.e.:

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T \tag{4.1}$$

$\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices containing the left and right singular vectors of $\boldsymbol{C}$ and $\boldsymbol{D}$ is a diagonal matrix whose diagonal elements are the corresponding singular values. Applied to our toy example, this gives:

$$\begin{pmatrix} 0 & 0 & 0 & 7 & 8 & 11 \\ 5 & 3 & 2 & 1 & 3 & 9 \\ 2 & 11 & 10 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} -0.75 & 0.42 & 0.51 \\ -0.58 & -0.02 & -0.82 \\ -0.33 & -0.91 & 0.26 \end{pmatrix} \begin{pmatrix} 18.28 & 0.00 & 0.00 \\ 0.00 & 15.06 & 0.00 \\ 0.00 & 0.00 & 5.22 \end{pmatrix} \begin{pmatrix} -0.19 & -0.29 & -0.24 & -0.32 & -0.42 & -0.73 \\ -0.13 & -0.67 & -0.61 & 0.19 & 0.22 & 0.29 \\ -0.68 & 0.08 & 0.19 & 0.53 & 0.32 & -0.33 \end{pmatrix} \tag{4.2}$$

This is shown graphically in the upper part of Figure 4.1. The representations obtained in this way are more compact and more robust than the original co-occurrence vectors. The decomposition effectively combines the 2 useful cues mentioned above, and smooths out their imperfections: the second basis vector (2nd column of $\boldsymbol{U}$) gives the left contexts *the*, *and*, *it* the weights 0.42, -0.02, and -0.91 respectively. That is, this vector expresses the behaviour of not occurring after *the* and of occurring after *it*; it is non-committal about the *and* context, giving this a weight near to 0. The central row of $\boldsymbol{V}^T$ shows the weights used to mix this basis vector into word representations to reconstruct the original co-occurrence vectors: essentially this row combines and smooths the outer rows of $\boldsymbol{C}$, giving a smoother, less redundant and more robust representation than either in isolation.

The diagonal elements of $\boldsymbol{D}$ are conventionally sorted in descending order of magnitude: taking the first r columns of $\boldsymbol{U}$ and $\boldsymbol{D}$ and the first r rows of $\boldsymbol{V}^T$ gives matrices whose product is the best rank r approximation of $\boldsymbol{C}$, here denoted by $\hat{\boldsymbol{C}}$:

$$\hat{\boldsymbol{C}} = \boldsymbol{U}_r \boldsymbol{D}_r \boldsymbol{V}_r^T \tag{4.3}$$

For example in the toy example, the small 3rd singular value shows that the corresponding dimension is of little importance in reconstructing $\boldsymbol{C}$ (it accounts for about 14% of the variation in the data) and can be discarded, along with the corresponding singular vectors:

$$\begin{pmatrix} 1.83 & -0.21 & -0.50 & 5.57 & 7.15 & 11.88 \\ 2.08 & 3.34 & 2.79 & 3.27 & 4.36 & 7.61 \\ 2.93 & 10.89 & 9.75 & -0.72 & -0.43 & 0.44 \end{pmatrix} = \begin{pmatrix} -0.75 & 0.42 \\ -0.58 & -0.02 \\ -0.33 & -0.91 \end{pmatrix} \begin{pmatrix} 18.28 & 0.00 \\ 0.00 & 15.06 \end{pmatrix} \begin{pmatrix} -0.19 & -0.29 & -0.24 & -0.32 & -0.42 & -0.73 \\ -0.13 & -0.67 & -0.61 & 0.19 & 0.22 & 0.29 \end{pmatrix} \tag{4.4}$$

This is shown graphically in the lower part of Figure 4.1. The 2nd singular value shows that the second latent dimension accounts for 39% of the data's variance. The first singular value is of course largest, but the corresponding component is not very informative as it has a shape that is very similar to the

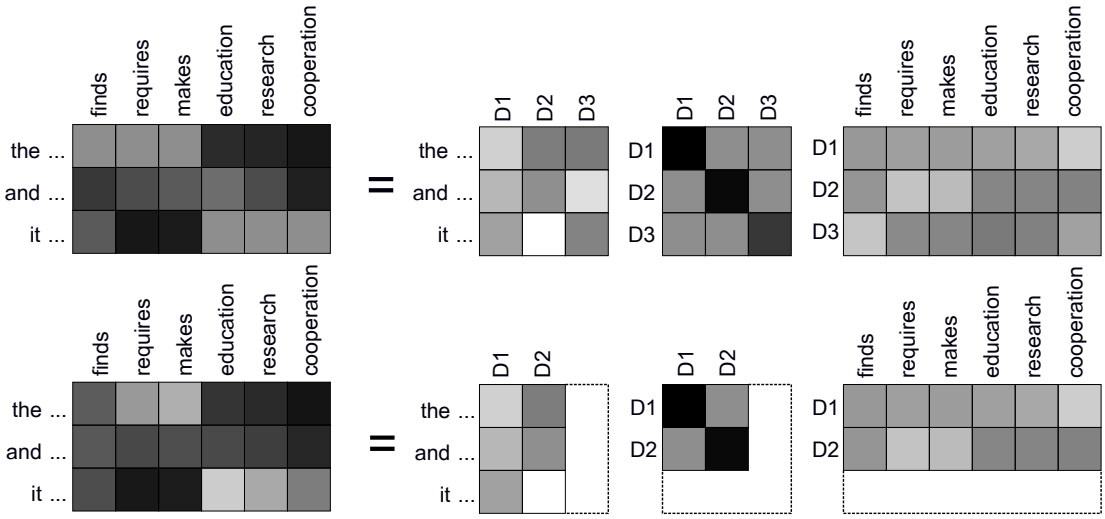


Figure 4.1: *Graphical toy example of the induction of word representations via singular value decomposition (a logarithmic grey-scale is used).*

average shape of the columns of $\boldsymbol{C}$; the first column of $\boldsymbol{U}$ correlates well with the sums of the rows of $\boldsymbol{C}$ (see Hu et al. (2003) and Widdows (2004, p. 184)). These effects could be removed by centering the data before performing SVD, but the sparsity of the co-occurrence matrix would be lost, together with the implementational advantages of that sparsity.

The matrix $\boldsymbol{V}_r^T$ obtained in this way can now serve as a sort of lexicon. A low-dimensional characterisation of a word's distributional behaviour can be looked up in the corresponding column of this matrix. A common choice is to weight the values in this representation by the singular values, and to use the columns of $\boldsymbol{D}_r\boldsymbol{V}_r^T$:

$$\begin{pmatrix} -3.54 & -5.38 & -4.47 & -5.81 & -7.70 & -13.40 \\ -1.93 & -10.04 & -9.11 & 2.93 & 3.30 & 4.42 \end{pmatrix}$$

As well as characterising types observed in the training data (i.e. the columns of $\boldsymbol{D}_r\boldsymbol{V}_r^T$), SVD also yields a linear transform that can be used to project newly constructed co-occurrence vectors into the same latent space, or to *fold them in*. Imagine that after building the space, two new words are seen:

investigation, *contains*

A sample of 100 examples of each is collected, and co-occurrences with the same left contexts that were used to build the vector space are collected in a matrix $\boldsymbol{Q}$, so that element $\boldsymbol{q}_{3,2}$ records the fact that *it contains* occurs 17 times in the sample:

$$\begin{pmatrix} 18 & 0 \\ 1 & 3 \\ 0 & 17 \end{pmatrix}$$

The equation to transform these newly acquired co-occurrence count vectors (columns of $\boldsymbol{Q}$) into lower dimensional representations in the latent space (columns of $\boldsymbol{L_r^T}$) can be found from Equation 4.3, when it is considered that we wish the same relation to hold between $\boldsymbol{Q}$ and $\boldsymbol{L_r^T}$ as holds between $\hat{\boldsymbol{C}}$ and $\boldsymbol{V}_r^T$. $\boldsymbol{U}$ is an orthonormal matrix so that $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$:

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{U}_r\boldsymbol{D}_r\boldsymbol{L_r^T} \\ \boldsymbol{U}_r^T\boldsymbol{Q} &= \boldsymbol{U}_r^T\boldsymbol{U}_r\boldsymbol{D}_r\boldsymbol{L_r^T} \\ \boldsymbol{U}_r^T\boldsymbol{Q} &= \boldsymbol{D}_r\boldsymbol{L_r^T} \end{aligned} \tag{4.5}$$

Projecting newly found items into the existing latent space merely involves multiplying the transpose of $\boldsymbol{U}_r$ by the count vectors of those new items. Note that this produces the singular value weighted representation mentioned above. In the current example, that gives the following compact representations of the two new words:

$$\begin{pmatrix} -14.03 & -7.37 \\ 7.56 & -15.48 \end{pmatrix}$$

This ability to place new objects in a low dimensional space that has already been estimated might be useful in models of phenomena where it is expected that novel items will frequently be encountered (such as a model of word types). In models of utterances or documents, where it might be expected that each new item encountered is of a novel type, it is crucial. This is discussed further in Section 4.2.

The conventional way to compare representations of items in a VSM is to compare whole vectors. Typically, cosine similarity is used. This measure reflects the size of the angle between two vectors, and so ignores magnitude (which reflects target word frequency). It is 1.0 when vectors share an identical direction, 0 when they are orthogonal, and -1 when their directions are opposite. One benefit of this measure is that no normalisation of vectors is required: words having very different frequencies but identical distributions will have vectors of different lengths, but their cosine similarity will be 1. See e.g. Manning and Schütze (1999, §8.5.1) for a discussion of this and other measures commonly used in vector space modelling.

However, in the work presented in this thesis, comparison of vectors is performed on an element-by-element basis, in the univariate splits of the vector space by *decision trees*. For this reason, length normalisation must be applied when items to be compared have vectors of very different lengths (see below, Section 4.3.2). This has been ignored for the purposes of this toy example, as the vectors here are of similar enough length for demonstration purposes. In this toy VSM, the information from 2 cues has been combined and smoothed; a split on the 2nd dimension at approximately 0 will separate these words into the classes noun and third person present verb. Essentially a question like *Is the value of dimension 2 less than 0?* is equivalent to *Does this word tend to occur after* the *and not after* it*?*; it is a compound question about a word's behaviour over multiple contexts, expressed by a linear rather than Boolean mechanism. Dimension 2 has a simple interpretation due to the toy nature of this example; such compound features will of course consist of many more terms in real-world examples, with much greater variation among the coefficients weighting those terms.

4.2 Vector Space Modelling Applied to TTS

4.2.1 Serial Tree Building: Shortcomings

Chapter 3 presented two preliminary attempts at inducing phoneme and letter features during the training of TTS systems, using *serial tree building* techniques. Those attempts met with considerable success: in the experiment described in Section 3.4, for example, phonetic categories discovered wholly automatically reduced synthesis distortion on a held-out set to a level comparable to that achieved by manually-specified expertise. The strength of this method is that it uses the primary data to find classes of unit that are by definition relevant to the acoustic modelling of that data. However, attention has already been drawn to some of the shortcomings of the methods used. Most obviously, the methods group objects observed in the primary corpus into classes, but their power to generalise beyond what is observed in that small data set is limited. When dealing with sets of items with only a few members, such as context-independent letters or phonemes, this is not problematic. The sets of phonemes of a language or characters of an alphabetic script are small enough that it is reasonable to expect that a number of examples of each of their members will be observed in a primary corpus of modest size. They are stable enough that it is reasonable to assume that novel phonemes and letters will not come into existence in the life-time of a

TTS system. Thus the third experiment of Chapter 3 introduces a technique that is useful, as it deals with determining classes of context-independent phonemes.

When dealing with sets of higher cardinality, however, such as letters in a 7-letter context window (as in Experiment 2 of the previous chapter), problems arise. Even though only a fraction of the 26^7 7-letter sequences of the lower-case English alphabet are linguistically plausible, the number that can be encountered in normal text is vastly greater than what is seen in a primary corpus for TTS. If the c. 32 million words of news-text in the British National Corpus (BNC, 2007), for example, are stripped of non-alphabetic characters, and all upper-case characters converted to lower-case, the number of 7-letter within-sentence sequences that are observed is greater than 12 million. Obviously, it is only reasonable to expect to observe a small fraction of these in a TTS primary corpus. It has already been observed that the method proposed in Experiment 2 of the previous chapter produced a significant improvement only on test sentences with words observed in the training set. When tested on sentences with words never before seen by the system, the improvement was slight and not found significant.

This problem becomes very marked when textual-linguistic analysis moves from subsyllabic units such as letters and phonemes to units on higher-level tiers, such as words, phrases, utterances, paragraphs, etc. To take words as an example, there may be in a primary corpus e.g. 5,000 surface forms of words which linguistic knowledge – in the form of a POS tag-set, should one be available – suggests should map to 40 or 50 classes. A POS tag-set and conventional secondary corpus of text labelled with these tags allows the construction of a classifier which enables generalisation over the many surface forms of the training set, and beyond them to ones not seen in the primary corpus but encountered at synthesis time. Any method of generalisation that relies on observing units in the primary corpus will obviously suffer with word-like units. The same obviously goes for units like phrases and utterances, whose representatives at synthesis time can reasonably be expected to be entirely novel types. The importance of being able to characterise units at these levels above the phoneme is clear, however, from results of the first experiment of Chapter 3, which showed that features defined at tiers above that of the phoneme have a positive impact on the naturalness of synthetic speech generated by a system (see Section 3.2.4).

4.2.2 Distributional–Acoustic Modelling: Vector Space Models and Decision Trees

In the remaining part of this thesis, a more general method for allowing decision trees to generalise over surface forms is developed. This method is designed to exploit the primary corpus in order to find partitions of sets of linguistic/textual objects that are directly relevant to acoustic modelling, just as the serial tree methods already outlined do. However, unlike the serial tree methods, it is designed also to exploit the availability of plain unannotated text data whose coverage of forms is much greater than can be achieved in a primary corpus. This method is based on vector space models of language, which have already been introduced. Crucially, the units to be modelled can be letters, phonemes, or ones at higher levels of analysis, such as words, morphemes, phrases, etc. Continuous-valued representations of textual/linguistic objects are learned from text using vector space models at various levels of analysis. These representations provide prior knowledge of the sorts of divisions of textual/linguistic units into classes that it might be useful to make *before any acoustic data have been observed.* Plain text is typically more readily obtainable in vast quantities than text aligned with audio; much richer distributional information for the textual/linguistic objects in the primary corpus will be available in the tertiary corpus. With large sets (such as sets of words) the number of types seen in the tertiary corpus will be greater than in the primary one, meaning that the resulting system will be able to generalise to account for *unheard* (but not *unseen*) objects. This distributional knowledge is introduced to decision tree-based clustering in the form of features that generalise over a possibly vast number of surface forms. Measurements in the vector space are made available to the system when synthesis models are built using decision trees. A vector space constrains the ways in which surface forms can be partitioned by decision trees and so guides tree-building, but the possible partitions are still numerous, and no small set of categories is dictated by the distributional analysis. This is because the vector space model maps linguistic objects to points in a continuous space, rather than to discrete categories. The vector space is partitioned by decision trees with 'supervision' provided by the acoustic signal in TTS system-building tasks; these tasks are for example the building of duration models, generative models of acoustic features such as spectral envelope and fundamental frequency, and models for predicting pauses from text.

4.2.3 Strengths of the proposed approach

The most obvious strength of vector space models is that they allow linguistic/textual objects to be characterised in an unsupervised manner. They are therefore cheap to acquire, relying only on the availability of plain text resources and the ability to tokenise this resource (relatively trivial in the case of orthographies that use whitespace to demarcate words). Importantly, this contrasts with methods conventional in TTS such as POS tagging, in that no annotation by human labellers is assumed by the framework.

The development of SVD algorithms specially designed to cope with large sparse matrices or data streams as input means that extremely large amounts of text can be used to induce the word representations. The use in the current work of an incremental SVD algorithm which requires only a single pass over the data means that the amount of text on which the vector spaces are based is in theory unlimited. For the work presented in this thesis, an implementation based on Brand (2006) was used.[1] More recent versions of the package have seen the introduction of distributed algorithms for single pass SVD (Řehůřek and Sojka, 2010; Řehůřek, 2011), which could further ease the use of vast datasets. Experiments with a corpus of 25 million words are reported in Chapter 7; although this is small by the standards of recent work in NLP, it is larger than the largest available corpus that is hand-annotated with parts of speech in the vast majority of the world's languages.

Because the vector space model performs unsupervised learning, the availability of labelled data in a given domain provides no constraint on training. Consequently, text can be collected from the domain in which the TTS system will be required to operate. It is shown in Chapter 6 that a vector space model trained over words of news-text from the *Wall Street Journal* clearly reflects the newspaper's financial domain, to the extent that certain dimensions of the space characterise vocabulary associated with stock prices. It is reasonable to expect that if the domains of the training set for a vector space model and the end synthesis task are more closely matched, the representations that emerge from the data will be all the more suited for synthesis.

Again, because they result from unsupervised learning, vector space models are ideally suited to exploiting user input. User input in the simplest sense can mean the text input into a TTS system, or the log of previous inputs to a system. It is worth stressing this point: extra text is the only additional data that is

[1] Part of version 0.5.0 of the package *Gensim* (Řehůřek and Sojka, 2010)

guaranteed available to a TTS system – by definition – as the system is run. The data are *free*: they don't require explicitly asking a user for feedback. They are acquired purely by virtue of a TTS system being run. Moreover the data are by definition drawn from the domain (or at least from one of the domains) in which the specific user wishes to generate speech. Different ways exist of exploiting newly-acquired text with a vector space model. When the space is a model of word types, for example, it is possible to use the transform estimated for the model to project newly encountered word-types into the latent space (see the example of *folding in* in Section 4.1 above, and the discussion of mixed and dynamic model types below in Section 4.2). This is especially useful for previously unseen domain-specific words which are encountered numerous times so that rich context vectors can be accumulated and then folded into the model.

Another advantage of the proposed approach is that because the VSM represents objects using continuous values, a hard classification of objects is delayed. Different sets of word classes are likely optimal for different NLP tasks, and also perhaps for different tasks that must be performed for TTS conversion. For example, divisions that are useful for some other NLP task (e.g. full syntactic parsing) might be irrelevant as far as the acoustic realisation of words is concerned, and thus for TTS. One of the main differences between the approach described here and the method that inspired it (Schütze, 1995, see Section 4.3.2 below) is that here, continuous-valued features are used directly, and the final clustering stage is omitted. The final rediscretisation stage is necessary in Schütze (1995) because the evaluation presented there is based on the overlap between induced and reference categories, a type of evaluation not used here. But working with continuous-valued features directly has several advantages (theoretical and practical) over conventional discrete POS-like sets. The key theoretical benefits of using continuous-valued features are firstly that they make no assumptions about the granularity of relevant categories (in the way that using a standard tag-set does), and secondly, that they make it possible to encode gradient phenomena and at the very least allow us to avoid having to make a hard decision about where to split along the continuum of such a phenomenon until strictly necessary.

It is shown in this thesis that it is possible to apply the VSM in a unified way to multiple levels of textual analysis relevant to TTS. In Chapter 5, the model is applied to phonemes and letters, in Chapter 6 to words, and in the final section of Chapter 7, the model is used to characterise whole utterances. End-to-end systems incorporating these levels of analysis are outlined in Chapter 8, where techniques developed initially on English are applied with little extra effort to

two additional languages.

4.2.4 The Vector Space Models Used in this Thesis

To make discussion concrete at this point, the three sorts of model that are built to characterise letters, words and utterances in Chapter 5, 6, and 7 respectively will be considered.

Letter- and word-level models To acquire representations for letters and words, one vector is assembled for each of the m letter or word types in the tertiary corpus. Co-occurrence is tallied with n letter/word types as left and right neighbours, and this value must be determined, in the range $[1, m]$. Where m is small (e.g. for letter models), n is set equal to m. A $2n \times m$ co-occurrence matrix, $\boldsymbol{C}$, is assembled such that $\boldsymbol{c}_{i,j}$ records a count of the number of times the i^{th} feature letter or word occurs as the left-hand neighbour of the j^{th} target letter or word type, and $\boldsymbol{c}_{i+n,j}$ records a count of the number of times the same (i^{th}) feature letter or word occurs as the *right*-hand neighbour of the same (j^{th}) target letter or word type.[2] The raw co-occurrence matrix is then decomposed by a singular value decomposition as already described, and the resulting matrices truncated. As already mentioned, the diagonal elements of $\boldsymbol{D}$ are conventionally sorted in descending order of magnitude: taking the first r columns of $\boldsymbol{U}$ and $\boldsymbol{D}$ and the first r rows of $\boldsymbol{V}^T$ gives matrices whose product is the best rank r approximation of $\boldsymbol{C}$. We will denote the matrix derived from $\boldsymbol{D}\boldsymbol{V}^T$ by taking its first r rows as $\boldsymbol{D}_r\boldsymbol{V}_r^T$, an r by m matrix whose columns are vectors summarising the interactions of the m letter or word types of the corpus with their neighbours. The discrete symbols of letter and word types from the corpus data are thus converted into continuous values giving the coordinates of points in a r-dimensional space. We wish to work with the values of individual elements of these vectors; before this can be done, however, we need to reduce some target letter and word frequency effects encoded in these values. One source of such effects is the discarding of low-valued singular values. $\boldsymbol{V}^T$ is orthonormal and so its columns are vectors of unit

[2]Note that the notation used in this section is different from the equivalent discussion in Watts et al. (2011). There, $\boldsymbol{C}$ is a word-by-context co-occurence matrix; this emphasised the method's origin in work like Schütze (1995). Here it is replaced by its transpose, a context-by-word matrix. This is to better reflect the most recent implementation used, but also to draw attention to continuity with the utterance space described below. There, as conventional in Latent Semantic Indexing (Section 4.3.2), columns of $\boldsymbol{C}$ represent documents / utterances. If $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$ then $\boldsymbol{C}^T = \boldsymbol{V}\boldsymbol{D}\boldsymbol{U}^T$, so that the two formulations are equivalent if columns of $\boldsymbol{V}^T$ are used instead of rows of $\boldsymbol{U}$.

length, but the truncated version of it, $\boldsymbol{V}_r^T$, where only r rows are retained, has columns of greatly differing vector lengths. Vector length is correlated with target word frequency. To counter such effects, unit length is imposed on the columns of $\boldsymbol{D}_r\boldsymbol{V}_r^T$, resulting in a matrix that will be denoted $\boldsymbol{D}_r\boldsymbol{V}_r^{T'}$. The elements of these columns can then be queried directly in isolation by modules that rely on the features (e.g. decision trees).[3]

It should be emphasised that the VSMs that are built in this thesis for words and letters are designed to characterise word and letter *types* rather than individual *instances* of words and letters. The subword-unit model is like that built in Experiment 3 of Chapter 3, rather than that of Experiment 2: no characterisation is done on the basis of a unit's context, and the 2 instances of *e* in the word *these*, for example, will be assigned the same representation by this model. Similarly, the word model will not distinguish between forms which would need disambiguation by context to be assigned the appropriate part of speech tag. For example, the word *winds* in both the fragments *It winds up* and *The winds are* will be assigned the same representation. There are two motivations for not seeking a model that can disambiguate instances on the basis of context. Firstly, Lamar et al. (2010) convert the output of a disambiguating tagger (which has been learned in an unsupervised way) so that for each word type, the tag most frequently assigned to that type by the tagger is *always* assigned that type. They show that converting the tagger to a type-based tagger in this way does not worsen its performance. Secondly, the word representations produced as described here are not an end in their own right, but are rather designed to be used on tasks where a context window over the type representations of neighbouring words means that disambiguation can be performed implicitly if it is beneficial to the task. Exactly the same type of model is used for letters because of a desire to treat analysis on each linguistic/textual tier as uniformly as possible.

If a static model is to cope with open sets, it is obvious that a method of dealing with unseen units must be devised. The method used in Chapter 6 is to learn a representation for a special symbol which is used for unseen words at run-time. For this, tokens in a 1% portion of the data whose types are not seen in the other 99% are rewritten with this *unseen* symbol. Different methods using hard thresholds for determining which words will contribute to the *unseen* model

[3]Both the scaling by singular values (using $\boldsymbol{D}\boldsymbol{V}^T$ instead of $\boldsymbol{V}^T$, in which we follow the original description of term-term comparisons in Deerwester et al. (1990)) and subsequent scaling to unit length are details that were omitted from the description of the algorithm given in Watts et al. (2011). Differing vector lengths of the columns of truncated $\boldsymbol{V}^T$ are also noted in Horn and Axel (2003), where the same solution is used.

are found to give more consistent results in Chapter 7.

Utterance-level models The characterisation of phrases and utterances, on the other hand, can obviously not proceed in the same way – it is unreasonable to expect types of these units to be seen more than once in training data, or for seen types to recur at testing time. For this reason, the type of model that is built for utterances in this thesis (and which would also be appropriate for intermediate units such as phrases or metrical feet) is different, and is much more like a classic Latent Semantic Indexing-type space (see Section 4.3.2 below). That is, it provides a model of documents (sentences) which is based on elements internal to the unit to be characterised (words within the sentence), and is dynamic (i.e. the linear transform itself is stored, and not a lexicon-like list of transformed representations for items already encountered).

A word–utterance matrix $\boldsymbol{C}$ is compiled from the tertiary corpus, where $\boldsymbol{c}_{ij}$ is a count of vocabulary item i in the j^{th} utterance of the corpus. No stop words are excluded from this vocabulary, contrary to standard practice in Information Retrieval: the representation sought here is ideally one that characterises utterances in terms of their pragmatic role in a text, rather than their semantics, and it seems clear that certain common words (*However*) and non-word tokens (*?*) reflect their sentence's role in a text (cf. the similar case of building word vectors for sentiment analysis in Maas et al. (2011)). All words occurring more than once in the corpus were retained in the model. Term-frequency inverse document-frequency (TF-IDF) weighting is applied to the raw co-occurrence matrix, and SVD is performed; r dimensions of the latent space are retained.

At synthesis time, newly encountered utterances are converted into vectors of counts which are weighted using TF-IDF, and projected into the latent space as outlined above to assign them an r-dimensional representation.

4.2.5 Design Choices

A great many design choices must be made at each step of the construction of a VSM, and the types of VSM that have just been described and which were built for the work presented in this thesis by necessity represent only a few points in the vast space of possible systems. Some mention is now made of factors that must be considered in building VSMs for TTS.

Model Type and Context Type

When a linguistic or textual tier has been selected for analysis, and a VSM is to be constructed to characterise the units occurring on that tier, two important decisions that must be made relate to what will here be called the *model type* and the *context type*. These decisions are closely connected, and result in the different types of model mentioned above being built, for letters and words on one hand and for utterances on the other. There are many other factors that can be varied in the creation and use of these spaces, one of which – transformation type – is also mentioned briefly below.

Model Type Estimating a dimensionality-reducing transform from an observation space to a latent space results in an embedding of the training data in the latent space ($\boldsymbol{D}_r\boldsymbol{V}_r^{T'}$), which was used in the example above as a sort of lexicon from which low-dimensional representations of words can be looked up. This will be called the *lexicon model type* or alternatively *static model type*, as representations that can be assigned at run-time are limited to be members of the fixed inventory of types seen during training. Another output from training, of course, is the transform itself ($\boldsymbol{U}_r^T$). In the static model type, this transform is simply discarded after training. In what will be called the *dynamic model type*, or *projection model type*, conversely, this transformation is kept and $\boldsymbol{D}_r\boldsymbol{V}_r^{T'}$ can be discarded. When units are to be assigned a representation, the contextual representation of the observed unit is projected into the latent space using the transform. A *mixed model type* is also envisaged, where a lookup table of representations of units in the tertiary corpus is retained, and also the transformation. Some criterion is defined that – when a representation is to be assigned to a unit – determines whether a representation from the lookup table should be used, or whether the transformation should be invoked. Most trivially, this criterion would be the presence or absence of the unit in the lookup table (i.e. whether the unit was seen during training on the tertiary corpus). It should be noted that the model types differ only in the way they are stored and deployed – the different model types imply no difference in training.

Context Type Another closely-related design choice is what context units are chosen to characterise target units: most critically, whether the context units are *internal to* or *external to* the target units. For example, a word might be characterised externally, by neighbouring words as in the toy model given in

Section 4.1. Alternatively, a word could be characterised by units internal to it, such as morphemes or morpheme-like units. The advantage of using such morphological information is clear, for example, in the the list of target words used in Section 4.1: *find-s*, *require-s*, *make-s*, *educa-tion*, *research*, *coopera-tion*. Here, the nouns and verbs can be well separated by the presence or absence of either the morpheme *-s* or *-tion*. As in the case of model type, a third option is a mixed type – considering both internal and external contexts.

How a model type or context type is chosen for a vector space model will depend on the nature of the phenomenon to be modelled, and the two factors are closely bound to each other. These two axes of variation are depicted schematically in Figure 4.2. The symbols L, W and U are placed in the relevant boxes of Figure 4.2 to represent the three types of VSM built in the work presented in this thesis – of letters, words and utterances, respectively.

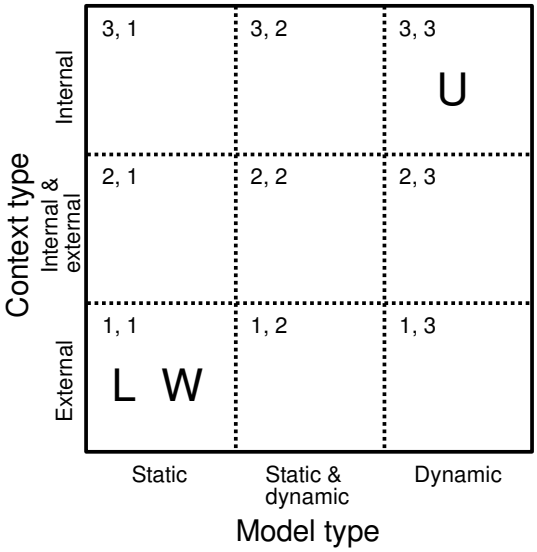


Figure 4.2: *Two dimensions of a space of possible vector space models.*

Other placements of the three symbols in the figure would correspond to system configurations not explored in this thesis. Many of the configurations suggested by the figure are attractive, and are left unexplored because they are too numerous rather than because they are unpromising. For example, using a model of letters with the configuration implied by position (1,3) of Figure 4.2 would result in a model capable of sensitivity to the difference between the two *i*'s and the two *g*'s in the word *nightingale*. A word model of the configuration implied by (2,2) would require some method of morphological segmentation, but the dynamic part of such a model would be capable of handling unseen words perhaps better

than the simple *unseen type* method described in Section 4.2.4. Such an approach would certainly give superior results in a language more morphologically rich than English. An utterance model of the configuration implied by (2,3) might be suitable where the speech corpus used is recorded with real communicative intention. In such data, unlike conventional TTS corpora, utterances will have a genuine communicative function which may be discernible from e.g. words or phrases in neighbouring utterances. These are just a few of the many possibilities suggested by Figure 4.2.

Dimensionality-reducing Transformation Another design choice relates to the way low-dimensional representations are obtained from raw co-occurrence vectors. Throughout this thesis, singular value decomposition is used to obtain these. As explained in Section 4.3, the use of this matrix factorisation technique for text analysis has its roots in Information Retrieval. Many other techniques have since been proposed for accomplishing dimensionality reduction in Information Retrieval. For example, Independent Components Analysis is shown by Isbell and Viola (1998) to outperform SVD on Information Retrieval tasks (see Section 4.3.3). These techniques could also be applied to inducing the sorts representation applied in this thesis to TTS, no doubt with the possibility of improved performance. The aim of this thesis, however, is to propose and test a general framework for incorporating unsupervised learning in TTS systems, rather than to exhaustively explore all possible design choices in search of the best performing system. SVD is used here as a tried-and-tested method.

In all experiments reported in this thesis, vector spaces are exploited using tree-based methods (CART and decision tree induction for state-tying). This choice of machine learning technique is pragmatic, as it allows vector spaces to be simply integrated into state-of-the-art systems which already rely on decision trees for the construction of their acoustic models. The choice of CART for front-end techniques is due to the fact that, as well as being a well-established method for front-end tasks in TTS systems, it allows vector spaces to be exploited in a way that is unified between the front and back-ends of the synthesiser.

There is very little in previous work that suggests that univariate splits of vector spaces such as the ones used here should be especially useful for the tasks attempted. A more common approach is to consider only distances between vectors in the space, and not the absolute positions of those points. Despite this, the experimental results presented in later chapters show that this approach produces useful features. However, it is to be expected that cleverer exploitation of vector

spaces than that allowed by the univariate splits of conventional decision trees will give better results. This would come at the price of having to make fundamental changes to the state-of-the-art TTS framework used as the foundation for these experiments.

4.3 Previous Work on Vector Space Models of Language

Past work on vector space models in language processing is here reviewed in greater depth than in the above introduction. First, the original vector space model as formulated by Salton et al. (1975) for IR is described. Then the extended application of variants of this basic model to the induction of semantic, then syntactic, word features is summarised. Next, an influential development of the basic VSM called Latent Semantic Indexing which underpins the work of this thesis is described, and its application to the same tasks. Lastly, work using other means of dimensionality reduction, and work using probabilistic topic models is mentioned. This division is somewhat arbitrary, and doesn't for example always respect chronological order. Properties of a few of these systems are summarised in Table 4.2 for convenience.

4.3.1 Salton's Vector Space Model

Information Retrieval

Salton et al. (1975) describe the classic vector space model for Information Retrieval. In it, each document is identified by one or more index terms. Where n different index terms are used over a collection of documents, each document is represented as a vector of length n, each of whose elements encodes the presence or absence or a particular term from a document (in the case of presence, possibly modified by some weighting scheme). Salton et al. (1975) describe measuring the distance between documents in the following way: vectors are normalised to unit length, and the distance between two documents is then the distance between the two points representing them on the unit sphere.

Salton et al. (1975) mention the simple *term frequency weighting*, where a given term-element in a document vector simply records a count of the number of times that term appears in that document. Also mentioned is the notion of *inverse document frequency* (IDF). The problem with simple term frequency weighting

is that some terms might be frequent across all documents of a collection, and thus will have little power to discriminate between the documents. IDF scales down the contribution of terms which occur in many documents. The *document frequency* of a given term t, $df(t)$, is a count of documents which contain that term. *Inverse document frequency* of a term ($idf(t)$) is then:

$$idf(t) = \log \frac{N}{df(t)} \tag{4.6}$$

where N is the number of documents in the collection.

The VSM model is used for information retrieval by treating a user's query as a pseudo-document, and finding the documents (or clusters of documents) in the collection whose distance to that 'document' is the smallest.

Distributional Semantic Features

The so-called *hyperspace analogue to language* that is presented in Lund et al. (1995) and Lund and Burgess (1996) uses a formal model similar to the one described by Salton to extract semantic representations of words from text. The motivation for the work is different: the authors wish to construct a cognitively plausible model of semantic memory which is then tested by being used to predict human reactions on lexical priming tasks. Some details of model construction are also different. The model described is induced from 160 million words of English Usenet text; the 70,000 most frequent words are chosen as target words, and also provide an initial set of context words. Target words take the place of the documents of Salton's model; the descriptors of a target word are the context words with which it co-occurs in a moving window (instead of index terms). The window used extends 10 words to the left and right of occurrence of the target word. Weighting is applied such that the weight for a context word is inversely proportional to the context word's distance from the target. The window is also directional: occurrence of a context word to the left of the target are stored as a descriptor that is separate from that for the same context word occurring to the right. This results in 140,000-dimensional word vectors. The semantic distance between words is then computed by Minkowski metrics on the relevant words' vectors normalised to unit length.

Such large word vectors are obviously unwieldy; a method is described of selecting a small set of descriptors by computing the variance of each of the 140,000 descriptors and keeping only the top k descriptors with highest variance.

Variance falls off sharply after the first 100 descriptors, and is very low after 200. It is reported that using only 200 descriptors provides very similar results to those using 140,000 dimensional vectors. It is mentioned that the model is influenced by Schütze (1992) (see section 4.3.2 below), and SVD is mentioned as a possible alternative for dimensionality reduction.

Distributional Syntactic Features

Finch and Chater (1992) describe a VSM for discovering syntactic categories from plain text; the motivation appears to be linguistic or cognitive: an attempt to determine empirically to what extent linguistic categories are learnable without a knowledge of syntactic rules, only exposure to the sequences output by those rules and suitable notions of context and similarity. The co-occurrence matrix is built from 40 million words of English text; word vectors are acquired for the 1000 most frequent words, and their co-occurrence with the 150 most frequent words is tallied. Four context positions are used, including the two neighbours of each instance of a target word, resulting in 600-dimensional feature vectors. Spearman Rank Correlation Coefficient (normalised to the range [0, 1]) is used to measure distance between the raw context vectors; this metric is used in a hierarchical cluster analysis of the 1000 target words to produce a similarity dendrogram that can be cut at various levels of cluster granularity. The authors also mention an implementation using the same contexts as a neural rather than a statistical model – a Kohonen self-organising map. However, they observe that the output of this model, though promising, does not present the same possibility as their statistical model of tuning the depth at which the dendrogram is cut to get a set of classes that is as similar to the gold standard set as possible.

Redington et al. (1998) use the same basic model as Finch and Chater (1992), but motivate the experiments more explicitly, and report objective evaluations both on the basic model and on the model when various aspects of its design are altered. The work is intended as an investigation into the extent to which distributional information can act as a cue to acquisition of syntactic category; the authors stress that distributional information is one among several such possible cues (others being extra-linguistic context, phonological cues, prosodic cues, and innate knowledge of syntactic categories). They consider distributional cues to enjoy 'no theoretical primacy', but given the availability of electronically stored text, they are the easiest to investigate by computer simulation. Aspects of the model which are varied in the experiments include, for example, the extent and

direction of context window used. Many conditions varied are intended to test whether the model is meaningful as a model of first language acquisition.

4.3.2 Dimension-Reduced Vector Space Model: Latent Semantic Indexing

Information Retrieval

Deerwester et al. (1990) introduce the idea of representing a document–term matrix by a low-rank approximation of it, calling the technique *Latent Semantic Indexing* (LSI)[4]. The motivation is due to two characteristics of terms that present problems for the classic VSM: synonymy and polysemy. That is, on one hand there are numerous ways to put a concept into words and so literally matching a user query in terms of the surface form of its words might be problematic. Conversely, terms can have multiple meanings, so a literal matching scheme will find documents that share terms with but are actually unrelated to a user query. The solution is to uncover latent semantic structure of documents and queries. This is accomplished by applying SVD to the term–document matrix, and retaining only the dimensions of the transformed space with the highest eigenvalues. The approach is described in factor analytical terms:

> [...] by virtue of the dimension reduction, it is possible for documents with somewhat different profiles of term usage to be mapped into the same vector of factor values (Deerwester et al., 1990).

The factors:

> [...] may be thought of as artificial concepts; they represent extracted common meaning components of many different words and documents. Each term or document is then characterized by a vector of weights indicating its strength of association with each of these underlying concepts. [...] The meaning representation is economical, in the sense that N original index terms have been replaced by the $k < N$ best surrogates by which they can be approximated (Deerwester et al., 1990).

However, unlike in factor analysis, it is emphasised that no explicit interpretation is sought for the underlying factors (via e.g. rotation or reduction to very few

[4]Note that the same technique is also often referred to as *Latent Semantic Analysis* (LSA).

dimensions). The SVD is rather a means to escape the 'unreliability, ambiguity and redundancy of individual terms as descriptors'. Individual elements of the low-dimensional representations obtained are not examined. Rather, the overall similarity of vectors is measured using *cosine similarity*, as described on page 66 of this thesis.

Deerwester et al. (1990) cite earlier work, such as that of Baker (1962) which proposes the adoption of latent class analysis – developed in sociology – for IR, and that of Borko and Bernick (1963) which suggests the use of factor analysis for IR.

The LSI approach to Information Retrieval is the most obvious precursor the utterance space presented above (see Section 4.2.4). As already mentioned, the documents of IR become sentences in the case of the utterance space. Also as already mentioned, no stop-word removal is performed in the case of the utterance space, contrary to usual practice in IR.

Distributional Semantic Features

Schütze (1992) describes the construction of what is termed *sublexical space* and its use on the task of (pseudo-)word-sense disambiguation (WSD). Spaces are built from large corpora of news-text, although precise details of their construction are scarce. Word co-occurrences are counted using a fixed-length window. Window lengths defined in (e.g. 1000) characters, 'because few long words are as good as (or even better than) many short words which tend to be high-frequency function words.' However, whole words are used as contexts; the contexts are apparently non-directional (i.e. the order of the co-occurrence is not encoded in context vectors). It seems that several thousand words are used as context words for the construction of the space.

Two different sets of WSD experiments are described. In the first, hierarchical clustering using cosine distance between vectors is used to find sense clusters in the space, and instances of ambiguous words are assigned to the nearest cluster. In the second set, Canonical Discriminant Analysis is used to find combinations of the dimensions of the word space that maximally separate pseudo-ambiguous words with their labelled senses. Using the cosine distance between whole vectors and finding linear combinations of the elements of those vectors represent quite different ways of using the word space, and the analysis presented in the second case yields some interesting results. Canonical Discriminant Analysis is used for classification by finding a combination of vectors' values to project them onto one

dimension; the best point on this line to separate distinct sense classes is then determined.

The effect of using different context window sizes are tested, and also the effect of combining different numbers of the top r dimensions resulting from SVD. Performance improves as more of these dimensions are added, and the trend of the results suggests that adding further dimensions (if they had been retained from the decomposition) would give further performance improvements. Inspecting the weights estimated in Canonical Discriminant Analysis reveals an interesting property of the space: that the importance of dimensions of the space for a given WSD pair does not seem at all positively correlated with the size of their singular value; if anything, there appears to be a negative correlation. Also, very different sets of dimensions are chosen for disambiguating different pairs of words. There appears to be considerable redundancy in the vector representations: the system is robust against the removal of various sub-sets of dimensions. This leads to the claim that:

> [...] the vector representations have the key properties of the distributed representations characteristic of parallel distributed processing (Schütze, 1992).

It also leads to the choice of term *sublexical space* by analogy with similar terms in connectionism.

A test is also made (using the clustering WSD set-up) of the extent to which the use of SVD influences results. Extreme differences among raw counts are dampened with a square root function, then clustered as before without the use of SVD; it is found that SVD does not influence performance for this task. The SVD is motivated instead by the compactness of the representations it allows.

Schütze (1993) describes a related approach, which is again tested on the WSD task. The construction of the space described there is very intricate. Counts are made over word fragments (letter fourgrams) rather than whole words to alleviate sparsity issues. Fourgrams are filtered in various ways to exclude redundancies and uninformative sequences. This results in a set of 5000 fourgrams. A co-occurrence matrix of target fourgrams and their left contexts is then produced; the context window is 200 fourgrams wide. SVD is applied, resulting in 97 dimensions for each fourgram type. Next, word features are induced. For each of c. 54,000 lemmas, feature vectors are assembled as follows: for each occurrence of a target word, for each fourgram within a window of 1001 fourgrams around the word occurrence, the relevant fourgram representations (computed in the previous step)

and normalised and summed. This sum represents the target word.

Bellegarda et al. (1996) describe the application of LSI to the clustering of word types for language modelling. The word representations obtained using conventional LSI (where words are characterised by the documents in which they occur) are clustered into 500 clusters. The word clusters are intended for use in class-based language modelling, where the semantic nature of the clusters is predicted to be beneficial.

Distributional Syntactic Features

The procedure for word-feature discovery described above in Section 4.2.4 is closely based on the first of three POS induction methods described in Schütze (1995) (where it is called *induction based on word type only*). Briefly, a matrix $\boldsymbol{C}$ is created where $\boldsymbol{c}_{ij}$ records the count of target word i in context j in a corpus. The contexts used in Section 4.2.4 are based on those used by Schütze (1995): counts of the 250 most frequent words as left and right neighbours. This matrix is factorised with singular value decomposition and the resulting matrices truncated as in Section 4.2.4 to yield a lower-dimensional representation of words. Here, the procedures outlined in Schütze (1995) and Section 4.2.4 diverge. In Schütze (1995), the discovered space is re-discretised by means of clustering the points representing word-types into a pre-specified number of clusters (using cosine similarity between vectors as a similarity criterion). This is due to the goal of that work: the induction of a set of symbols analogous to a POS tag-set. In 4.2.4, the final clustering stage is omitted, and instead the undiscretised space derived from two of the truncated matrices resulting from the decomposition is used directly, as described. Some advantages of using continuous-valued representations over a discrete clustering were given. The model of Schütze (1995) clearly has its roots in the tradition of Latent Semantic Indexing described in this section.

Schütze (1995) goes on to introduce various refinements to the basic system. For example, a variant that tags not word types, but tokens in context is described. Also, the use of generalised context vectors is mentioned: this operates in two stages, where the clusters resulting from the basic model are used to replace words as contexts for a generalised model. Lamar et al. (2010) revive a variant of this method, testing it against more recently devised techniques for POS induction, against which it compares favourably with less computational effort.

Table 4.2: *A summary of some VSMs described.*

Publication	Target Unit	Context	Weighting scheme	Transformation	Application
Salton et al. (1975)	Document	Index terms	E.g. TF-IDF	None	Information retrieval
Lund and Burgess (1996)	Word	L and R neighbours (10 word windows)	Frequency and distance from target	High variance context-selection	Psycholinguistics
Redington et al. (1998)	Word	LL, L, R and RR neighbours	Frequency	None	Psycholinguistics
Deerwester et al. (1990)	Document	Index terms	E.g. TF-IDF	Slim SVD	Information retrieval
Schütze (1992)	Word	Words in a context window	Frequency	Slim SVD	Word sense disambiguation
Schütze (1995)	Word	L and R neighbours	Frequency	Slim SVD	POS induction
Honkela et al. (2010)	Word	L and R neighbours	Log frequency	Slim SVD & ICA	NLP
Lagus et al. (2005)	Morpheme	R neighbour	Log frequency	Slim SVD & ICA	NLP

4.3.3 Reduced Vector Space Model: Other Projections

Various alternatives to SVD have been proposed for vector space models. Any alternative factorisation that is proposed for IR can be applied also to the induction of syntactic or semantic word spaces. For example, Kaski applies random projections to IR; Sahlgren (2006) builds many word-spaces using this same technique. Isbell and Viola (1998) use Independent Components Analysis (ICA) for IR where it outperforms SVD. Honkela et al. (2010) describes the use of an ICA algorithm for extracting word representations from text. The perceived problem with LSI that motivates this work is that:

> [...] the latent concept space is difficult to understand by humans (Honkela et al., 2010).

The main difference between SVD/PCA and ICA in practical terms is described as follows:

> [...] while PCA finds projections which have maximum variance, ICA finds projections that are maximally non-Gaussian (Honkela et al., 2010).

The second, larger-scale experiment reported uses a corpus of c. 22 million words (c. 188,000 types) of text from Project Gutenberg. 10,000 target words are used and 1,000 context words. Context positions are narrow, consisting of immediately neighbouring tokens. An initial transformation is applied to the raw co-occurrence matrix: 1 is added to all elements, of which the logarithm is then taken to dampen differences in word frequency. As a preliminary to ICA, the matrix is 'whitened', and PCA is performed on it for decorrelation and dimensionality reduction. Then ICA is applied. ICA finds a rotation of the PCA feature space; it is argued that the effect of this is that it favours features which are meaningful representations of linguistic phenomena. No clustering is performed: it is considered that the features will be useful in their own right, although no task-based evaluation is carried out. A *separation measure* is used for evaluation of the continuous features.

Lagus et al. (2005) use ICA to induce representations of morphemes. Context vectors of morphemes are compiled from a corpus of Finnish newstext where orthographic words have been semi-automatically segmented into morphemes. Context features used are the counts of the 506 most frequent morphemes occurring to the right of the target. Note that although a gold standard segmentation is used here, units segmented automatically (Creutz and Lagus, 2007) would also be amenable to this sort of characterisation.

4.3.4 Probablistic topic models

Information retrieval

Latent Semantic Indexing has been reformulated as a *probabilistic topic model* (Steyvers and Griffiths, 2006), a type of mixture model. In *Latent Dirichlet Allocation* (LDA: Blei et al., 2003), the words of an observed document are considered to be drawn from a mixture of multinomial distributions over the words in the vocabulary, each corresponding to a latent *topic*. The contribution of topics to a document is in turn modelled by a multinomial distribution over topics for each document. Distributions over topics are drawn from a symmetric Dirichlet distribution, the setting of whose parameter can enforce sparsity: i.e. that documents drawn mainly from a few of the topics have a higher probability. Distributions over words can also be drawn from a Dirichlet distribution; sparsity in this part of the model means that a few vocabulary items contribute richly to a given topic. Given a collection of documents and the number of topics, model parameters can be estimated using variational expectation-maximisation (Blei et al., 2003) or Gibbs sampling (Griffiths and Steyvers, 2004).

Syntactic Features

LDA is used directly by Chrupała (2011) for the induction of word classes. The *documents* of the topic model become *word types*, *words* become *context features*, and *topics* become *word classes*. The procedure of Chrupała (2011) therefore bears the same relation to LDA as the procedure outlined in Section 4.2.4 bears to LSI. The context features of a target word are precisely the same as in Section 4.2.4: the counts of the immediate left and right neighbours of that word in the training corpus. Christodoulopoulos et al. (2011) present a mixture model in some ways similar, where basic word co-occurrence features are supplemented with features derived from bilingual corpora and unsupervised morphological segmentation.

4.4 Other Techniques for the Induction of Linguistic Representations

Some past work on the use of vector space models and related techniques for the induction of word and document representations from text corpora has now been

surveyed. However, much other work has been done on such induction using distributional methodologies other than the vector space model. These deserve to be mentioned here. The work is here partitioned into four methodological categories for the purposes of discussion: hierarchical clustering approaches, HMM-based approaches, graph partitioning approaches and connectionist approaches.

4.4.1 Hierarchical Clustering Approaches

Hierarchical Clustering of Words

A common way to induce representations for words is to cluster them according to their context in a large text corpus. Much work using such clusters builds on the methods outlined in the influential paper of Brown et al. (1992). There, clustering algorithms are presented in the context of class-based bigram language modelling, where a deterministic assignment of words to classes is assumed, and where the probability of a word being generated depends on the class of that word and on the class of the previous word. The likelihood of a corpus computed using this model provides a criterion which a class map of a given size seeks to optimise. Brown et al. (1992) show that maximising the likelihood of the training data is essentially the same as maximising average mutual information (AMI) between adjacent word classes.

The core algorithm presented in Brown et al. (1992) is a greedy hierarchical algorithm that starts by assigning each word to its own class, then finding the pair of classes which gives the smallest loss in AMI when merged. Repeating this procedure until a single cluster remains, but keeping a record of the merges made, results in a binary tree whose root corresponds to the whole vocabulary, and whose leaves correspond to individual words of the vocabulary. Intermediate nodes represent clusters of varying coarseness. It is observed that exchanging words between classes after inducing the required number as already described can produce further gains in performance.

Brown et al. note that the classes they extract:

> [...] have the flavor of either syntactically based groupings or semantically based groupings, depending on the nature of the underlying statistics (Brown et al., 1992).

The core algorithm, using AMI over adjacent word classes, gives more 'syntactic looking' results. But another mutual information criterion (called *semantic stickiness*) is presented at the end of the paper that considers not the relation of

neighbouring words, but of non-neighbouring words in a 1000-word window. All work building on Brown et al. (1992) discussed here, however, uses the adjacent word rather than the *semantic stickiness* method.

A great deal of later work has made use of Brown features in simple two-step semi-supervised learning set-ups. The Brown algorithm and the optimisation criterion it uses are closely linked to language modelling. However, the fact that the clusters produced by the algorithm represent in many cases word classes of a generally applicable nature means that clusters resulting from this and related algorithms have been used by other researchers for a variety of different tasks. A feature of the resulting tree-structure frequently used in this later work is that it encodes clusterings at differing granularities, which means that the optimal number of clusters does not need to be determined ahead of time. A path from root node to a leaf of a binary tree can be represented as a bit-string; a common strategy is to not choose one cut of the dendrogram to be used (i.e. bit-strings of a single fixed length), but rather to specify a range of bit-string prefixes of varying lengths for each word, or even to query individual bits in a word's bit-string.

Ushioda (1996) uses a form of the AMI algorithm (modified to produce more balanced tree structures) to induce word features which are then used to train a decision-tree based part of speech tagger. The decision-tree building algorithm is allowed to query individual bits in words' bit-strings, as well as using a set of basic questions. Using the AMI features provides up to 10% reduction in tagging error rate (depending on the size of the untagged corpus and on which POS-tagged corpus is used) over a system where the AMI features are randomly permuted. As also previously reported in Brown et al. (1992), continuing to exchange words between classes after hierarchical clustering has finished produces further gains in performance.

A resurgence of interest in the use of Brown features for semi-supervised learning was heralded eight years later by the paper of Miller et al. (2004). Here, bit-string prefixes of four different lengths (8, 12, 16, and 20 bits) are taken from an AMI hierarchical clustering tree built on approximately 100 million words of news-text, and incorporated into the training of an averaged perceptron named-entity tagger. Using the features improves system F measure on all sizes of name-tagged training dataset (from 5000 up to 1 million words). Again, the authors stress that the inclusion of a variety of lengths of bit-string prefixes is important for avoiding commitment to any particular granularity of clustering ahead of training on the tagged data.

Following Miller et al. (2004) were a series of publications reporting the use of

such two-step training schemes with Brown cluster features for semi-supervised learning on a variety of NLP tasks. For example, Li and McCallum (2005) apply Brown clusters (as one of their unsupervised features) once again to POS tagging, and in Koo et al. (2008) and Candito and Crabbé (2009), Brown clusters are used for dependency and PCFG parsing, respectively.

Hierarchical Clustering of Phonemes

Chelba and Morton (2002) describe the induction by distributional means of phonetic classes for use in state-tying decision trees for speech recognition. From a phonemic transcription, phoneme bigram counts are made, and the AMI clustering algorithm of Brown et al. (1992) is applied. Recognition experiments are performed, both with the resulting phoneme classes and with conventional expert-specified classes as a baseline. State-tying trees are built in such a way that they have a comparable number of leaf nodes across conditions. The induced classes give results similar to, and sometimes better than, expert-specified classes in German and Spanish systems.

4.4.2 HMM-based Approaches

HMMs for the induction of word classes

Among the experiments of Merialdo (1994), an HMM part of speech tagger is trained by maximum-likelihood estimation using a dictionary mapping tokens to possible tags and an untagged text corpus. Recent work has shown the possibility of doing without any dictionary information, and inducing POS-like tags from scratch (Goldwater and Griffiths, 2007; Gao and Johnson, 2008). This work adopts a Bayesian approach, where the multinomial state–word emission and state transition distributions are drawn from (usually symmetric) Dirichlet distributions. The parameters of these Dirichlet distributions determine what sort of emission and transition distributions are given high probabilities. They can be set in a way that gives higher probabilities to sparse distributions. Such sparse distributions are characteristic of linguistic phenomena, where for example a given POS tag is generally followed by one of a limited set of POS tags.

HMMs for the induction of phoneme classes

Goldsmith and Xanthos (2009, §3.3) describe the induction of the classes *vowel* and *consonant* distributionally using a two-state HMM. The model is trained

using maximum-likelihood estimation on the phoneme sequences from lists of phonemically transcribed words. The resulting model is evaluated as follows: for each phoneme in the vocabulary, the logarithm of the ratio of the probability of that phoneme's emission by state 1 to the probability of its emission by state 2 is taken. For experiments in English and French, this results in negative values for each of the phonemes that are known to be consonants, and positive values for the vowels, showing that one state of the model specialises in emitting vowels, and the other, consonants.

4.4.3 Graph Partitioning Approaches

A graph partitioning approach is used to determine word classes and phoneme classes in Biemann (2006) and Goldsmith and Xanthos (2009, §3.2) respectively. In both cases, induction of the classes proceeds in two stages. First, a weighted undirected graph is constructed where the weight on the arc between two objects (words, phonemes) indicates the similarity of contexts in which they occur. In both cases this graph is derived from a co-occurrence matrix, of phonemes with neighbouring phonemes in Goldsmith and Xanthos (2009) and of target words with neighbouring high-frequency words in Biemann (2006). In Goldsmith and Xanthos (2009) the conversion from co-occurrence matrix to adjacency matrix is done by making and then normalising a Markov transition matrix, and in Biemann (2006) the weights between nodes are derived from the cosine similarity of word type co-occurrence vectors. Finally a graph partitioning algorithm is employed to induce clusters of phonemes or words from this graph. Some refinements are used in Biemann (2006) to handle high- and low-frequency words differently.

4.4.4 Connectionist Approaches

Connectionist Word Representations

In his review of work on distributional semantics, Lenci (2008) notes that the vector-space lexical representations commonly used have three salient characteristics, that the representations are inherently:

1. context-based
2. distributed
3. quantitative and gradual

It is noted that these three qualities also characterise word representations used in connectionist work. Elsewhere also the output of linear algebraic operations is compared with connectionist representations or described in terms of neural net analogues (Schütze, 1993; Landauer and Dumais, 1997). It is not surprising therefore that connectionist methods have been used directly to induce word representations from co-occurrence observations. Much of this work presents what are meant to be cognitively plausible models of word representation (e.g. Borovsky and Elman, 2006; Rogers et al., 2004). Another, more practically-motivated tradition of work on representing words with connectionist models started a decade ago: Schwenk and Gauvain (2002) and Bengio et al. (2003) present neural net-based language models. Schwenk and Gauvain (2002) note that the main motivation for projecting words into a continuous space is that the probability functions estimated over events will be smooth. The neural net approach is presented as a way of achieving this. However, another major benefit of using neural nets for this work is that both the projection of the words into a continuous space and the weights for combining n-gram histories to make predictions can be learned simultaneously via back-propagation. This overcomes objections to two-step semi-supervised learning approaches where there is no guarantee that the features learned in the first step will be useful for the second (as is the case for the methods developed in this thesis).

Above it was seen that Brown word clusters – learned to optimise a criterion motivated by language modelling – have been used for many tasks besides language modelling. The same applies to word projections arising from the training of a neural net language model: they are general enough that they can used for other tasks. For example, in Turian et al. (2010), three types of word feature obtained in an unsupervised way are evaluated on two different supervised tasks, named-entity recognition and chunking. Conventional features and the unsupervised features are aggregated; the aggregate systems outperform the conventional systems having access to near state-of-the-art supervised features, and combinations of the different types of unsupervised word features lead to additional improvement. This approach is open to the same criticism that all 'two-step' approaches face – there is no guarantee that the embeddings will be useful for the task.

Collobert and Weston (2008); Collobert et al. (2011) take the idea of task-based estimation of word embeddings from neural net language modelling and apply it to an array of NLP tasks in a multitask learning framework. Once again, language model learning is used as an auxiliary task: the language model's

predictions are not used, only the embeddings of words are required. However, models for five other tasks (POS tagging, chunking, named-entity recognition, semantic role labelling and semantic relation classification) are trained in parallel; the word projection initialised by the language model training is a shared layer for all these tasks, hence back-propagation allows all tasks to affect the word representation in a mutually beneficial manner. This would be an attractive new architecture for TTS, where models for several NLP tasks in a single domain are conventionally trained and applied as a cascade, leading to probably sub-optimal results.

Connectionist Letter Representations

Jensen and Riis (2000) describe the learning of representations for letters within a multilayer perceptron for performing letter-to-sound conversion. Codes for letters in a five-letter context window form the input to the network, which outputs phoneme posterior probabilities. The baseline system uses orthogonal one-hot codes for letters (i.e. the code consists of 27 entries for lowercase letters and a *null* token). A *codeblock network* which takes the 27 inputs of the orthogonal codes and produces n outputs is inserted into the system. By training the whole network and having all instances of the codeblock network share the same weights, a set of 27 codes of length n is obtained, that expresses correlation between input letters in a way that is optimal for the task. In the evaluation performed, setting n to 15 gives highest accuracy, and using the codeblock network with any length of code outerperforms the one-hot codes, and even a set of codes that are manually specified and include information about categories of letters (vowel, consonant, etc.).

Chapter 5

Letter Space

5.1 Introduction

The method of distributional–acoustic analysis introduced in Section 4.2 is here applied to sub-syllabic units of English: both to letters and to phonemes. The experiment presented in this chapter has the same goal as Experiment 3 of Chapter 3: the induction of phonetic classes useful for acoustic modelling. In comparison with the serial tree building technique developed there, however, the method used to attempt this task in the current chapter – the vector space model (VSM) – is expected to be more easily extensible to other levels of analysis.

The serial tree building technique developed in Chapter 3 achieved a performance that rivalled that of a benchmark system incorporating expert knowledge. It might therefore be argued that approaching the same task with a different method is redundant. Furthermore, this is a task in which no use is made of one of the great strengths of the VSM – its ability to generalise from units *heard* in a primary corpus to ones only *seen* in a tertiary corpus. This is because the set of units to be characterised (letters or phonemes) is small and closed. The purpose of returning to this task is to try the VSM on a familiar task before moving to levels of analysis where the VSM's ability to generalise will be of value. Furthermore, if similarly good results could be obtained with the VSM as with serial tree building, it would enable the same technique to be employed on multiple levels of analysis in a unified way, simplifying system construction.

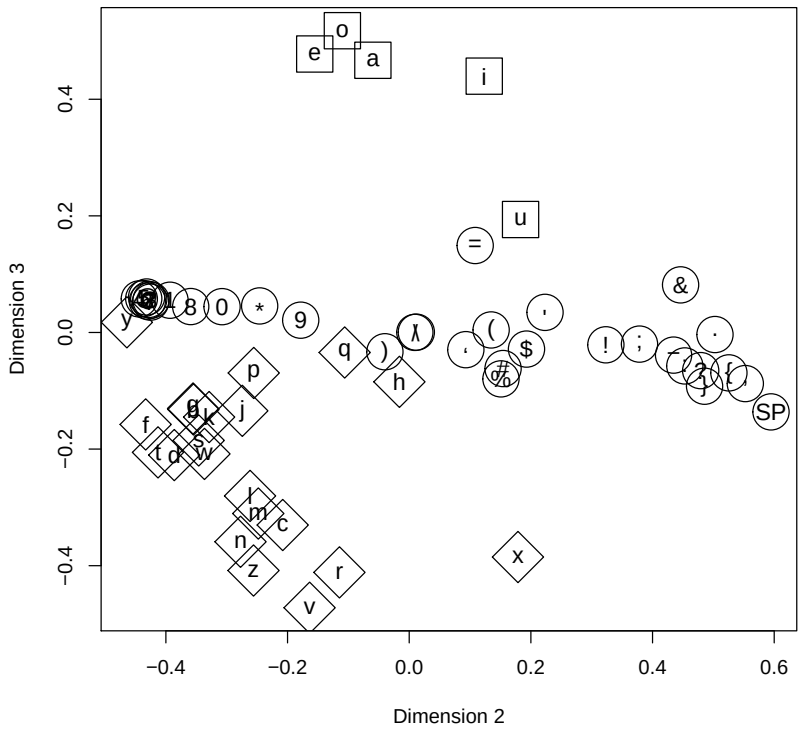


Figure 5.1: *Two dimensions of a space of letter types built from Wall Street Journal text. Squares denote letters that are nominally vowels, diamonds consonants, and circles numerals and punctuation. SP represents whitespace.*

5.2 A Vector Space Model of Letter Types

A vector space model over letters observed in a corpus of running English text was built. The text of the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993) was used for this purpose. These data are available in tokenised form. To mimic the situation where time or expertise is not available to produce an intricate, language-specific tokenisation, untokenised text was obtained by automatically reversing the Penn Treebank tokenisation of the distributed data. The resulting untokenised text consists of c. 1 million words and just over 6 million letter tokens (including counts of spaces and other non-alphabetic characters). This text was converted to lower-case, resulting in a vocabulary of 59 types: 10 numerals, 26 letters and 22 punctuation and miscellaneous symbols, and whitespace. A vector space model of these types was built, following the procedure outlined in Section 4.2.4 and using immediate left and right neighbours of each type as contexts.

Two dimensions of the space resulting from singular value decomposition are

shown in Figure 5.1. To aid interpretation, nominal letter categories have been coded in this figure by shape. These categories are called nominal, because the traditional bipartite division of English letters into vowels and consonants glosses over the opaque letter-to-sound relationship of English orthography. For example, the letter ⟨y⟩, traditionally called a consonant, commonly corresponds to either a vowel or a glide. With this proviso, squares in the figure denote vowels, diamonds denote consonants, and circles all other symbols (numerals, punctuation, etc). *SP* in this figure represents whitespace.

Ignoring the non-alphabetic characters (in circles), it is immediately obvious from the figure that the discovered space is able to represent a partition of the symbols that separates vowels from consonants. The non-alphabetic characters are densely concentrated over a small range of dimension 3, and roughly group into numerals and non-numerals along the other axis shown in the figure.

Informal inspection of a couple of dimensions of the space, therefore, shows that the model has learned some key properties of the inventory of letters in an unsupervised way. However, it is already known from previous work that the most obvious division found by the space – into vowels and consonants – can be obtained by distributional means (see Section 4.4.2). It is desirable to know whether other phonetically-relevant classes are also represented in such spaces. This model over letter types is hard to evaluate in a more rigorous way because there is no well-established set of categories against which to test it. It already seems clear that the space can handle the only well-established division of letters (into vowels and consonants). To be able to compare an induced vector space over sub-syllabic units rigorously against a well-established set of reference categories, a similar model was built using English phonemes instead of letters.

5.3 A Vector Space Model of Phoneme Types

The Unilex Received Pronunciation Lexicon (based on the Unisyn accent independent keyword lexicon: Fitt and Isard, 1999) was used for this experiment. Stress and syllabification were removed from the word pronunciations, resulting in a list of 105,394 unique pronunciations, made up of 903,378 phoneme tokens, not counting the word boundary markers which were added. This preprocessing results in a list of pronunciation types. In this regard, the approach taken here is more similar to the one described by Goldsmith and Xanthos (2009), where pronunciations from lexicons are used as training data, than that of Chelba and

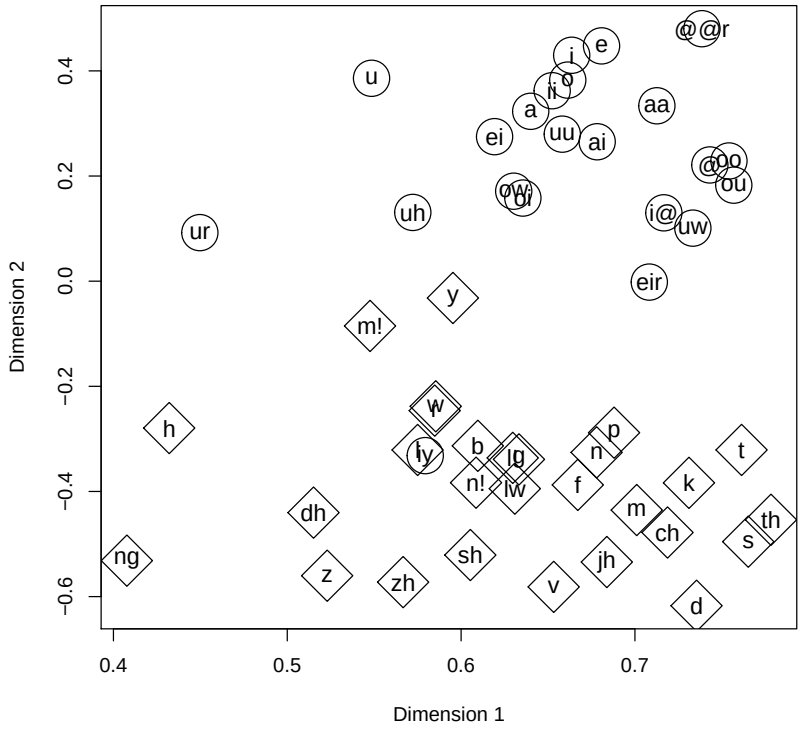


Figure 5.2: *Two dimensions of a phoneme type space induced from the Unilex Received Pronunciation Lexicon. Vowels and consonants are indicated in this figure by circles and diamonds respectively.*

Morton (2002), where phonemic transcriptions of running text are used with the result that word frequency effects are incorporated into those models.

The phoneme space was constructed from this list of pronunciations using the same procedure as for the letter space (described in Section 4.2.4). Specifically, a context vector is assembled for each of the 50 (m) phoneme types in the corpus. The number of types used to give context (n) is set to 51: a word boundary symbol was added to provide word-terminal contexts, and is used as a left and right context along with the 50 phoneme types. Truncated SVD gives a space of r dimensions; for building this phoneme space, r was set to 5.

Two dimensions of the resulting space are plotted in Figure 5.2. It can be seen that there is a clear grouping of vowel and consonant phonemes. From this, an orthogonal partition of the space – of the type that the decision tree building algorithms used in the current work are restricted to – would be able to almost perfectly represent these two groups. The knowledge-based classes of vowels and consonants are indicated in this figure by shape: circles denote vowels, and diamonds denote consonants. Other finer clusterings of symbols appear to

correspond to smaller intervals along a sonority hierarchy. For example, fricatives and affricates are all characterised by low values in dimension 2, and so can be found along the lower edge of the plot. Stops are clustered in a small area of the bottom right corner of the plot. Liquids, nasals and glides are mostly grouped together in the central area of this plot.

This sort of informal inspection of a few dimensions of the discovered space suggests that some useful phonetic generalisations about segments are captured by it, but is ultimately unsatisfactory. Often it seems from looking at a couple of dimensions in isolation that oblique partitions would in many cases be more useful for isolating linguistically plausible groups of segments than ones orthogonal to the axes of the latent space. This raises the question of whether univariate decision trees will be able to exploit the space. The following section moves from informal inspection of the space to determining its practical value to a TTS system.

5.3.1 Experiment

As already mentioned, using phonemes instead of letters for the VSM by-passes the problem of not having many well-defined classes of letter against which to benchmark. In the case of phonemes, there are well-defined phonetic classes that can be withheld from a baseline benchmark system and incorporated into a topline system. The challenge for the VSM system is then to close as much of the gap in performance between the baseline and the topline as possible. If the VSM system performs similarly to the topline system, we can conclude that the VSM has found some representation – in an unsupervised way – that is beneficial for TTS in the same way that expert-specified phonetic classes conventionally used in TTS are beneficial.

Procedure

Five systems were used in this experiment; they are summarised in Table 5.1. For all systems, the data-set *RJS-1000* (described on page 53) was used. The HTS-2010 procedure (see Section 2.1.3) was used to build acoustic models (10 iterations of the context clustering procedure were used). The only aspect of the five systems that was varied was the set of features incorporated into the annotation made available to them. The baseline system (here called System Q and identical to the baseline system of the same name in Experiment 3 of Chapter 3) incorporates only features about the identity of phonemes in a 5-phoneme

Table 5.1: *Details of systems evaluated in the phoneme space model experiment. Table 5.2 demonstrates the feature types named at the top of this table.*

System	*Quin-phones*	*Phonetic (All)*	*VSM*	*Phonetic (V/C)*
Q	X			
QC	X	X		X
QV	X		X	
QL	X			X
QCV	X	X	X	

Table 5.2: *Examples of feature types in Table 5.1*

Feature type	*Featureexample*
Quinphone	{LL, L, C, R, RR} phoneme is {i, ai, uh, ng, ...}
Phonetic (All)	{LL, L, C, R, RR} phoneme is {vowel, nasal, plosive ...}
Phonetic (V/C)	{LL, L, C, R, RR} phoneme is {vowel, consonant}
VSM	{LL, L, C, R, RR} phoneme's representation is $< x$ in dim. {1, 2, 3 ...}

window around the modelled unit (feature class F in Table 2.1 on page 7). The topline system (QC) supplements these basic questions with questions referring to expert-defined phonetic sets of units (vowel, nasal, plosive, etc.) for each of the 5 quinphone contexts (feature class FC in Table 2.1). The experimental system (QV) has access to the VSM already described, but makes no use of expert-defined categories.

Two further systems were built, in addition to the baseline, topline and experimental systems, as follows. It is clear from visualisations of the discovered space such as that shown in Figure 5.2 above that it will allow vowels to be distinguished from consonants by a univariate decision tree. As has already been mentioned, it is less clear to what extent other phonetically-useful categories can be made to emerge on purely distributional grounds. System QL was built to control for the possibility that System QV uses the vowel-consonant split implicit in the phoneme space, but makes no other use of the space. Like system QC, System QL has access to expert specified knowledge, but rather than the full set of 91 phonetic classes used by QC, QL has only knowledge of two of those categories: vowels and consonants. Finally, system QCV was built to find out if the expert-provided knowledge used by QC and the automatically induced

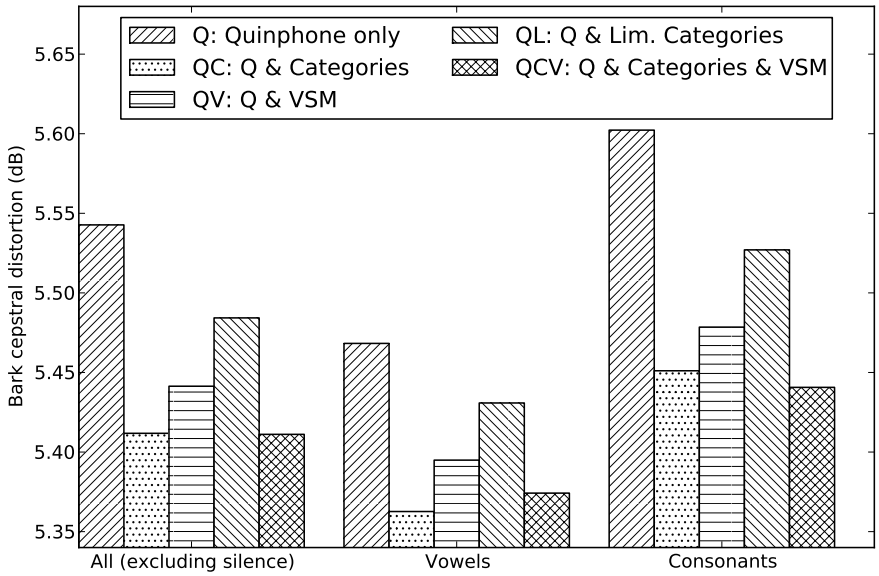


Figure 5.3: *Results of phoneme-space experiment. See Table 5.1 for explanation of the system identifiers.*

knowledge used by system QV are in some way complementary, or whether one representation is redundant given the other.

As already mentioned, 5 dimensions of the phoneme space were available for use. The space was represented in the feature set used by Systems QV and QCV by features representing a binary split for each pair of the set of 50 phonemes adjacent along each dimension of the space. This gives $(50 - 1) \times 5$ features for each quinphone position, resulting in 1225 features over all five of these positions.

Results

An objective evaluation of the three voices was carried out using the same procedure as for Experiment 3 of Chapter 3, described on page 56. To recap, 100 utterances for which time-aligned labels were available were held out of training, and synthesised with natural segmental durations. The synthesised speech parameters could then be compared on a frame-by-frame basis with the parameterised natural samples. Mean Bark cepstral distortion of non-silent frames was computed for this experiment, because spectral envelope is the attribute of speech that we expect to be most affected by the manipulation of phonetic representations available to a system. Segments labelled as pause and silence were excluded throughout. As well as a general evaluation over all non-silent segments, measures were also computed separately over vowel and consonant segments.

Results are presented in Figure 5.3. It can be seen that, as expected, the use

of conventional phonetic categories improves performance, decreasing Q's Bark cepstral distortion from 5.54dB to 5.41dB for QC. While it does not perform as well as the system using the serial tree method in the experiment of Section 3.4, the VSM-based system, QV, still closes most of the gap between topline and baseline. It outperforms system QL (having knowledge of consonants and vowels), confirming that the phoneme space embodies acoustically-relevant partitions besides the obvious partition of the space into vowels and consonants that is most obvious in Figure 5.2. When the knowledge-based questions are combined with the VSM-derived ones, performance is very similar to that of the topline system, which suggests that the useful categories implicit in the VSM are similar to ones in the expert-specified phonetic classes.

5.4 Conclusions

This chapter has outlined the construction of two vector space models of sub-syllabic units of English: a space characterising letter types, and one characterising phoneme types. Systems were built with the phoneme space principally for ease of evaluation, as for phonemes there exist well-defined categories that serve as a point of reference in evaluation. The systems built for the experiment are however in no way considered to be toy systems. The use of phonemes in an English system is comparable to the direct use of letters as modelling units in languages with a more transparent letter–phoneme correspondence (e.g. Finnish). The experimental systems built are therefore of relevance to real-world situations in which pronouncing dictionaries and phonetic knowledge are not available for some target language with a transparent alphabetic orthography. The experiment demonstrates that distributional analysis, which takes place before any acoustic signal has been introduced into the voice-building procedure, gives the system useful indications about which partitions of its inventory of units will be acoustically relevant. Real advantage of this approach's strength is not taken in the present task, where it can safely be assumed that all units will be encountered in the primary data. Such is not the case with word-sized units, and it is to this higher level of analysis that the following chapter applies the same vector space model approach.

Chapter 6

Word Space

6.1 Introduction

The vector space model outlined in Chapter 4 has been shown to yield acoustically relevant representations of sub-syllabic units of English. The experiments presented in this chapter are designed to establish whether similar representations at the word level are also relevant to TTS. In the present chapter, two such tasks are used as the basis of experiments: decision tree clustering of acoustic model states, as in Chapter 5, and the prediction of phrase-breaks from text, also using decision trees.

The hypothesis of both these experiments is that the word-level vector space model – derived automatically from unannotated text – can be successfully used in place of a conventional part of speech tagger with minimal degradation of performance. In the second of these (state-tying), the space of word types is partitioned using distributions over acoustic values as a response variable, as in the experiment of Chapter 5. The partitioning therefore relies on no hand-annotated data, but is made using a response (distributions over acoustic parameters) obtained automatically from a naive 'labelling' by a speaker of a set of prompts with appropriate waveforms. This speaker's expertise might be no more specialised than a knowledge of the target language and literacy in that language. The first experiment presented in this chapter (phrase-break prediction), in contrast, relies on a corpus that is hand-labelled with phrase-breaks by experts. This task was chosen mainly because previously-published work using the same dataset provides points of reference (see Section 6.3.3). However, it is expected that features obtained much more naively from the acoustic signal (e.g. silent segments whose duration exceeds some threshold) will be reasonable surrogates for

manually-labelled phrase-breaks. This approach is taken when systems are built with no reliance on hand-labelled data in Chapter 8. Before the experiments are presented, details of construction of the word space used in both experiments will be specified, along with some informal observations about the space's characteristics.

6.2 A Vector Space Model for Word Types

Data

The data used to build the VSM of word types that is employed in the experiments of the present chapter is the text of the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993). The tokenisation of the corpus is used as provided (punctuation tokens were retained), but all other information was discarded (parse trees, POS tags, sentence boundaries, capitalisation and sentence boundaries – except as signalled by punctuation).

Procedure

The resulting text consists of 1,173,767 word tokens and 43,767 word types including punctuation. A vector space model of these types was built, following the procedure outlined in Section 4.2.4 and using immediate left and right neighbours of each type as contexts. Values of parameters n (the size of context vocabulary against which co-occurrence is counted) and r (the dimensionality of the latent space) used were the same as those used in Schütze (1995): 250 and 50. Initial experiments suggested that these would yield good results for present purposes. The 250 most frequent words of the corpus which constitute the feature words used are given in Table 6.1. For the word space described in this chapter, the simple *unseen* token method was used to handle out of vocabulary words as described in Section 4.2.4. That is, tokens in a 1% portion of the data whose types are not seen in the other 99% are rewritten with a special symbol to represent unseen words at run-time. 260 types contributed to the unseen model. In this way, a 50-dimensional vector representation for each of 43,507 word types and one for the unseen symbol were induced (i.e. the value of m for this space is 43,508).

Table 6.1: *250 most frequent tokens in the Wall Street Journal text used, used as feature words for the Word Space.*

,	the	.	of	to	a	in	and	's
for	that	$	“	is	”	it	said	on
%	at	by	as	from	with	million	mr.	was
be	its	are	he	but	has'	n't	'an	have
will	new	or	company	they	this	which	year	would
about	market	–	says	more	were	had	billion	their
his	up	u.s.	one	than	stock	been	some	who
also	other	share	not	:	we	corp.	when	;
last	if	i	all	shares	)	president	years	(
trading	first	two	after	inc.	because	could	sales	&
out	there	do	only	business	such	most	can	co.
york	into	may	over	group	many	time	now	federal
companies	prices	no	government	so	any	cents	quarter	bank
investors	down	you	price	exchange	'	what	people	even
say	yesterday	much	big	while	months	securities	under	week
rose	them	bonds	stocks	major	next	three	net	interest
earnings	did	financial	still	1	make	chairman	american	just
earlier	board	through	investment	before	those	since	chief	industry
executive	these	state	money	national	program	off	officials	friday
10	expected	made	analysts	like	rate	she	unit	month
markets	days	house	does	30	profit	sell	buy	between
against	rates	both	plan	capital	firm	income	back	get
recent	revenue	japanese	ago	general	average	products	during	well
should	own	funds	index	international	offer	fell	issue	another
court	part	debt	then	take	trade	among	higher	including
?	her	being	15	however	8	japan	1988	each
reported	according	world	computer	tax	vice	sale	plans	work
sold	lower	several	way	report	traders	past		

Analysis of the Word-Level VSM

The word type space was inspected subjectively to determine whether its dimensions have any intuitively straightforward interpretation. Of course, the model is not expected to express a crisp division of words into traditional knowledge-based classes, but it is expected that the discovered space will have some ability to represent such a division. Indeed, it is the assumption that the space will be able to stand in for conventional part of speech tags that motivates the design of the two experiments presented later in this chapter, where the type of module used for assigning representations to words (either a POS tagger or VSM) is varied between different experimental conditions.

To this end, the nearest neighbours of each of the first 50 latent axes of the space were found from among the 3000 most frequent words of the corpus. Similarity was measured with cosine similarity. Although not used in the experiments and systems presented in this thesis, this is the most commonly used measure of similarity between vectors employed in LSI-style approaches to Information Retrieval (see Section 4.1).

A selection from among the 50 dimensions is given in Table 6.2 on page 107. Some of these dimensions have clear syntactic interpretations. For example, the close neighbours of axis 1 consist mainly of prepositions: *of*, *against*, *into*, *across*, *toward*, *below*, *above*, *over*, *throughout*, *on*, *by*, *during*, *inside*, *beyond*;

among axis 3's neighbours are many adverbs: *largely*, *primarily*, *formerly*, *mainly*, *mostly*, *partly*, *usually*, *especially*; axis 6's neighbours are made up entirely of singular nouns: *boom*, *case*, *team*, *committee*, *bill*, *firm*, *field*, *floor*, *moment*, *game*, *day*, *machine*, *problem*, *group*, *event*, *network*, *campaign*, *organization*, *trial*, *gap*;[1] auxiliaries dominate the top positions of axis 9's neighbours: *should*, *must*, *might*, *can*, *would*, *will*, *may*, *could*, *ll*.[2]

As expected, the space captures properties of words besides purely syntactic ones. For example, axis 3's adverbs, already listed, are mainly adverbs of degree or extent. Some dimensions seem to represent semantic aspects of the domain in question: many of the nearest neighbours of axis 17 reflect the space's construction from Wall Street Journal text, and are not only past tense verbs, but of verbs used to describe changes in stock prices: *slid*, *rose*, *fell*, *jumped*, *plunged*, *soared*, *climbed*. Dimension 29 perhaps reflects newspapers' more general concern with reportage and the dating of events – these are words commonly collocated with the word 'year(s)': *consecutive*, *last*, *this*, *next*, *fiscal* (Wall Street makes its presence felt again here), *every*, *full*.

Thus while it is clear that certain dimensions of the space are amenable to human interpretation, the interpretation can rarely be made on a purely syntactic basis. This is expected, and is not necessarily a disadvantage. It is not clear to what extent the word representations ideal for our task should be similar to conventional POS tags. If a class of 'words that describe what happens to stock prices' is an important one in the domain of our end task, it might well be a useful one to be able to represent. This would be the case, for example, if we wanted to predict phrase-breaks in Wall Street Journal-like text; unfortunately, this isn't the case in the experiments presented here. Essentially, word representations obtained as described here reflect the contents of the data used to induce them, and – as already noted in Section 4.2.3 – the closer the domain of the untagged data to that of the supervised task, the better we would expect those representations to perform.

[1] Although it should be noted that the POS of some of these words is ambiguous.
[2] The final item here is the contracted form of *will* or *shall*.

Table 6.2: *The 20 nearest neighbours of a selection of 12 of the first 50 latent axes of the word space described in Section 6.2, from among the 3000 most frequent words of the corpus, together with their cosine similarity scores.*

Dimension 1		Dimension 3		Dimension 6		Dimension 7	
.	0.64	’	0.62	boom	0.9	entire	0.72
reflects	0.6	compared	0.61	case	0.89	underlying	0.72
of	0.59	stearns	0.6	team	0.89	supreme	0.72
against	0.59	largely	0.58	committee	0.88	year-ago	0.72
into	0.59	primarily	0.58	bill	0.87	no.	0.72
across	0.58	formerly	0.58	firm	0.87	year-earlier	0.71
,	0.58	sachs	0.58	field	0.87	previous	0.71
toward	0.57	mainly	0.58	floor	0.87	latest	0.71
below	0.56	who	0.58	moment	0.87	current	0.71
;	0.56	harris	0.57	game	0.87	main	0.7
above	0.56	peabody	0.57	day	0.87	same	0.7
over	0.55	via	0.55	machine	0.87	soviet	0.7
throughout	0.54	effective	0.55	problem	0.87	original	0.7
entered	0.53	mostly	0.55	group	0.87	dow	0.69
on	0.53	partly	0.55	event	0.86	nikkei	0.69
by	0.52	judges	0.54	network	0.86	fourth	0.69
during	0.51	usually	0.54	campaign	0.86	biggest	0.69
inside	0.5	plus	0.54	organization	0.86	newly	0.69
runs	0.5	whose	0.54	trial	0.86	nasdaq	0.69
beyond	0.5	especially	0.53	gap	0.86	nine	0.69

Dimension 9		Dimension 17		Dimension 26		Dimension 29	
should	0.64	totaled	0.59	risen	0.4	consecutive	0.56
must	0.63	totaling	0.58	occurred	0.4	last	0.55
might	0.63	earned	0.57	grown	0.4	this	0.51
can	0.63	slid	0.54	been	0.39	next	0.42
would	0.63	rose	0.53	fallen	0.39	fiscal	0.41
will	0.62	fell	0.51	charged	0.36	model	0.27
may	0.6	jumped	0.49	gone	0.36	claimed	0.19
could	0.57	offered	0.49	settled	0.35	crop	0.17
ll	0.51	roughly	0.49	shown	0.35	prior	0.16
never	0.49	estimated	0.48	changed	0.32	signal	0.16
actually	0.48	(	0.43	turned	0.32	indicated	0.15
to	0.47	paid	0.42	pulled	0.31	every	0.15
really	0.44	plunged	0.41	caught	0.31	full	0.14
probably	0.4	contributed	0.41	resulted	0.31	circuit	0.14
simply	0.4	soared	0.41	taken	0.31	than	0.14
spend	0.38	climbed	0.41	heard	0.3	bet	0.13
receive	0.37	planned	0.41	begun	0.3	complained	0.13
immediately	0.37	committed	0.4	ruled	0.3	how	0.13
eventually	0.36	ought	0.4	stopped	0.29	decided	0.13
add	0.36	exposure	0.4	purchased	0.29	promised	0.13

Dimension 30		Dimension 33		Dimension 35		Dimension 36	
fewer	0.58	than	0.79	sotheby	0.36	adds	0.26
less	0.56	aggressive	0.72	moody	0.36	owns	0.24
more	0.55	sophisticated	0.59	mac	0.34	succeeds	0.22
greater	0.55	expensive	0.55	lloyd	0.33	added	0.21
rather	0.52	attractive	0.51	mae	0.31	knows	0.21
faster	0.5	effectively	0.49	lawson	0.31	thinks	0.21
better	0.47	cautious	0.47	poor	0.29	believes	0.21
we	0.42	heavily	0.35	wang	0.28	turns	0.2
longer	0.4	profitable	0.32	thatcher	0.27	looks	0.2
slower	0.4	quickly	0.32	fe	0.26	sees	0.2
i	0.38	important	0.31	fidelity	0.25	expects	0.17
smaller	0.37	sluggish	0.3	nicaragua	0.25	immediately	0.17
circuit	0.37	widespread	0.3	circuit	0.25	gets	0.16
extremely	0.37	competitive	0.29	mixte	0.24	p&g	0.16
considerable	0.35	complex	0.27	lehman	0.24	sotheby	0.16
you	0.34	closely	0.27	dinkins	0.23	finished	0.15
appropriate	0.34	frequently	0.27	gorbachev	0.23	moody	0.15
higher	0.32	memory	0.26	hooker	0.22	swiss	0.15
sophisticated	0.32	rapidly	0.25	china	0.22	takes	0.15
fair	0.32	active	0.25	mason	0.22	poor	0.15

6.3 Experiment 1: Word VSM for Phrase-Break Prediction

These induced word representations are now applied to two TTS tasks: the front-end task of phrase-break prediction in the current section,[3] and the back-end task of acoustic model clustering in the section that follows.

6.3.1 Background: Phrase-break Prediction

Phrase-break prediction as a stage of TTS conversion was mentioned in Section 2.1.1. It was mentioned there that the conversion of word sequences to part of speech (POS) tag sequences is a useful first step for predicting phrase-breaks from text. The availability of a part of speech tagger is a central requirement in the majority of work on phrase-break prediction. Many different machine learning techniques have been applied to phrase-break prediction; for example, decision trees have been used (Wang and Hirschberg, 1992; Navas et al., 2008), n-gram models (Taylor and Black, 1998; Schmid and Atterer, 2004), finite-state transducers (Bonafonte and Agüero, 2004), and memory-based learning (Busser et al., 2001). The input to whatever classifier is used, however, has consistently included POS tags as features of central importance. In all of the above-mentioned works they are used directly as input into a phrase-break classifier. In Parlikar and Black (2011) they are used additionally to construct parse trees which in turn provide predictor features for a decision tree. Previous work therefore suggests that reducing the set of surface-forms to a smaller set of symbols on knowledge-based, distributional grounds (i.e. via POS tagging) is a necessary first step for phrase-break prediction. We here test whether the induced word representations already described above can stand in as surrogates for POS tags in this task.

Various researchers have shown that a pre-defined set of tags (such as the Penn Treebank tag-set) is often not optimal for the phrase-break prediction task. For example, mapping the full Penn Treebank set to a smaller, coarser set gives improved performance in Taylor and Black (1998), where the mapping is manually specified, and in Read and Cox (2007), where is it learned through an optimisation procedure. Conversely, producing a finer set of classes for certain words is tried both in Stolcke and Shriberg (1996), successfully, and in Busser et al. (2001), with less success. In both cases, systems using surface forms of words are

[3] An earlier and more limited version of the work here called Experiment 1 was presented in Watts et al. (2011).

compared with systems using POS tags and a 'mixed mapping', where POS tags are used except for some small groups of words (e.g. function words), for which the surface forms are used. It can be seen, then, that a knowledge-based POS tag-set provides a good – but not optimal – set of word classes for phrase-break prediction. This suggests the potential for unsupervised induction of representations for this task to improve on the use of generic knowledge-based classes, an idea that was introduced in Section 4.2.3.

6.3.2 Experiment

Data

The phrase-break annotated data used for this experiment was obtained by combining data from the Lancaster/IBM Spoken English Corpus (SEC: Knowles et al. (1996a,b)) and its machine-readable extension, MARSEC (Roach et al., 1993). These corpora consist of material from a variety of speakers and genres, broadcast on BBC radio programmes during the 1980s. A subset of 39 stories was used, the main exclusions consisting of material from the *Poetry* and *Dialogue* genres. The text of SEC is punctuated by volunteers, without reference to prosodic cues from the audio or phrase-break annotations (Knowles et al., 1996a, p. 130). Phrase-breaks in MARSEC are annotated by two annotators from the audio as follows:

> The division [into tone units] is made on phonetic criteria such as pause, lengthening of a preceding syllable, or a break in the rhythm ... (Williams, 1996, p. 51).

For this experiment, MARSEC and SEC were automatically merged to obtain a text in plain (although tokenised) orthography that is both punctuated (from SEC) and annotated with phrase-breaks (from MARSEC). All tokens were converted to lower-case, and punctuation marks and phrase-breaks were associated with the token that precedes them as a feature of that token.[4]

There are 34,824 tokens in the subset of the corpus prepared (not counting punctuation marks). 11% of these are from the *overlap* section – that part of

[4]We note the similar (but more careful) preparation of part of this material (ProPOSEC) by Claire Brierly (Brierley and Atwell, 2010), and acknowledge her generous sharing of her version of the data. This was not eventually used in the work presented here, however, as only the first 10 stories were available, and for consistency our own more rough-and-ready approach was used to prepare a larger data set.

the material annotated prosodically by both the corpus's annotators. These are set aside as a test set; this choice of test set enables easy replication of the training/test division used and is balanced with respect to speaker and genre (consisting of whole sentences from across the different stories and genres of the corpus). 10% of the remaining 89% was set aside as a development set for system tuning. The points for this development set were picked randomly from across the non-overlap section of the data (on a word-by-word basis). This results in a training–development–test division of 80%–9%–11%.

Procedure: CART

CART (Breiman et al., 1993) is used as the classification method in this experiment. CART is expected to yield respectable predictors for this work as its suitability for the task has been established previously (Section 6.3.1). This choice is also motivated by similarities between the phrase-break classification trees and the regression-type trees used for acoustic model clustering in HMM-based speech synthesis. Using CART as the machine learning framework for phrase-break prediction means that a single methodology – the partitioning of a discovered vector space by decision-tree building – can be employed in a unified way across multiple modules of a single TTS system.

The procedure used to build trees for all experiments will now be explained. Classification trees were fully grown using the Gini diversity index as impurity measure and then pruned using minimal cost-complexity pruning. The complexity parameter used for pruning was determined by 10-fold cross-validation, the chosen parameter being the one resulting in the smallest tree from among the values within 1 standard deviation of the value giving highest accuracy on the cross-validation (Breiman et al., 1993). It should be noted that this procedure is slightly different to the one followed in an earlier published version of this work (Watts et al., 2011), where the value minimising prediction error in cross-validation was used to select an appropriate value for the complexity parameter. Furthermore, cross-validation is accomplished by a random partition of the training data into folds. To counter the effects of this non-determinacy, 10 trees are here built for each condition and the means and standard deviations of the relevant values over 10 trees are reported (in contrast to Watts et al. (2011), where values from a single run of the tree-building algorithm per system were presented).[5]

[5]The implementation of CART used in this work is the R package *rpart* (RPART).

System	*Positional*	*GPOS*	*Brill*	*TnT*	*VSM*
B	X				
G	X	X			
Tb	X		X		
Tt	X			X	
U	X				X
GU	X	X			X
TU	X		X		X
Tbt	X		X	X	

Table 6.3: *Details of features used in systems built for phrase-break prediction.* ***Positional*** *denotes a basic feature-set relating to punctuation and tokens' position in utterance;* ***GPOS***, ***Brill*** *and* ***TnT*** *denote feature-sets derived from different part of speech taggers;* ***VSM*** *denotes a feature-set derived from representations of words automatically induced in a vector space model.*

Experimental Conditions

Systems were built in eight sets of experimental conditions. The system-building procedure was identical in each case except for the predictor variables to which the systems had access. Features used by each of the systems are here described, and the different conditions are also summarised in Table 6.3.

All Systems The feature to be predicted for each word by all systems was the level of phrase-break associated with it. All three break types in the original annotation (major break, minor break, disfluent pause) were mapped to a single symbol for *break*, *B*, and words with none of these associated were given a symbol for *no break*, *NB*.

B: Baseline System The baseline system B had access only to features relating to punctuation and tokens' position in utterance. For each token in the corpus, the following 5 basic features were used:

- The identity of the punctuation symbol following the word;
- The number of words {since, until} a word with a *strong punctuation* mark (i.e. excluding quotes);
- The number of words {since, till} the beginning/end of the utterance;

Baseline system B was built with access to only these features; all other systems built had access to these features in addition to further features, as now explained.

G: Baseline System with Guessed POS (GPOS) As mentioned in Section 2.1.1, full POS tagging can be approximated deterministically by compiling a list of the closed set of function words (or possibly, several sub-lists giving distinctions such as *pronoun*, *modal verb*, etc.), tagging these words by look-up in the list(s), and tagging all out-of-list words as content words. Such an approach is tested in system G, which uses 9 lists of different types of function words modelled as closely on those distributed with the Festival TTS system (Black et al., 1999) as differences in tokenisation allow. The lists are nominally of prepositions, the word *to*, determiners, modals, coordinating conjunctions, *wh*-pronouns, possessive pronouns, auxiliaries, and an extra class for the morpheme *'s*. Obviously this method cannot handle the POS ambiguity of function words such as *that*, and makes no distinction between different types of content words. The tags obtained in this way for a token and its right-hand neighbour are included as features of that token. An extra *null* token is used for 'next word's POS' at the end of utterances. The CART algorithm using the *standard question set* (Breiman et al., 1993, §2.4.1) searches all binary partitions of this small tag-set, and thus the simple content–function distinction is modelled implicitly. System G was constructed using these features in addition to the positional features of system B.

Tt and Tb: Topline Systems with Full POS Systems Tt and Tb are toplines, and in addition to the basic features used by System B, incorporate features obtained from two different state-of-the-art POS taggers that had already been trained on approximately 1.2 million words and 300,000 words of manually-tagged data from the Wall Street Journal respectively. System Tt uses tags from the trigram tagger *TnT* (Brants, 2000), and System Tb uses tags from Brill's Transformation-Based Learning tagger (Brill, 1992). As mentioned above, collapsing some fine distinctions made by the full Penn Treebank tag-set has been found to give improvements on this task in previous work (Taylor and Black, 1998; Read and Cox, 2007). Early trials on the development set showed that the 23-tag system of Taylor and Black (1998) (grouping verbs, nouns, adjectives and adverbs into single classes) gives improved results also in the present set-up. This 23-tag set was therefore also used for all of the POS features in the current work. For each token, the tag of that token is included as a feature as well as

the tag of its right-hand neighbour. Contrary to the results with decision trees in e.g. Wang and Hirschberg (1992) and Sun and Applebaum (2001), early trials on the development set suggested that a wider tag window does not improve results, and so a 2-word window was used in all of the current experiments (in effect, tags either side of a juncture are used to predict break type at that juncture). End-of-sentence contexts were represented with the *null* symbol as in the case of the GPOS features.

U: System Using Unsupervised Word Features Besides the positional features of System B, System U incorporates features obtained from untagged *Wall Street Journal* text as described in Section 6.2. It should be noted that this is the same corpus that had been used to train the taggers used for Systems Tb and Tt, except here – obviously – no use is made of the POS annotation. As with Systems Tt and Tb, the obtained features of a token and its right-hand neighbour were associated with that token as features; in the case of System U, however, these features amounted to 100 continuous features (50 for current and following words) rather than two POS tags. End-of-sentence contexts were represented by the mean vector of the whole VSM 'lexicon'.

Systems B, Tb, Tt and U are the systems to make the basic comparisons. In addition to these, three extra systems were built using different combinations of the features already described.

GU: System Combining Unsupervised Word Features with POS This system was built using both the GPOS features of System G and the VSM features of System U to determine if there is any complementarity between the benefits offered by each type of feature. This is a possibly attractive combination as it allows the easy integration of some human expertise into an unsupervised system without a full part of speech tagger being required.

TU: System Combining Unsupervised Word Features with POS This system combines the Brill tagger features of System Tb with the unsupervised word representations of system U. This idea here is to test for complementarity between conventional, topline POS features and the proposed word features. If there is some, then it might be possible even in the best case scenario (where all the desired conventional features are available) to add extra word representations at little expense to see some added improvement.

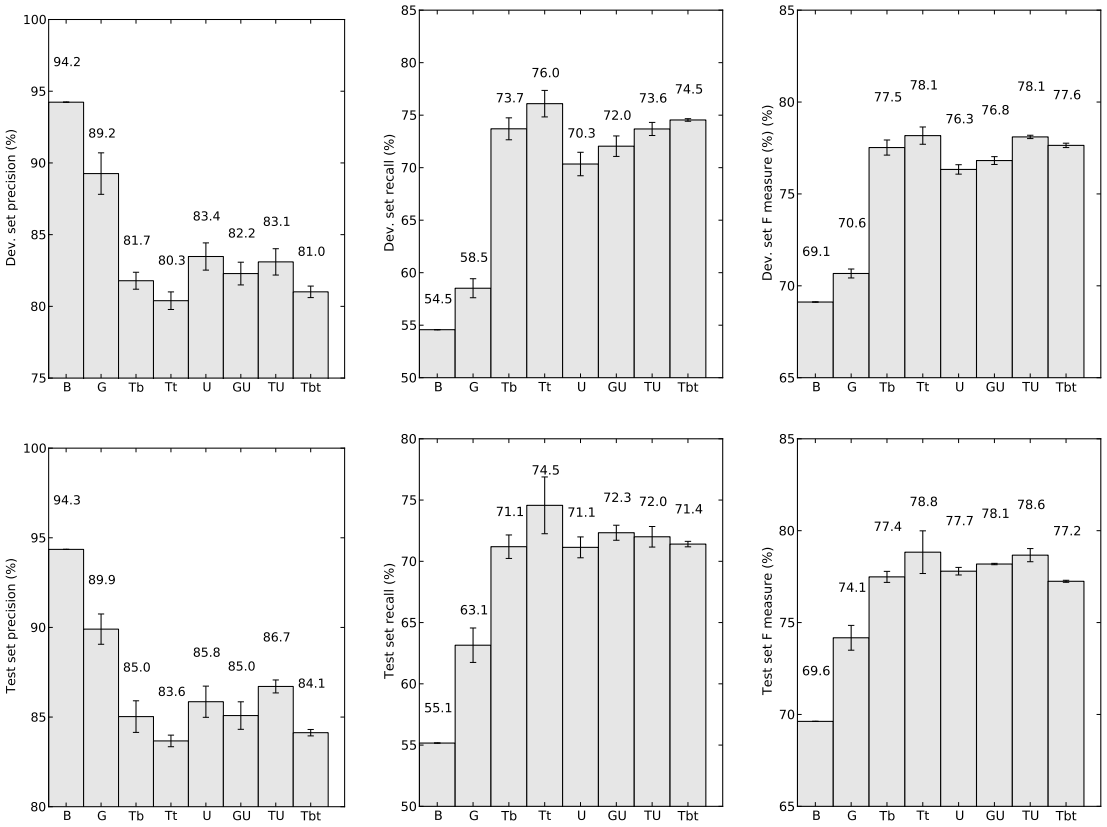


Figure 6.1: *Precision, recall, and F-measure of phrase-break predictors on development (upper) and test set (lower). Means and standard deviations (error bars) over 10 repetitions of tree building are given.*

Tbt: System Using Two POS Taggers Finally, a system was built combining the features of the two taggers already mentioned. The motivation for this system is to compare the effect of combining knowledge-based features with the induced representations against the effect of combining knowledge-based features with other similar knowledge-based features. The expected redundancy between tags from these two systems means that this system is not expected to perform better than the best system using a single tagger.

6.3.3 Results

Results of evaluation of the systems on development and test sets are given in Figure 6.1. The sizes of trees built (measured in number of leaf nodes) are given in Figure 6.2 on page 115. All these plots give the mean scores (precision, recall and F measure) over 10 repetitions of tree building (motivated above); error bars show 1 standard deviation. Plots of the top parts of trees for systems Tb and U are shown in Figure 6.3.

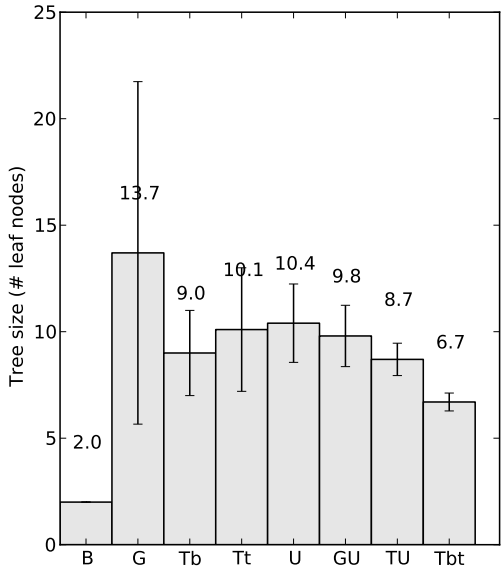


Figure 6.2: *Sizes of trees built for phrase-break prediction; means and standard deviations (error bars) over 10 repetitions of tree building are given.*

Baseline system B achieves an average F measure on breaks in the test-set of 69.6%. The 'guessed POS' (GPOS) features that were incorporated into system G increases this to 74.1%, although performance of this system is less stable across test and development sets than that of any other system; on the development set, the increase in performance between systems B and G is much less (F scores of 69.1% and 70.6% respectively). The size of the tree created using the GPOS features is much less stable between runs of the tree-building algorithm than for any other set of features, shown clearly in Figure 6.2.

Topline Systems Tb and Tt gain mean F measures on the test-set of 77.4% and 78.8%, and thus represent a reasonable level of performance (cf. F measures of 74.4%, 78.3%, and 81.6% in Busser et al. (2001); Taylor and Black (1998); Read and Cox (2007) respectively on MARSEC data[6]). Examination of the binary partitions of the tag-set that are chosen for these trees suggests one reason why GPOS features provide a smaller improvement in performance than POS features: in the case of both splits in the upper portion of the tree for system Tb shown in Figure 6.3, the partition separates different classes of *content* words, and

[6]Possible differences of data division and preparation mean this comparison needs to be made with caution.

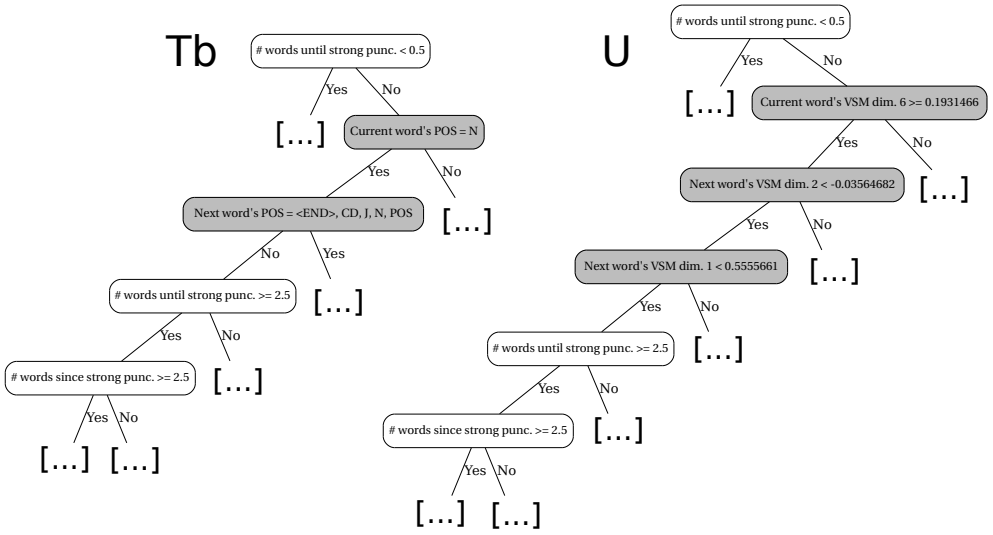


Figure 6.3: *Top portions of trees built for System Tb (Basic and Brill tagger features) and U (Basic and Unsupervised VSM features); nodes querying basic features are shown in white, those querying supervised POS features (in the left-hand tree) or unsupervised VSM features (in the right-hand tree) are shaded grey.*

distinction between content word classes is obviously not enabled by the GPOS features.

The system using unsupervised word features, System U, gains an average F measure on breaks in the test-set of 77.7%, thus closing most of the gap in performance between baseline system B and topline systems Tb and Tt, and giving superior performance to system G. Note the generally similar topology of the top portions of the trees induced for systems Tb and U shown in Figure 6.3: Tb's question about the POS of the current word is mirrored in tree U by a question about dimension 6 of the current word's VSM features. Similarly, the following question about the POS of the following word in tree Tb is mirrored by 2 questions about dimensions 1 and 2 of the next word's VSM features in tree U. The final 2 questions in the tree fragments shown are identical.

Adding GPOS features to the system using induced word representations yields a small gain in performance. Adding the word representations on top of tags from the Brill tagger also improves performance, nearly equalling that of the better-performing TnT tagger. Interestingly, combining the two taggers' features degrades performance: it seems that there is some complementarity between conventional tagger features and the induced features, but that this is not true of features produced by the two full-scale POS taggers.

System	Short description	Contexts Specific to System
B	Baseline	*None*
P	Part-of-speech topline	POS of word and neighbours, from TnT; distance to content word (from GPOS function)
V	VSM	50 VSM features for word and its neighbours (discretisation: 50 evenly-spaced bins)

Table 6.4: *Systems built to test VSM for decision-tree-based state tying and context features characterising them.*

6.4 Experiment 2: Word VSM for State-Tying

Results of the first experiment of this chapter show that it is possible to replace knowledge-based POS features with induced VSM features in at least one module of a TTS front-end cascade with considerable success. The second experiment performed using features from the VSM of word types is designed to discover whether similar success can be achieved by applying these features to the task of decision-tree state-tying in place of POS features.

6.4.1 Experiment

Data and Training Procedure

For this experiment the data-set denoted RJS-1000 on page 53 is used. This same procedure is used for voices built for all conditions of this experiment: HTS-2010 (see Section 2.1.3). The only aspect varied between conditions is the type of context features used. Features relating to phrase, and to tone and accent (feature-sets P and T respectively in Table 2.1) are omitted from all voices; all voices make use of the standard features relating to phones, syllables and the utterance (feature-sets F, FC, S and U respectively in Table 2.1). The features that are varied across the three systems are specified in Table 6.4, which also assigns identifying codes to the systems.

Systems B and P are baseline and topline systems, respectively: B has no other features than those available to all systems, relating to phone, syllables and the utterance: it represents the best system that can be built without using word- and phrase-level features. To these basic features, P adds contexts derived from part of speech (POS) assigned by a high-quality tagger, TnT (Brants, 2000). Specifically, the contexts used by P are the POS of the current phone's word and those of its left and right neighbours. Sets of POS tags are also used for

these contexts to generalise over the many fine-grained tags of the Penn tag-set. (See Section 6.3.1 for a case where using a coarser set of tags gives improved results.) These sets, which include ones such as *verb* and *content word*, are manually specified with expert knowledge. Clusters of tags are specified with varying degrees of granularity, from the full Penn treebank set at the finest to the function–content word division at the coarsest.

System V substitutes three vectors of 50 VSM features each (again, for the current phone's word and those of its left and right neighbours) for the three POS contexts of system P. The same word space described in Section 6.2 and used for Experiment 1 in this chapter is used here. However, the decision-tree building routine used for context clustering in HTS uses a manually-specified question set rather than using the full *standard question set* (Breiman et al., 1993, §2.4.1) that can be inferred from the data.[7] Assembling the full question set explicitly would result in an unwieldy question set, and would prove too computationally expensive for decision tree state-tying.[8] Therefore, each dimension of the VSM is discretised: the maximum and minimum values in the VSM look-up table are found for each dimension, and values along that dimension are binned into 50 bins of uniform width in that range. The questions incorporated into the question set refer to the 49 sets of (consecutive) bins that represent splits on each of the 50 dimensions used.

Evaluation

Objective evaluation of the three voices described was conducted using the labels and audio of 100 utterances from the same corpus that had been held out during training. Similarity measures were computed between the natural speech of these utterances and speech synthesised from the corresponding time-aligned transcription. To make comparison of natural and synthesised parameters on a frame-by-frame basis, the synthesiser was constrained to use natural phone durations. With the resulting frame-aligned natural and synthetic speech the following measures were computed: mean Bark-cepstral distortion per frame, and (over correctly voiced frames) Root Mean Square Error of F_0 and Pearson correlation coefficient of F_0. Segments labelled as pause and silence were excluded

[7] As the CART implementation *rpart* does.

[8] This is partly for reasons of implementation (*rpart* computes improvements in cluster purity efficiently, updating already computed statistics for similar previous splits rather than starting from scratch for each split) and partly because the data for state clustering is of higher dimensionality (the response is represented by vectors of state means and variances, not just a single symbol).

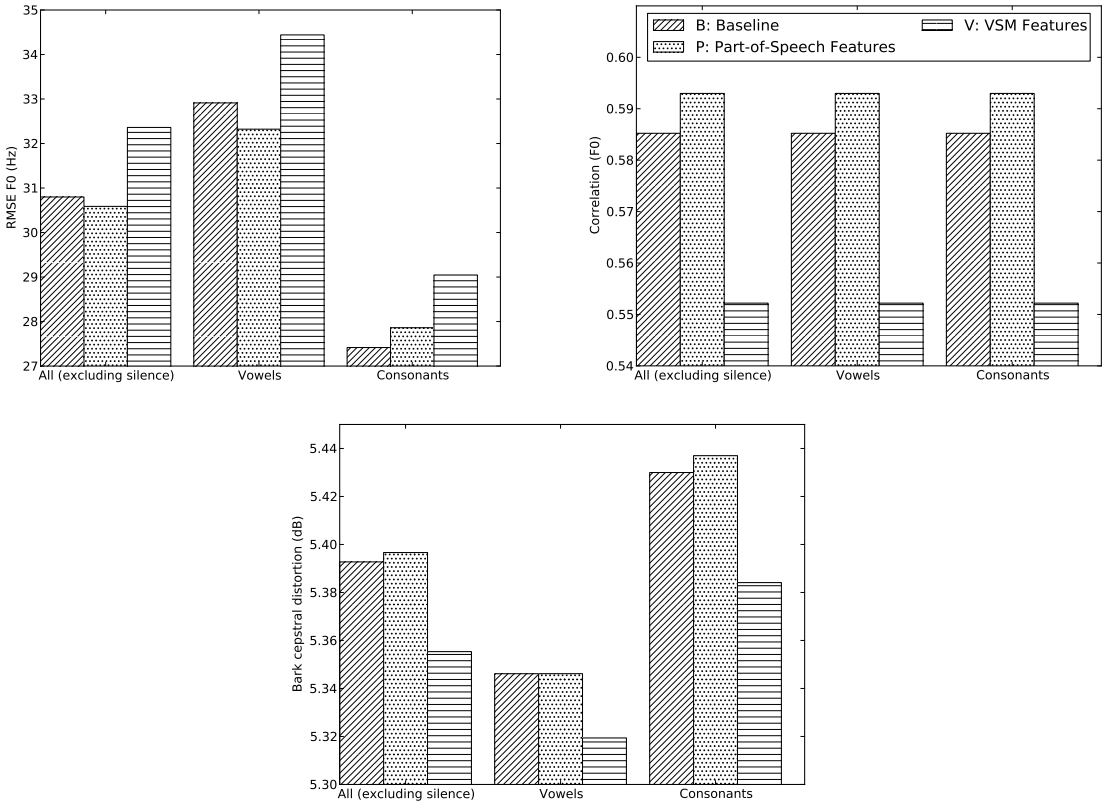


Figure 6.4: *Results of state-tying experiment*

throughout. As well as a general evaluation over all non-silent segments, measures were also computed separately over vowel and consonant segments. Results of this evaluation are shown in Figure 6.4.

The relation between **systems B and P** is as we would expect: part of speech has little effect (or a slightly negative effect in the case of consonants) on the spectral envelope measures, but improves F_0 measures consistently (with the exception of RMSE for consonants).

System V: The effects of using the VSM features are surprising: according to the objective measures, synthesised spectral envelope is improved, but the quality of synthesised F_0 trajectories is degraded. This is surprising because features modelling word level effects might be expected to improve prosody (which tends to be associated more with pitch than spectral envelope) rather than segmental quality (generally associated more with spectral envelope than pitch). Word-level VSM features might be expected to behave like POS features in this regard. Such is not the case as can be seen in Figure 6.4: where POS has a negligible affect on generated spectral envelope, VSM features improve it. Whereas POS improves F_0 generation, VSM features degrade it.

System	Number of questions defined
B	1889
P	2042
V	9389

Table 6.5: *Numbers of features used in the systems built.*

One possible explanation of the degrading effect of the VSM features on the F_0 trajectories generated by system V is the large number of contextual features used by this system. Despite the discretisation used as described, this system still has a very large number of features compared with B and P, as shown in Table 6.5.

Furthermore, in comparison with the POS features, the features derived from the VSM are noisy: it is to be expected that along each dimension of the space, many splits represent partitions of words having no linguistic or prosodic relevance at all. As well as increasing the time taken to build the trees, this large number of noisy attributes might have a degrading effect on the trees built. It is well known that decision-tree inducers are particularly sensitive to the degrading effects of noise (or noisy) attributes (John, 1997, pp. 68 ff.). This is due to the myopic partitioning strategy they use, where a split is chosen based only on the data points that have made it to the current node. The more attributes that define possible splits, the greater the chance that one will just happen to split the small training data well, regardless of whether that attribute represents noise or not (i.e. of whether the split chosen will generalise to new examples). This problem is expected to be more marked for the trees built for F_0 distributions, which are larger and so whose splits on average are chosen based on smaller sub-sets of the data.

6.5 Conclusions and Open Problems

The results of the first experiment of this chapter (phrase-break prediction) are pleasing. They show that most of the performance improvement between baseline and topline systems B and Tb/Tt, gained by incorporating knowledge-rich resources into the system via a state-of-the-art POS tagger, can alternatively be achieved by adding features extracted from plain text in an unsupervised manner (system U). Use of the unsupervised features also yields an improvement over System G where full part of speech tagging is approximated by the use of manu-

ally constructed word lists. Encouragingly, not only do the VSM-derived features give a quantitatively similar increase in performance over system B compared with the knowledge-based features of system Tt, but they appear to be used in a qualitatively comparable way (see the similarity of the trees' structures mentioned in Section 6.3.3). It seems, however, that the features differ enough that there is some complementarity between them: combining a tagger's features with the unsupervised ones leads to improvements over using either set of features in isolation, something *not* achieved by directly combining the features from 2 conventional taggers, where performance degrades. These results are encouraging because they show that at least some components of TTS systems can be built with very little manual effort, but very little degradation of state-of-the-art performance. The less restrictive data requirements for the VSM system mean that straightforward modifications – such as increased amount of training data – could improve performance further. This possibility is explored further in Chapter 7.

The use of VSM-derived features in the second experiment described (state tying) did not produce the results expected. The features improved the generation of spectral envelope where no such improvement was hypothesised, but degraded F_0 generation where improvement was expected. The explanation proposed, that the degradation caused to F_0 trees by these features is due to the features' noise and high-dimensionality combined with the trees' average small node size, is explored in Chapter 7. There the hypothesised problem is addressed by using a feature selection procedure that takes a more global view of the features before construction of the final trees even starts.

Chapter 7

Refinements and Extensions

The current chapter considers some refinements of – and an extension to – the vector space models of language that have already been presented. Two types of refinement are considered. First, the ability of the vector space model of words presented in Chapter 6 to exploit larger quantities of training data is examined. Also, various parameters were set somewhat arbitrarily in Chapter 6 and subsequently remained unchanged. Both the effect of larger sizes of training corpus and of different settings of these parameters are tested here, by revisiting Experiment 1 of Chapter 6 (phrase-break prediction). Second, a method of feature sub-set selection particularly appropriate for decision trees is used in combination with the vector space model of word types in an attempt to improve upon the results of Experiment 2 of Chapter 6 (state-tying). In addition, the extension of VSMs to higher levels of linguistic analysis than the word is considered: a model is formulated that is designed to capture characteristics of utterances that are relevant to TTS conversion.

7.1 Word Space: Corpus Size and Parameter Settings

One of the great attractions of the vector space model (and of unsupervised approaches in general) is that the size of a training corpus is limited only by the amount of plain text data that can be collected. There is no limit imposed by the need to manually annotate any data. As mentioned in Section 4.2.3, the development of incremental algorithms for singular value decomposition lifts restrictions on the dimensions of the matrices to be decomposed, and data incoming from a stream can be used to update a vector space incrementally. It is therefore easy

to imagine improving the performance of systems such as system U in Section 6.3 simply by increasing the size of the corpus from which the word representations are obtained. The experiment presented here tests whether this is true. Also tested is the effect of altering settings used in the procedure for building the vector space whose values were set arbitrarily in Section 6.2. Two settings – the value of n and the method for handling unseen words – were chosen for investigation here as it is expected that ideal values for these settings will vary with the size of corpus used.

7.1.1 Data and Vector Space Models

The text used for this experiment was taken from the news portion of the British National Corpus (BNC, 2007). Six progressively larger sub-sets were defined, each consisting of running text from the corpus and subsuming the previous subset. The sizes of these are 1, 5, 10, 15, 20 and 25 million words. Note that the smallest of these is similar in size to the Wall Street Journal text on which the model used in Section 6.3 was built, and so provides a point of comparison.

The method used in Section 6.3 to determine the set of word types on which to train the representation for unseen words will here be called *unseen method A*. To recap, tokens in a 1% portion of the data whose types are not seen in the remaining 99% are rewritten with the *unseen* symbol. Rather than adjusting the percentage used by this method, a more straightfoward method was used here for comparison, which will be called *unseen method B*. In this method, all types that are seen t or fewer times in the training corpus are rewritten with the *unseen* symbol. Four different values of t were tested: 1, 2, 5 and 10. *Unseen method B* using these four values of t will be denoted B_1, B_2, B_5 and B_{10}, respectively.

The other parameter whose value was varied in these experiments was n: the size of context vocabulary (number of types with which left and right co-occurrence is counted). Besides the value of 250 used in Section 6.3, four further values were tested: 500, 750, 1000 and 2000.

Word type models were induced using all 150 combinations of these 6 corpus sizes, 5 unseen type methods, and 5 values of n. In all respects other than these three settings, the procedure used to build the models was identical (see Section 4.2.4). The usefulness of these spaces was then tested by incorporating them into phrase-break prediction systems like System U in Section 6.3. In all respects other than the word representations used, the training and testing of these systems are identical to that of System U in Section 6.3. To summarise:

Table 7.1: *F measure (%) on phrase-breaks of SEC/MARSEC test set for 1) different methods of handling unseen words, 2) different values of n (number of context types) and 3) different sizes of text corpus. Means and standard deviations (±) over 10 runs of CART building are given for each configuration. Means shown in italics are plotted in Figure 7.1, and means shown in bold type are plotted in Figure 7.2.*

Unseen method: A					
# words	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$
1,000,000	*78.08* (±1.27)	77.27 (±0.16)	75.35 (±1.03)	78.12 (±1.18)	76.01 (±2.33)
5,000,000	*77.05* (±1.26)	75.53 (±0.23)	76.94 (±2.24)	74.67 (±0)	76.86 (±0.63)
10,000,000	*77.8* (±0)	77.2 (±1.57)	78.01 (±0.12)	78.09 (±1.86)	77.27 (±1.05)
15,000,000	*78.77* (±0.3)	76.92 (±1.82)	76.37 (±0.63)	77.8 (±0.1)	77.11 (±2.45)
20,000,000	*78.34* (±1.35)	76.05 (±0.51)	77.97 (±0.65)	76.4 (±0.71)	76.75 (±0.93)
25,000,000	*79.02* (±1.13)	76.58 (±1.1)	77.82 (±2.01)	77.9 (±1.59)	75.38 (±1.5)
Unseen method: B_1					
# words	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$
1,000,000	*78.73* (±0)	78.12 (±0.99)	78.17 (±0)	78.17 (±0)	78.09 (±0.21)
5,000,000	*78.55* (±0.26)	77.11 (±2.03)	76.65 (±0.01)	76.71 (±0.01)	78.69 (±1.37)
10,000,000	*78.36* (±1.12)	79.03 (±0.5)	78.38 (±0.51)	79.3 (±1.12)	79.08 (±0.44)
15,000,000	*78.18* (±1.41)	78.11 (±0.52)	78.24 (±0.72)	78.5 (±0.16)	77.68 (±1.05)
20,000,000	*77.05* (±0.82)	77.43 (±0.53)	77.8 (±0.68)	78.27 (±0.32)	78.31 (±0.23)
25,000,000	*78.05* (±1)	78.39 (±1.57)	78.19 (±1.83)	78.66 (±1.17)	75.92 (±0.28)
Unseen method: B_2					
# words	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$
1,000,000	*78.38* (±0)	78.27 (±0.19)	78.12 (±0.71)	78.34 (±0.12)	78.32 (±0.16)
5,000,000	*77.72* (±3.53)	79.22 (±0.49)	79.19 (±0.71)	79.25 (±0.18)	79.39 (±0.47)
10,000,000	*78.76* (±0.58)	78.33 (±0.48)	79.67 (±1.51)	79.34 (±1.12)	78.08 (±0.65)
15,000,000	*77.91* (±0.91)	77.65 (±1.65)	77.94 (±1.07)	77.96 (±3.37)	77.19 (±2.56)
20,000,000	*77.87* (±1.53)	78.45 (±1.19)	78.35 (±1.35)	78.46 (±0.81)	78.82 (±0.42)
25,000,000	*77.96* (±0.97)	78.42 (±0.46)	78.9 (±0.32)	78.23 (±0.13)	77.96 (±2.43)
Unseen method: B_5					
# words	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$
1,000,000	***78.12*** (±0.81)	**78.06** (±0.01)	**77.21** (±2.82)	**77.86** (±1.41)	**77.9** (±0.25)
5,000,000	***78.1*** (±0.57)	**78.06** (±0.21)	**77.84** (±0.26)	**77.88** (±0.52)	**78.19** (±1.22)
10,000,000	***79.12*** (±1.3)	**78.97** (±1.02)	**78.15** (±2.92)	**78.81** (±1.25)	**78.4** (±2.87)
15,000,000	***79.04*** (±0.77)	**78.88** (±0)	**78.91** (±0.03)	**78.92** (±0)	**78.91** (±0.7)
20,000,000	***79.01*** (±0.26)	**78.95** (±0.12)	**79.02** (±0.66)	**78.97** (±0.15)	**79.04** (±0.39)
25,000,000	***79.13*** (±0)	**79.11** (±0.65)	**79.12** (±0.34)	**79.1** (±0.52)	**79.15** (±0.61)
Unseen method: B_{10}					
# words	$n = 250$	$n = 500$	$n = 750$	$n = 1000$	$n = 2000$
1,000,000	*77.79* (±1.54)	78.27 (±1.73)	77.12 (±1.2)	78.57 (±1.27)	79.06 (±0.75)
5,000,000	*78.72* (±0.66)	78.79 (±0.44)	78.87 (±0.29)	78.81 (±0.41)	78.72 (±0.32)
10,000,000	*78.69* (±0.09)	78.8 (±0.28)	79.1 (±1.73)	79.21 (±0.99)	79.15 (±0.51)
15,000,000	*78.24* (±0.47)	78.61 (±1.48)	78.39 (±1.64)	78.17 (±0)	78.52 (±1.04)
20,000,000	*78.93* (±1.72)	79.17 (±1.76)	78.97 (±1.69)	78.74 (±1.27)	79.01 (±1.93)
25,000,000	*78.49* (±4.35)	79.19 (±1.67)	78.9 (±2.4)	78.93 (±2.42)	78.31 (±3.71)

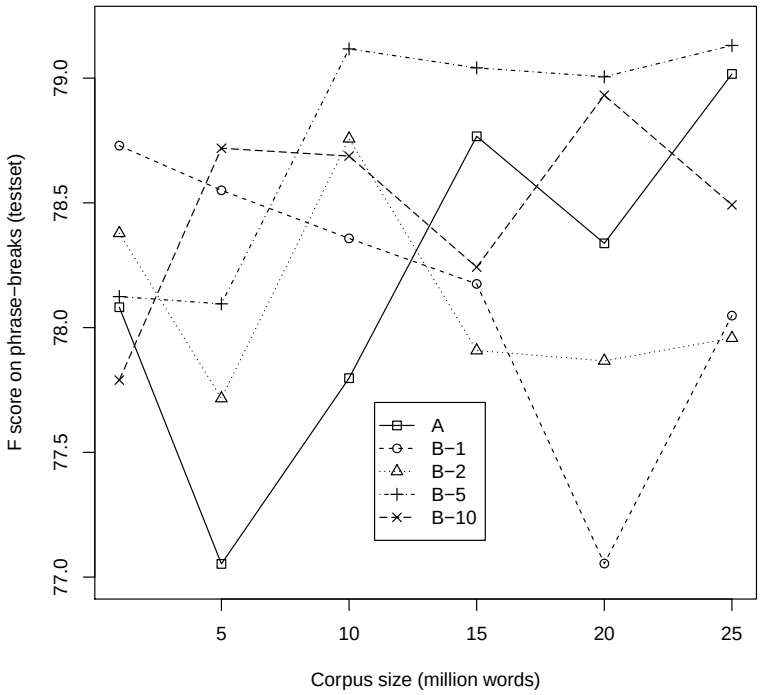


Figure 7.1: *The effect of varying method for handling unseen words (shown in the legend); n is held constant at 250 for this plot. (The values plotted here are the mean values shown in italics in Table 7.1.)*

phrase-break predictors were trained and tested on disjoint sets of data from the MARSEC/SEC corpus. The predictors are CART trees, which are fully grown and then pruned using a complexity parameter obtained by 10-fold cross validation.

7.1.2 Results

The performance of the 150 configurations tested on the same set of SEC/MARSEC data used in Section 6.3 is given in Table 7.1. Mean F measures and standard deviations over 10 runs of CART building are given as in Section 6.3. Some key points of these results are presented graphically in Figures 7.1 and 7.2.

Figure 7.1 plots the left-hand column of mean values which is shown in italics in Table 7.1. It shows the effect of varying the method used for handling useen words and size of corpus used to train the vector space model. The value of n is held constant in this plot (at 250). It is clear from this plot that most of these methods are very sensitive to corpus size, and that some methods are suited to a particular size of corpus. Method B_5 gives the most stable results. Although it is not the best-scoring method with smaller sizes of corpus, with corpora of 10

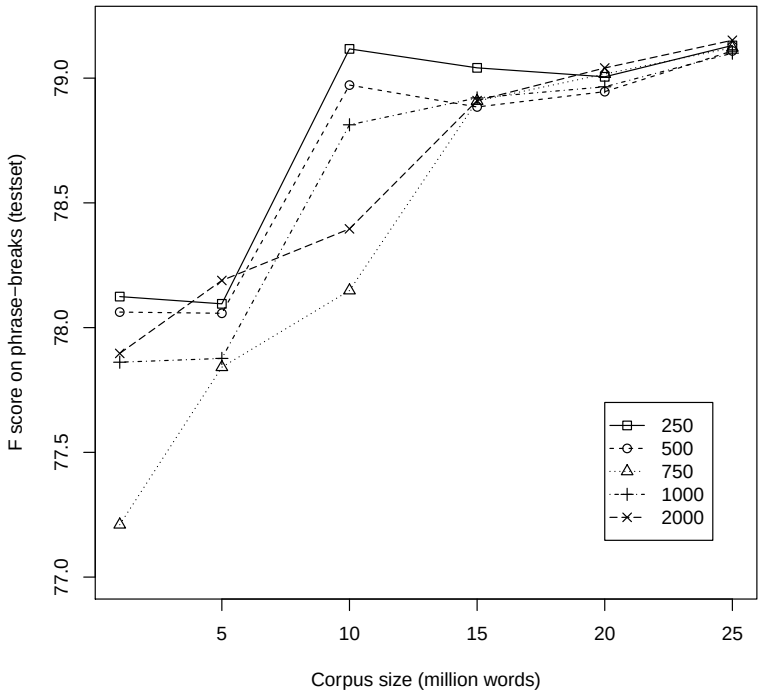


Figure 7.2: *The effect of varying context vocabulary size (n: different values of this parameter are shown in the legend); unseen method is held constant (B_5) for this plot. (The values plotted here are the mean values shown in bold type in Table 7.1.)*

million or more words it outperforms the other methods. It is also the method for which performance does not depart far from a monotonic improvement as a function of corpus size.

In Figure 7.2, the unseen method is kept fixed (at B_5), and the value of parameter n and the size of training corpus are varied. The plot represents the rows of mean values that are shown in bold type in Table 7.1. The plot shows that the size of context vocabulary (n) is more critical for optimal performance with small corpus sizes. When the corpus size is increased to 15 million words and beyond, the difference in performance between different systems is much smaller. A context vocabulary size of 250 is better with smaller corpus sizes. A larger value of n (500) is marginally better when 20 and 25 million words are used. This might indicate a trend that larger values of n become better as corpus size increases. This might be expected, given that larger context vectors will be less sparse as more data are observed.

Figure 7.3 repeats scores for the baseline, best performing topline, and VSM-based systems (systems B, Tt and U respectively) already presented in Section 6.3. In addition, the system U′ is added, taken from among the better-performing

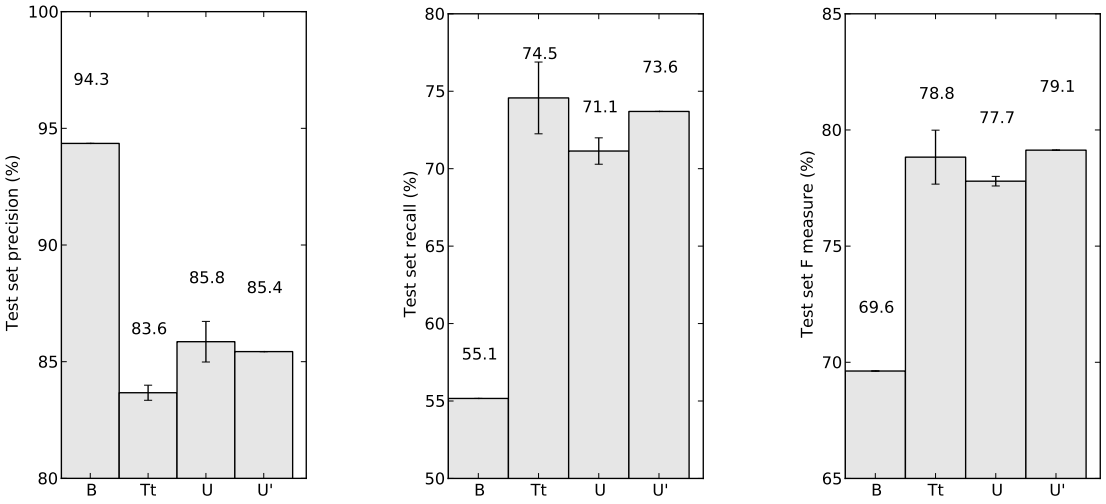


Figure 7.3: *Precision, recall, and F-measure of phrase-break predictors on test set. All results except those of system U′ were presented in Section 6.3.3, but are reprinted here for comparison. Means and standard deviations (error bars) over 10 repetitions of tree building are given.*

systems whose results are plotted above. U′ is the system whose VSM is trained on 25,000,000 words of news-text, uses 250 for its value of n, and uses unseen method B_5. It can be seen that simply by increasing the amount of text on which U's VSM is trained and adjusting the configuration used for VSM training can improve system performance. Moving from system U to U′, the systems using induced word representations overtake the topline system (in terms of mean F measure). Naturally, retraining system Tt's POS tagger on a comparably larger annotated corpus (rather than the c. 1.2 million words on which it had actually been trained) would also be expected to improve its performance, but the difficulty of acquiring such large amounts of annotated data represents a major limitation, compared with the ease with which unannotated text can be collected for many languages. System U′ makes use of 25 million words for training its VSM, but training on much larger quantities is a trivial extension, and the upwards-trend of the right-hand side of Figure 7.2 suggests that larger data-sets could yield further improvement to system performance. Furthermore, the VSM-derived features give much more stable results on the phrase-break task than the features used by system Tt, which are derived from the *TnT* tagger. That is, the same CART structure is produced over 10 runs of training for system U′, resulting in identical predictions and thus a standard deviation of 0.0 for this system's scores.

7.2 Feature Selection for HMM TTS

The use of features derived from a vector space model of word types for the task of state-tying in Section 6.4 produced unexpected results. The features improved the generation of spectral envelope where no such improvement was hypothesised, but degraded F_0 generation where improvement was expected. Section 6.4 tentatively attributed the degradation of the F_0 model caused by the VSM-derived features to the great quantity of those features and to the noise contained in them. F_0 is expected to be particularly susceptible to these characteristics of the features because the trees with which it is modelled tend to be deep. Here, this hypothesised problem is explored, and a feature selection procedure is proposed that is designed to address it. In essence it does this by taking a less myopic view of the data than that allowed to those nodes of decision trees far from the root, where the set of data upon which a split is to be chosen is much smaller than the whole data-set.

The selection of a good subset of features from among those available for input into a machine learning algorithm is a much researched topic, of which Guyon and Elisseeff (2003) give a review. The motivations for applying feature selection are various: for visualisation or interpretability for example, to save computational cost or to achieve better prediction. The latter is the motivation for its use here. The current task is to select a subset from among the 9,389 text-derived features available to system V in Section 6.4, then to use that sub-set to build a model which better predicts the acoustics – particularly F_0 – of the held-out set. The method that is used here is based on elements of the one presented in Tuv et al. (2009), which uses random forests for feature sub-set selection. Ensembles of trees are introduced, after which their application to feature selection in the work presented here is explained.

7.2.1 Ensembles of Trees

Predictive models such as decision trees can be used in *ensembles*. In an ensemble, predictions of many simple models are combined, typically by having the members of the ensemble vote on the class in a classification problem, or taking the average of their predictions in a regression problem. It can be seen from the error bars in Figures 6.1 and 6.2 that decision trees are inherently unstable. The variance on the results presented in those plots derives from the fact that the 10 trees contributing to each of those results were pruned to varying depths, which in turn

derived from the fact that a random partition of the data was used to determine pruning parameters. If two trees are actually *trained* on two randomly perturbed versions of the same training data, small differences in the versions of the data used can create very different tree structures, and different predictions on new data. This instability means that tree-based methods are well-suited to ensemble learning, as small variations in the data mean that different trees will model it in very different ways, and each of the members of the ensemble will specialise in some particular aspects of the data.

Decision trees have therefore formed the basis for several very successful ensemble methods. Many of the differences between these methods are in how the data are perturbed to force the induction of a varied set of trees. In *bagging* (Breiman, 1996), each of the trees in the ensemble is grown on a different bootstrap sample of the training data. That is, for each tree to be grown, a new data-set is created by sampling with replacement from the original data-set, until the new data-set is the same size as the original. In the *random subspace* method (Ho, 1998), trees are grown on all instances in the original training set, but using a tree-wise random subset of the independent variables. The *random forest* of Breiman (2001) combines both these approaches: each tree is grown on a bootstrap replicate of the training data, and splits are chosen from among a subset of independent variables randomly selected at each node. All such methods employing ensembles of trees typically yield improvements in prediction over single-tree methods. Of more relevance in the present context, however, is a by-product of such forests: variable importance measures.

7.2.2 Variable Importance Measures

Variable importance measures can be computed from an ensemble of trees in various ways. Breiman (2001, §10) suggests a method based on *out-of-bag* estimates of misclassification. In an *out-of-bag* estimate, the instances of the training set are classified by the forest that has been learned on it, but each tree votes only on those members of the training set not present in the replicate on which it was grown. Scoring predictions made in this way against class labels gives the out-of-bag misclassification rate. Before the importance of any variable can be assessed, the out-of-bag misclassification score is computed for the model. Then, to measure the importance of variable i, the values which that variable takes over the dataset are randomly permuted, and out-of-bag misclassification is recomputed. The change in misclassification after variable i's values are permuted

gives a measure of that variable's importance. This is repeated for each variable in the model, after which they can be ranked by importance score. A simpler measure of variable importance can be obtained by summing the improvement in the splitting criterion for each variable over all splits made in building the forest using that variable. Both methods are mentioned and their results compared graphically in Hastie et al. (2009, pp. 593–4).

The second type of variable importance described forms the basis of the feature subset selection method presented in Tuv et al. (2009). The main innovation there is the introduction of significance testing to determine not only a ranking of variables by their importance, but a subset of significantly important variables. The mechanism which allows significance testing – artificial contrasts – is adopted here. Small alterations are made to the way these contrasts are used, however, which will be mentioned below. Also, a major element of the method presented in Tuv et al. (2009) that is designed to handle variable masking effects is not incorporated into the system presented here.

7.2.3 Feature Selection for the Distributional–Acoustic Method

The feature selection developed for the present work will now be explained, followed by a presentation of the experiment in which it was applied to TTS.

The Learning Sample

As outlined in Section 2.1.2, acoustic models which are to be clustered consist of distributions over acoustic values associated with a set of linguistic–prosodic attribute values for c unique contexts. These distributions represent the data in the training set, and might generally be thought of as a synthesis model, but in the present context will be termed the *learning sample* for a decision tree. This learning sample, which will here be denoted $\boldsymbol{Z}$, is of the form $\{(\boldsymbol{x}_1, \gamma_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \ldots (\boldsymbol{x}_c, \gamma_c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)\}$, where $\boldsymbol{x}_j$ is a q-dimensional vector of binary values for the j^{th} context, whose k^{th} element specifies whether the k^{th} linguistic–prosodic attribute is true or false of context j. $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are (possibly poorly estimated) parameters for distributions over acoustic values associated with the j^{th} context. γ_j is the quantity of frames of the training corpus attributed to context j in a forward–backward alignment of the model with the observations. Because this is a soft alignment, γ_j for a given context j will typically not be a whole number.

Bootstrap Resampling

Because – as already mentioned in Section 2.1.2 – each item in this learning set represents a distribution over a varying number of observed values rather than an observed value itself, bootstrap resampling cannot proceed (as in Breiman (2001) and Tuv et al. (2009)) on a point-by-point basis. Instead, the learning sample is perturbed in each replicate by adjusting the values $(\gamma_1 ... \gamma_c)$. The elements of sequence $(\gamma_1 ... \gamma_c)$ of the original learning set are normalised, so that for each element j, the normalised element is:

$$\frac{\gamma_j}{\sum_{i=1}^{c} \gamma_i} \tag{7.1}$$

The elements of this normalised sequence are then treated as the event probabilities of a categorical distribution with c outcomes. A perturbed version of $(\gamma_1 ... \gamma_c)$ is created by initialising a sequence where all values are 0.0, drawing n times from this distribution, and where the outcome of the draw is j, incrementing element j of the new sequence by some fixed quantity.[1]

Where a forest of t trees is to be grown, the learning set $\boldsymbol{Z}$ is duplicated k times to create a sequence of learning sets, $\{\boldsymbol{Z}_1 ... \boldsymbol{Z}_t\}$. In each of these replicated learning sets, a bootstrapped set of values for $\{\gamma_1 \dots \gamma_c\}$ is created as explained above. All other elements of the original learning sample (linguistic contexts, mean vectors, covariance matrices) are duplicated exactly in each replicate of the learning set. To build a standard ensemble of trees, tree building would then proceed independently for each of the t new learning sets, resulting in a ensemble of t trees.

Artificial Contrast Variables

In the present experiment, however, artificial contrast variables were added before building the forest. $\boldsymbol{X}$ is a $c \times q$ matrix whose rows are $\{\boldsymbol{x}_1 ... \boldsymbol{x}_c\}$; $\boldsymbol{x}_{jk}$ therefore presents the value (true or false) of the k^{th} linguistic attribute for the j^{th} context. $\boldsymbol{X'}$ is also a $c \times q$ matrix whose k^{th} column is generated by random permutation of the k^{th} column of $\boldsymbol{X}$. Column k of $\boldsymbol{X'}$ represents an artificial variable having the same distribution as the real variable represented by column k of $\boldsymbol{X}$, but only a

[1] This quantity will be $\frac{1}{(\sum_{i=1}^{c} \gamma_i) n}$ if it is desired that the replicate of the learning set should consist of the same number of frames as the original, although for present purposes this is not important. Note that the bigger the value of n used, the finer the quantisation of state occupancy used in the resampling. For this experiment, n was set to $100(\sum_{i=1}^{c} \gamma_i)$, i.e. $\frac{1}{100}$ frame units were used.

chance relationship with the distributions over acoustics. For each context j in a replicate, q-dimensional vector $\boldsymbol{x}_j$ of linguistic–prosodic values is appended with another q-dimensional vector, row j of $\boldsymbol{X'}$. The resulting $2q$-dimensional vector represents the true–false values for q linguistic variables and for q corresponding contrast variables. The randomisation to produce artificial contrasts is performed for each of the t trees in the ensemble.

Feature Selection

The ensemble of trees is then built: tree construction proceeds independently for each of the t replicates of the learning set with added contrast variables, resulting in an ensemble of t trees. For the present experiment, 200 trees were built for each of five states, both for spectral envelope model trees and trees modelling fundamental frequency. Trees in the forest are grown as far as they can be. As the ensemble is built, a record is kept of the log-likelihood improvement given by each split made; these improvements are summed within each tree for each of the q linguistic attributes and each of the corresponding contrast attributes.

In the present experiment, the ensemble is used only for the purpose of feature selection. The trees themselves will not be used to make predictions and can be discarded. All that is retained are two $t \times q$ matrices, $\boldsymbol{R}$ and $\boldsymbol{R'}$, where $\boldsymbol{r}_{ij}$ records the sum of improvements to log likelihood of the training data yielded by linguistic attribute j in tree i, and $\boldsymbol{r'}_{ij}$ records the corresponding sum for the contrast feature corresponding to attribute j in the same tree i.

Using these two matrices, feature sub-set selection is performed by hypothesis testing. For each feature j, hypothesis H_1 is that j contributes more to log likelihood improvement during forest training than its counterpart contrast feature j' (which shares j's distribution but bears only chance relationship to the acoustics). The null hypothesis H_0 is that j and its permuted counterpart j' contribute the same amount to log likelihood improvement. In the present experiment, the hypothesis was tested for each variable j of q variables using Wilcoxon's signed rank test (Wilcoxon, 1945) over the paired values of columns j of $\boldsymbol{R}$ and $\boldsymbol{R'}$. The acceptable probability of false rejection of the null hypothesis (α) is set to 0.05 for this experiment; Bonferroni correction is used to account for the number of tests being made ($\frac{\alpha}{q}$ is used). The sub-set of features found in this way to give an improvement in forest building that is significantly better than the improvement given by the permuted counterpart is selected for the next stage of training. The final trees that are actually to be retained for speech synthesis are then built

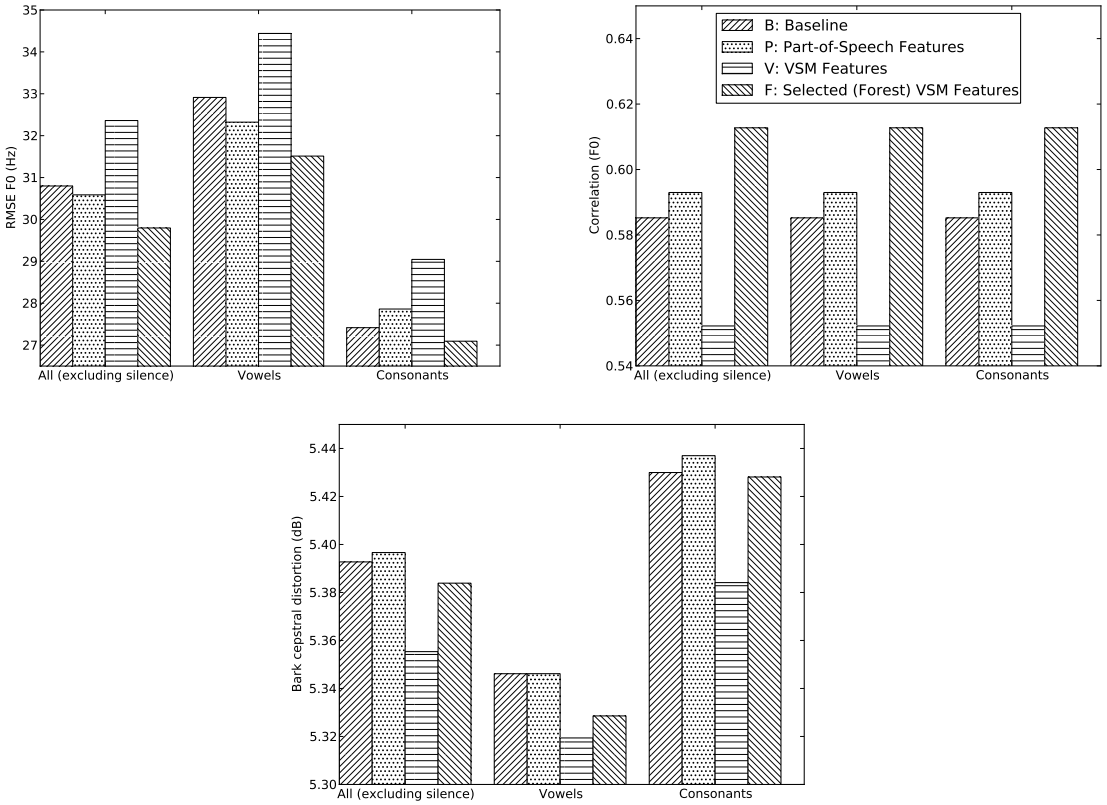


Figure 7.4: *Results of state-tying experiment: for all systems except F (using feature selection) results are repeated from Figure 6.4*

using this subset of features.

Modifications

Note the following differences between the procedure outlined in Tuv et al. (2006, 2009) and the one presented here. For reasons already mentioned, the bootstrap resampling takes a different form. Wilcoxon's signed rank test is used here rather than Student's t-test because the data do not meet the assumptions of that test (e.g. in many cases they are not normally distributed). Each row of $\boldsymbol{R}$ corresponds in the present experiment to a single tree, whereas in Tuv et al. (2006, 2009), several trees are used for each comparison. Tuv et al. (2006) motivates this as a way to avoid zero scores, where some variable is not used at all in a given tree. Such zero values are not an issue where the Wilcoxon test – which considers only rank and not absolute values – is used. In Tuv et al. (2006, 2009), it is reported that the 75th percentile of all contrast variables was used in place of individual contrast variables' scores; in the present experiment, the columns of $\boldsymbol{R}$ and $\boldsymbol{R}'$ were compared directly.

7.2.4 Experiment: Word VSM for State-Tying, Revisited

System F was built in the following way to test the usefulness of the feature selection method that has been outlined. Five iterations of clustering were performed exactly as for System V in Section 6.4. Using the resulting models, feature selection was performed as described. For the first 5 iterations of training, the system had the same 9389 features available to it as System V. A sub-set of 1736 of these (18.5% of the original set) was chosen by feature selection. This sub-set was used for the remaining 5 iterations of training. Note that Systems V and F received a comparable amount of training of the models that are finally used for synthesis.

Evaluation was conducted following the same procedure as in Section 6.4. Results of the evaluation of System F are plotted in Figure 7.4; the scores for topline, baseline, and VSM system (without feature selection: systems B, P and V respectively) are re-plotted here for comparison.

It can be seen that incorporating feature selection into the vector space model based system greatly improves its performance on the F_0 measures: this was the expected result. F is the best-performing system in the plot, achieving lower RMSE of F_0 and higher F_0 correlation than even the topline system incorporating a high-quality tagger. For spectral envelope, F still outperforms the topline system; however, much of the unexpected gain achieved by system V over the benchmarks is lost after feature selection is applied.

7.2.5 Conclusion

The experiment presented in this section has modified an existing method of feature sub-set selection and applied it to speech synthesis for the first time. In it, an ensemble of trees is grown and used to find linguistic features that are irrelevant to the task of clustering distributions over acoustics. A global approach is thus taken to evaluating the relevance of features; this can be contrasted with the local, short-sighted approach used by a decision tree inducer when selecting a feature for splitting a node. By excluding features found to be irrelevant ahead of decision tree building, the problem presented by this short-sightedness is mitigated.

There are design choices where optimal values have not been established by the experiment presented here. For example, a single set of features is selected over the entire forest (trees for all states, for both fundamental frequency and spectral envelope), and applied to all parts of the synthesis model. Many different variations can be envisaged that might prove beneficial: feature sets could be

selected for example from a specific state position or type of tree, and used for building the corresponding part of the synthesis model. It is almost certain, for example, that different subsets of the feature-set are useful for modelling spectral envelope and F_0. Incorporating this notion into the design of a set of sub-forests might therefore give improved results. Not doing so may account for the fact that single-forest selection improves the F_0 of system F but degrades spectral envelope (in comparisons with system V).

The success of the feature selection used on F_0 shows that the distributional features that are derived from the VSM of word types are useful for the task of state-tying just as they were shown to be in Section 6.3. However, they contain many features that are irrelevant, and this is harmful to large decision trees such as the ones used for modelling F_0 distributions if these irrelevant features are not removed.

The experiment also suggests that feature selection with ensembles of trees could have wider application than winnowing the chaff from VSM-derived features. 18.5% of the total feature set was selected; this includes 13.5% of the 7500 features derived from the VSM of words, but also 38.5% of the 1889 baseline features. That is, more than half of the conventional linguistic features used were excluded by the forest. This suggests that the method of feature selection developed here might also be useful in the case where only conventional linguistic features are used.

7.3 An Utterance Space Model

In Chapter 5 the vector space model was applied at a sub-syllabic level of analysis, and in Chapter 6, extended to the word level. The current section continues this trend by considering a level of linguistic analysis above the word level: the level of the utterance. This part of the thesis is particularly relevant given current trends in TTS research, where large, continuously-recorded databases are starting to be tackled, in which utterances occur in some genuine discourse context (Prahallad and Black, 2010; Braunschweiler and Buchholz, 2011; Székely et al., 2011). As the vector space model approach to characterising textual entities was originally formulated in Information Retrieval for use at the document level, it is ideally suited to this level of analysis, and the work presented here might be viewed as a homecoming for the vector space model.

However, the sort of characterisation of utterances that is expected to be useful

for TTS is different from the characterisation of documents ideal for Information Retrieval. Two utterances can have very different semantic content, but share the same utterance type. There is no well established classification of utterances into types reflecting their role in a text or dialogue, but many typologies and annotation schemes have been proposed both for utterance-type units themselves (e.g. Carletta et al., 1997; Bunt et al., 2010), and for the relations that can hold between them (e.g. Mann and Thompson, 1988). For both types of scheme, experiments have shown that at least some of the distinctions made in the sets of classes have acoustic correlates (e.g. Surendran and Levow, 2006; Murray et al., 2006). It seems obvious, then, that some sort of characterisation of utterances will prove beneficial for TTS. An experiment was therefore conducted to test whether one obvious division of utterances – into questions and non-questions – can arise from the distributional–acoustic analysis proposed in this thesis.

7.3.1 A Vector Space Model of Utterances

The experiment presented here seeks to determine whether a simple division of sentences into questions and non-questions can be expressed by a space trained in an unsupervised way without access to the labelling of these two classes. To this end, a vector space model of sentences was built. A small corpus was used for this experiment, as its aim is proof of concept, and it is not incorporated into any synthesis system. To build this vector space, the first 10,000 sentences of a corpus of 3 million sentences of web-scraped text made publicly available in the Leipzig Corpora Collection were used (Quasthoff et al., 2006).[2] The sentences were tokenised and used to build a dynamic vector space of the type described in Section 4.2.4. To recap, all words occurring more than once in the corpus were retained in the model, and no stop words are excluded. In the present case, this results in a vocabulary size of 9,524 word types. A word–utterance matrix $\boldsymbol{C}$ is compiled, where $\boldsymbol{c}_{ij}$ is a count of vocabulary item i in the j^{th} utterance of the corpus. Term-frequency inverse document-frequency weighting is applied to the raw co-occurrence matrix, and SVD is performed. For this experiment, 100 dimensions of the latent space were retained.

The fact that a characterisation of utterances in terms of topic is not sought here is reflected by a difference in how the space described here is constructed, compared with a classic Latent Semantic Indexing model. Namely, stop-word

[2]Specifically, the first 10,000 sentences of the corpus named *eng-au_web_2002_3M-text* were used.

Can Job Focus tell me what to do ?
Do I have to use the password provided by HeSA ?
Guess who backed everything else on Saturday including Ken Cheval but did n't take the trifecta ?
NEXTEP > What is DSL ?
I said to them , " what you fellas come here for ? "
How many people live in the house ?
" And which book did you read , Eddie ? "
Can you see any connections ?
Have you seen it on air ?
Does each paragraph deal with a coherent aspect of the topic ?
...

Table 7.2: *The first 10 sentences of the question-testset labelled as questions*

removal is not employed, as function words such as *however* and non-words such as *?* are expected to be most useful for characterising utterances in a way that is useful for TTS. Other modifications can easily be imagined: actively excluding infrequent words, for example, and including contexts from neighbouring sentences to incorporate discourse context of an utterance. Other context types than words might well be useful for this task; although using a *bag-of-phrases* instead of a *bag-of-words* approach has generally provided little benefit for Informational Retrieval (Koster and Seutter, 2003; Coenen et al., 2007), sequences such as *can you* and *in fact* would probably be more useful for utterance type classification than the constituent words in isolation. For this experiment, however, no such modifications were tested.

For this experiment, the final 10,000 sentences of the same corpus of web-scraped text were tokenised and used as unseen sentences. The space is dynamic: these newly encountered utterances are therefore projected into the latent space and are each thereby assigned a 100-dimensional representation.

7.3.2 Experiment: Separability of Questions and Non-Questions

For this experiment, a crude set of class labels was made for the 10,000 unseen sentences. Sentences are labelled as *question* or *non-question* simply on the basis of whether a question-mark is present in them. The first 10 sentences identified in this way as questions are listed in Table 7.2.

A sub-set of the 10,000 unseen sentences that will be called the *question*-testset was chosen as follows. The first 200 sentences labelled as questions and the first 200 labelled as non-questions were selected. Even numbers of questions and non-questions were used for the test set to yield clearer visualisations of the space and

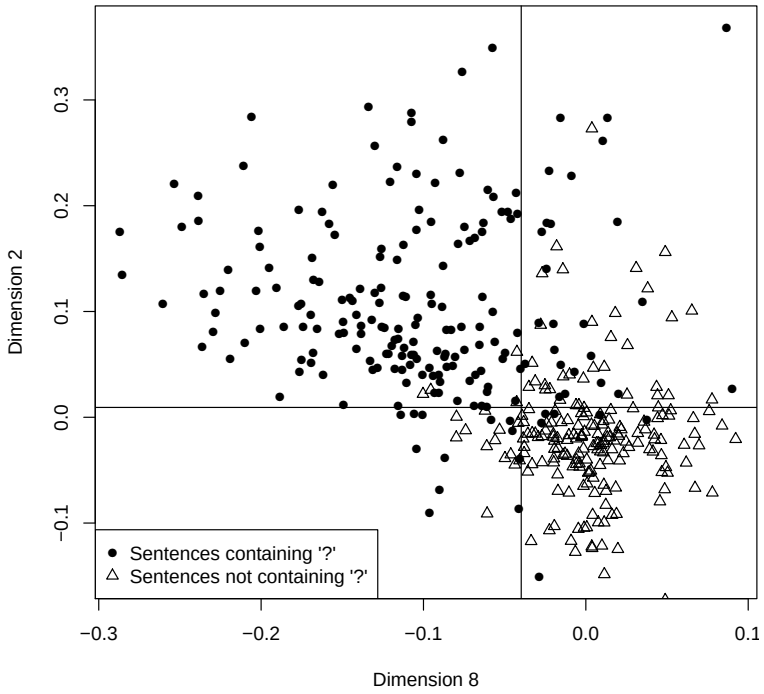


Figure 7.5: *The 400 points of the question-testset plotted against 2 dimensions of an induced utterance space.*

make the interpretation of classification accuracy obvious (50% accuracy is the chance level). As is to be expected, the distribution is not uniform in the training set, where exactly 3% of the sentences would be labelled as questions (contain the token *?*). In Figure 7.5, the 400 points of the *question*-testset are plotted on the 2 dimensions of the space in which the purest partitions of the two classes can be made. It can be seen that although the 2 groups are not perfectly separable with partitions orthogonal to either of the two latent axes represented in the plot, they do cluster in two groups in such a way that a fairly clean division can be made.

An obvious objection at this point is that the space could be performing an operation no more sophisticated than the one used to produce the class labels – that the two dimensions plotted simply reflect the presence or absence of the frequent token *?* in the sentences. To address this issue, the following control test was run. Another word type, present in a similar number of sentences of the training set as *?* was chosen, but which does not have the obvious connection with an utterance type that *?* has. The word type chosen was *time*, which occurs in 292 out of the 10,000 training sentences (*?* occurs in 300 of them). A second control test set (which will be called the *time*-testset) was created by taking the first 200 test sentences containing the token *time*, and the first 200 from which

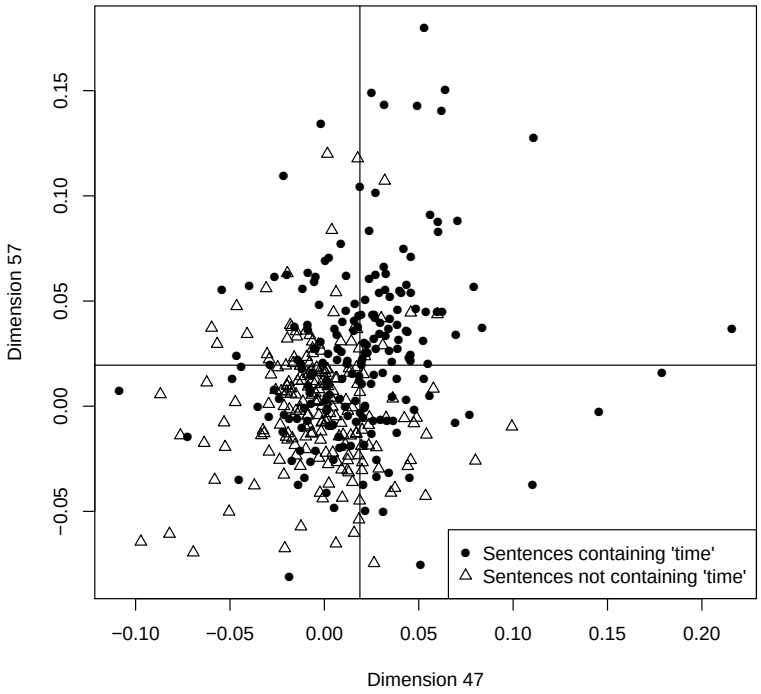


Figure 7.6: *The 400 points of the time-testset plotted against 2 dimensions of an induced utterance space.*

that token is absent. The two dimensions of these 400 utterances' representations in the latent space are plotted as before in Figure 7.6. Once again, the two dimensions chosen are the ones that allow the best and second best separation of the two classes with a single split orthogonal to those dimensions' axes. It is immediately obvious that the classes of the *time*-testset cannot be separated as cleanly as those of the *question*-testset.

To quantify this difference in separability, a series of decision stumps were built for both test-sets. These decision stumps are classification trees that make only a single split of the data, and thus have only two leaf nodes. The two stump-sequences were built as follows: the first stump selects the split on any dimension of the utterance space that best partitions the 2 classes (goodness of partition is measured with the Gini index). That split is recorded, then the dimension on which it is made is excluded from the data, and the procedure is repeated. In this way a ranked list of splits that separate the classes with decreasing effectiveness is obtained for each test-set. The first 2 stumps' decision boundaries for each test-set are represented by the vertical and horizontal lines respectively in Figures 7.5 and 7.6. Tables 7.3 and 7.4 give details of further splits of the *question*-testset and *time*-testset, respectively, and include accuracy scores.

Rank	Dim.	Split point	Accuracy
1	8	-0.040	88%
2	2	0.009	86%
3	13	0.035	84.75%
4	9	-0.055	83.5%
5	7	0.043	77.5%

Table 7.3: *Ranked stump accuracy for the question-testset.*

Rank	Dim.	Split point	Accuracy
1	57	0.020	67%
2	55	0.024	66.25%
3	56	-0.037	66%
4	69	0.006	65.5%
5	67	0.025	64.5%

Table 7.4: *Ranked stump accuracy for the time-testset.*

These tables confirm what can be seen clearly in Figures 7.5 and 7.6: that the space offers multiple ways to separate the question and non-question sentences much more accurately than for the *time* and non-*time* sentences. This is to be expected as the word *time* has no obvious association with utterance type in the way that *?* does. Even for the *time* set, however, the accuracy with which the space allows the two classes to be separated is above the chance level (50%, as the two classes are balanced in the test-sets). It appears therefore that the token *time* is bound up with some factors in the space's representation of utterances that are not obviously interpretable. But the better separation of the question set is obviously not simply attributable to the frequency of the token *?*: the space has done something non-trivial in its unsupervised separation of these classes. To gain some intuition about what it is doing, the weights (elements of basis vectors) of each dimension listed in Table 7.3 with the largest absolute values were found, and are shown in Table 7.5 next to their corresponding vocabulary items. The basis vectors for the latent space are the columns of the matrix called $\boldsymbol{U}$ in Section 4.1, whose rows correspond to contexts – in the case of the utterance space, the contexts are vocabulary items – and whose columns correspond to dimensions of the latent space. For each of these dimensions, *?* has a weight with a large absolute value, as expected. However, for each of these dimensions, the other weights of the same sign as *?* having the largest absolute values are obviously context vocabulary items that co-occur in various types of questions: dimension

Dimension 8		*Dimension 2*		*Dimension 13*		*Dimension 9*		*Dimension 7*	
Word	*Weight*	*Word*	*Weight*	*Word*	*Weight*	*Word*	*Weight*	*Word*	*Weight*
are	-0.387	you	0.488	:	-0.371	:	-0.375	be	-0.365
?	-0.352	(	-0.341	?	0.339	?	-0.366	i	-0.314
was	0.258	)	-0.340	are	-0.308	is	-0.350	:	-0.306
your	0.237	do	0.207	how	0.248	it	-0.309	you	0.256
how	-0.227	i	0.180	be	0.220	’	0.235	will	-0.218
you	0.227	?	0.177	;	0.192	“	0.232	?	0.207
he	0.184	your	0.174	is	-0.192	be	0.230	information	-0.183
it	0.179	if	0.174	&	0.185	are	0.209	has	0.143
they	-0.178	have	0.147	for	0.182	how	-0.182	in	0.141
on	0.166	of	-0.146	that	-0.171	will	0.142	have	0.139

Table 7.5: *Elements with largest absolute values of basis vectors for the five dimensions of the utterance space listed in Table 7.3.*

8 has *how*, *are* and *?*; dimension 2 has *you*, *do* and *i*, *?*; dimension 13 has *how*, *be* and *?*, etc.

It seems, then, that the space induced is able to separate the classes questions and non-questions because there are co-occurring sets of vocabulary that signal membership of these classes. It is reasonable to expect that other classes of utterances beside questions will be marked out by such patterns of co-occurrence, and therefore able to be discovered by such a VSM of utterances. Of course, many possible splits of many dimensions of such spaces will not represent any useful class of utterance. The method of feature-selection developed in Section 7.2 was shown to be able to exclude irrelevant features stemming from a VSM of word types. Such feature selection is also expected to be crucial to the useful employment of features from utterance spaces in acoustic modelling using decision trees.

Chapter 8

Evaluation of Entire Systems

8.1 Introduction

Up until this point of the thesis, the techniques described have been tested in a single language: English. The motivation for this is practical: multiplying the number of languages in which techniques are to be evaluated means multiplying the number of systems to be built and evaluations to be conducted, and therefore limits the number of novel techniques that can be considered. The use of a single target language in evaluations, however, brings with it the obvious inevitable disadvantage that those evaluations give no assurance that the techniques tested will be useful in other target languages.

Furthermore, the evaluation presented so far has been conducted in a piecemeal fashion, by exchanging isolated modules of a TTS system. That is, to test a novel component of a TTS system, only the module of a conventional benchmark system having a role most similar to that of the module to be tested is removed and the novel one put in its place. Other modules necessary to synthesise speech from text are left untouched. For example, in Chapter 6, a novel way of assigning representations to words – a vector space model of word types – was tested through the comparison of a benchmark system that employs a conventional part of speech tagger with one where that tagger is replaced with the vector space model. However, parts of the voices (such as the pronunciation lexicon) that concern subword-level features were left untouched and corresponded exactly in the two voices. The advantage of such an approach to evaluation is that it is controlled, and allows the contribution of the module in focus to be isolated. The closely related disadvantage is that it is quite an unrealistic test of system perfomance. In real-world scenarios it might be necessary to employ naive modules

at various levels of textual analysis. If detrimental interactions occur between naive modules on these different tiers of analysis, the module-by-module evaluation undertaken so far in this thesis will not reveal them. For this reason, it is also necessary to attempt to build what are here termed *end-to-end* systems: systems that make minimal use of expert knowledge and rely on naive modules on every level of analysis that they encompass.

The current chapter addresses the issues of language dependence and end-to-end design by detailing the construction and evaluation of end-to-end systems for three target languages: English, Romanian and Finnish. Romanian and Finnish were selected primarily for pragmatic reasons: the ready availability of high quality annotation for benchmarking purposes, and the ability to arrange subjective tests with native listeners. In these three target languages, both the Finno-Ugric and two branches (Romance and Germanic) of the Indo-European language families are represented. However, they do not demonstrate the kind of range of typologically diverse features mentioned in Section 2.3 as being necessary to claim *language independence* for a technique. Indeed, such a range would be impossible to achieve with so few languages. These experiments are not intended to make the claim that the methods involved are language independent. It is rather to show that despite being developed on a single language, the techniques outlined in this thesis are linguistically naive enough that they can be applied to further languages with no additional effort. Some details of characteristics of the languages relevant in connection with text-to-speech conversion will now be mentioned.

8.2 Language Characteristics

Like English, both Finnish and Romanian have alphabetic orthographies and mark word boundaries orthographically. As such, the assumptions made in Section 2.3 about the types of scripts whose characters can be used as units for acoustic modelling are clearly met. Both orthographies have a much more transparent phoneme–letter relationship than that of English. With respect to the 'careful normative pronunciation of the standard language', the orthography of Finnish defines a one-to-one mapping between letters and phonemes, with the single exception of /ŋŋ/ which is written ⟨ng⟩ (Karlsson, 2006). Romanian also employs a transparent orthography, although the correspondance of vowel sequences and the diphthongs and triphthongs represented by them is often not

straightforward, and neither is the pronunciation of some neologisms (Augerot, 2006; Stan et al., 2011).

Primary lexical stress is always on the first syllable of the word in Finnish (Karlsson, 2006); in Romanian (as in English) its location is harder to predict (Augerot, 2006; Stan et al., 2011).

As already mentioned, Finnish and Romanian both have orthographically-defined word units, which means that the implementation of vector space models of word types and of utterances like those described for English in Chapters 6 and 7 is straightforward. However, the inflexional morphology of both Romanian and, in particular, of Finnish is much richer than that of English. For example, the English noun *house* can take only two inflexional forms: *house* and *houses*. Inflexion for case and number and suffixation of the definite article in Romanian mean that the noun for 'house' in Romanian is seen in the following orthographic words: *casa*, *casă*, *case*, *casei*, *casele*, *caselor* (Augerot, 2006). Because Finnish suffixes morphemes for number (2 morphemes), case (14 morphemes), and possession (5 morphemes), the number of orthographic word types in which the 'house' morpheme can occur in Finnish is much larger. For example: *talo*, *talot*, *talossa*, *taloistani*, *talonne*, etc. ('house', 'houses', 'in (a/the) house', 'out of my houses', 'into your house', etc.) (Karlsson, 2006).

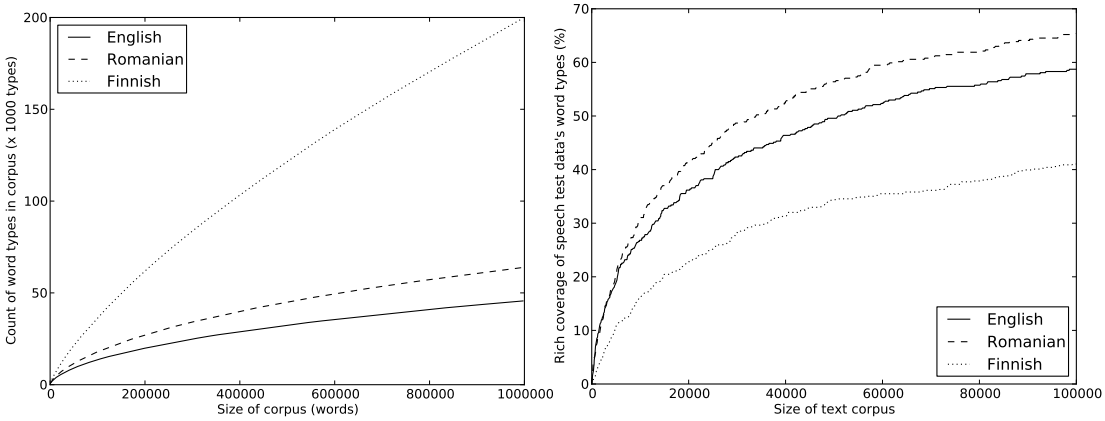


Figure 8.1: *Figures demonstrating varying morphological richness of the three target languages.*

These characteristics are illustrated quantitatively in Figure 8.1. The left-hand panel of Figure 8.1 is based on 1 million tokens of the news text collections described below in Section 8.4.2, and shows a running count of orthographic word types encountered plotted against the (gradually increasing) size of the sample (in word tokens). It can be seen that 1 million word tokens consists of very different

numbers of types in the three languages: nearly 49,000 for English, 67,000 for Romanian, and 260,000 for Finnish.

The right-hand panel of Figure 8.1 considers the *rich coverage* of word tokens in the first 800 words of the test sets described below in Section 8.5. A word is considered to be *richly covered* if it has been seen 20 times or more in the text corpus, reflecting the fact that it is probably necessary to observe a word type in several contexts to estimate useful features for it. Using 1 million words of news texts, 40% of Finnish tokens in the test set can be observed in 20 contexts, and this number is higher for Romanian and English (c. 65% and 60% respectively). The higher score for Romanian here is presumably due to the better match of the Romanian news corpus with its test sentences than is the case for English. The morphological richness of Finnish is predicted to be the biggest challenge in extending the word- and utterance-level features described in Chapters 6 and 7 to that language without the sort of unsupervised morphological analysis mentioned in Section 2.3.

8.3 Initial Hypotheses and Systems Built

These experiments are based around two sets of benchmarks: the performance of *baseline* systems and of *topline* systems. The baseline system for a language is the simplest system imaginable, built using minimal language-specific knowledge (i.e. knowledge of which characters are letters, which are punctuation, and which are word-delimiters), and a primary corpus of text and speech. The topline system for a language is one built in a more conventional way, making free use of language-specific expertise and manually annotated data. In Table 8.1, where systems built are given identifiers, topline systems are denoted {EN,FN,RM}-T-1, and baselines as {EN,FN,RM}-A-1.

The first hypothesis of these experiments is unlikely to be controversial:

> **Hypothesis 1**: The topline system of a given language will outperform that language's baseline.

Performance will be measured in terms of the results of listening tests for intelligibility and naturalness. The overarching second hypothesis of these experiments – to be broken down and expressed as several sub-hypotheses in the following discussion – is:

> **Hypothesis 2**: by utilising additional unannotated text data in the target language to build vector space models of elements at different granularities

	English		Finnish		Romanian	
Feature Set	Plain	Selection	Plain	Selection	Plain	Selection
A	EN-A-1	EN-A-2	FN-A-1	FN-A-2	RM-A-1	RM-A-2
B	EN-B-1	EN-B-2	FN-B-1	FN-B-2	RM-B-1	RM-B-2
C	EN-C-1	EN-C-2	FN-C-1	FN-C-2	RM-C-1	RM-C-2
D	EN-D-1	EN-D-2	FN-D-1	FN-D-2	RM-D-1	RM-D-2
T	EN-T-1	EN-T-2	FN-T-1	FN-T-2	RM-T-1	RM-T-2

Table 8.1: *Details of 30 end-to-end systems built. The codes EN, FN and RM in the system identifiers denote system target language (English, Finnish and Romanian, respectively). The codes A, B, C, D, and T indicate the feature set used by the voice, and are explained in Table 8.2. 1 and 2 indicate whether feature selection was used (2) or not (1).*

Feature	A	B	C	D	T
Letters and positions	X	X	X	X	
Letter VSM		X	X	X	
Word VSM			X	X	
Utterance VSM				X	
Topline features					X

Table 8.2: *Key to feature sets of Table 8.1.*

of analysis, it is possible to significantly close the gap between the baseline systems and the toplines.

To this end, VSMs are built in each language at the level of the letter (following the methods used in Chapter 5), the orthographic word (as in Chapter 6), and the utterance (as in Chapter 7). Systems *-{B,C,D}-1 are built, incorporating features derived from these VSMs. In each set of systems, features from a VSM built on one tier of analysis are added, meaning that the last group mentioned incorporates features derived from all 3 VSMs (letter, word and utterance). The feature sets denoted by the letters B, C and D in these system identifiers are summarised in Table 8.2.

Based on results presented in Chapters 5, 6 and 7, the **Hypothesis 2** is broken down into the following subhypotheses:

Hypothesis 2a The VSM of letter types will improve system performance in all languages without feature selection needing to be performed (see Section 5.3.1).

Hypothesis 2b The VSM of word types will improve system performance, but will require feature selection to be performed (see Sections 6.3 and 7.2).

Hypothesis 2c The VSM of utterances will improve system performance, but will require feature selection to be performed.

Hypothesis 2d When feature selection is applied in a system, adding features from any source will not degrade performance beyond that of the baseline (see Section 7.2).

A subset of hypotheses will be selected and refined in light of the objective evaluation of Section 8.6 before they are tested subjectively in Section 8.7. As hypotheses already given state predictions related to voices where feature selection is performed, two voices were built for each naive system: one where no feature selection is performed (voices *-{A,B,C,D}-1) and ones using the same data and feature set, but with the use of an ensemble of trees during acoustic model training to select a set of linguistic features to be used (voices *-{A,B,C,D}-2).

Vector space models and ensembles of trees for feature selection are presented in this thesis as a means of mitigating the effects of a lack of language-specific expertise and annotation when training TTS systems. However, there is no inherent restriction that limits their use to cases where resources are scarce: it might be that in some cases, the features obtained by these techniques complement those that arise from the supervised learning more common in the training of TTS systems. In Section 6.3.3 above, for example, it was found that combining features derived from the Brill tagger with ones derived from a VSM of word types to predict phrase-breaks gives an average F measure that is very slightly higher than those of systems built with either set of features exclusively. In other work as well, such as that of Turian et al. (2010), word representations learned in an unsupervised manner are proposed as a complement to traditional features such as part of speech, rather than as a replacement for them. Also, given that features used in conventional system building are typically numerous and noisy, and given the known lack of robustness of tree-based methods against such features, it is also thought that the method of feature selection developed in Chapter 7 will be beneficial to TTS systems otherwise built in the conventional way.

Systems were built that allow the evaluation of the latter idea. For systems {EN,FN,RM}-T-2, the procedure followed to build topline systems {EN,FN,RM}-T-1 is taken as a basis, but tree-ensemble feature selection is performed before the final stage of model estimation. Feature selection is chosen to test in conjunction

with standard feature-sets rather than vector space models. This is motivated by the fact that the space of reasonable configurations of systems combining topline and VSM features is a vast one. Feature selection can be combined with standard systems much more straightforwardly, and poses fewer design choices. The third hypothesis of these experiments, which relates to these systems, is that:

> **Hypothesis 3** Each member of {EN,FN,RM}-T-2 will outperform the respective system in {EN,FN,RM}-T-1.

The following section gives details of the 5 feature-sets used in these experiments (A, B, C, D, and T), and of the voices built using them.

8.4 System Construction and Features Used

Acoustic models were built in all languages and with all feature-sets using a common training procedure. This is the recipe called HTS-2010 in Section 2.1.3. In every case, 3 iterations of context clustering were performed. For systems *-*-2, an ensemble of trees was built prior to the final pass through this loop, so that apart from the training of the ensemble, all systems were built with a comparable amount of training.

The only aspect of the voices that was varied besides data-set (and concomitantly, language), was the configuration of textual-linguistic features used. These features are varied in one of two ways: either at the outset, by means of adding more features from different VSMs, or else during building of acoustic models, by applying feature selection.

The construction of the naive systems *-A,B,C,D-* is summarised diagrammatically in Figure 8.2 on page 150. Reference is made to this diagram in the course of the following discussion.

8.4.1 Feature-set A

Voices {EN,FN,RM}-A,B,C,D,T-{1,2} were built with the components shown in ellipses in Figure 8.2. These components and the procedures by which they are related will now be briefly described.

Primary Corpus: Speech Waveforms and Text Transcription

Speech corpora of similar sizes were used for each of the three languages. These corpora consisted of recordings segmented into utterance-sized files and a plain

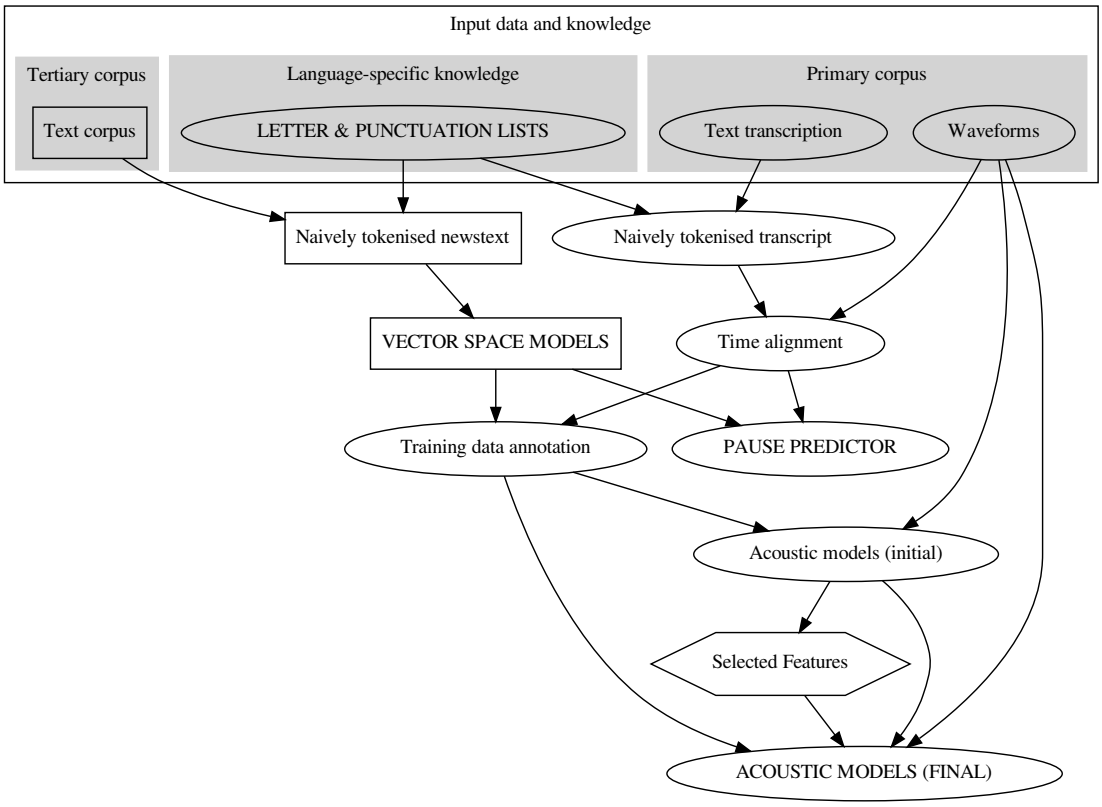


Figure 8.2: *Construction of naive systems ({EN,FN,RM}-{A,B,C,D}-{1,2}). Rectangular boxes show elements only present when using feature-sets B, C & D. The hexagonal box shows an element only present in systems *-*-2. Uppercase shows the components that make up a trained voice.*

orthography transcription of that speech, aligned at the utterance level. Where the speech data were made available at sampling rates higher than 16kHz, a downsampled 16kHz version was created for these experiments.

Although the transcription used to build annotation for the *naive* systems uses standard orthography and does not incorporate any specalised technical annotation such as phonetic transcription, stress marks, etc., some care was taken to ensure that numerals and abbreviations were expanded in the standard way in the transcript.

For the English voices, a subset of RJS-1000 (see corpus description on page 53) of similar size to the corpora of the other languages was used: 650 utterances, giving 58 minutes of speech (discounting initial and final silences), and 41,614 phoneme tokens.

For the Finnish voices, a database consisting of 600 phonetically balanced Finnish utterances spoken by a 39-year-old Finnish male was used. The database contains 50 minutes of speech data (discounting initial and final silences) and

Language	No. utterances	Minutes	No. of phoneme tokens
English	650	58	41,614
Finnish	600	50	41,823
Romanian	1000	46	40,215

Table 8.3: *Details of speech corpora used for experiments in Chapter 8. Durations given do not include initial and final silences. Counts of phoneme tokens do not include any silence segments.*

41,823 phoneme tokens.[1]

The Romanian voices were built from a subset of the publicly released Romanian Speech Synthesis corpus (Stan et al., 2011). The subset used consists of 1000 sentences: all of the news sentences selected for diphone coverage, and a few extra randomly chosen news sentences. This subset contains 46 minutes of speech (discounting initial and final silences), and 40,215 phoneme tokens.

Orthographic Information: Letter and Punctuation Lists

The only language-specific expertise used to build systems *-{A,B,C,D}-* is contained in a set of lists specifying which characters are punctuation and which are letters, and which represent orthographic word boundaries. These lists were obtained in an interactive manner: starting with empty lists, a script scans the transcription part of the primary data and the input lists, and outputs a list of characters present in the data but absent from the input lists. The user then adds the found characters to the list they think most appropriate and reruns the script until all characters are accounted for. The user also specifies implementationally-safe substitutions together with the symbols in the lists. This is primarily to ensure a set of names for letters that all parts of the system will be able to handle as the names of acoustic models, etc., but is also where knowledge of e.g. upper- and lower-case variants is incorporated (mapping e.g. both *a* and *A* to a single symbol).

This procedure assumes some familiarity with the script of the the target language, but not necessarily any with the language itself. It is clear, for example, to a person familiar with the variant of the Roman alphabet used to write English that the Romanian characters *ş* and *î* are alphabetic characters rather than punctuation marks. Likewise, *ş* and *Ş* are clearly upper- and lower-case variants

[1]Thanks to Martti Vainio and Antti Suni of the University of Helsinki for making the Finnish database available for the experiments

of one another. Even less obvious variants (such as *ş* and *ș*) become obvious as such to an observer of a Romanian corpus without any knowledge of the language due to their graphical similarity and similar distribution in frequent words (e.g. *şcoala*, *şi*).

The user who produced the lists for the present experiments is a native speaker of English, and has no knowledge of Finnish or Romanian. Note that the information about character type that must be generated is exactly the sort that could be obtained fully automatically from the General Categories of Unicode characters (Korpela, 2006, pp. 210–212).

Naively tokenised transcript

A naive tokenisation was performed on the transcription part of the primary data. This tokenisation is performed by using the lists described in the previous paragraph to compile a regular expression. This regular expression performs a simple tokenisation – words are considered to be delimited by sequences consisting of one or more punctuation symbol together with one or more word boundary symbol (i.e. whitespace). The tokenised text consists of a sequence of segments, consisting of words, punctuation symbols, and word delimiters (whitespace).

The regular expression is very naive; for example, English word-final *'* is not disambiguated as final apostrophe or quotation mark. Word-internal non-letters (as in *Jean-Paul*) are treated as speech-emitting letters. Despite this naivety, this technique gives reasonable results, providing that the assumptions made about the transcription already mentioned are met (i.e. that the transcription is text-normalised and therefore free of non-standard words).

Time Alignment

The transcription part of the primary data, naively tokenised as outlined in the previous paragraph, is used to initialise a set of labels for the speech recordings. These labels are then time-aligned with the speech using an alignment procedure based very closely on that of the tools described in Clark et al. (2007). Briefly, the parameters of a set of hidden Markov models representing context-independent units are estimated.

The inventory of modelling units is made up of all of the letters in the letter list that are present in the training corpus, and two models for *junctures*: a model of silence called *sil*, and a non-emitting model called *skip*. Punctuation symbols resulting from naive tokenisation are mapped a single symbol, *PUNC*, and word

delimiters (i.e. whitespace) are represented by the symbol *SPACE*. Both these are junctures, and correspond to either of the juncture acoustic models, *sil* and *skip*. At the start of training, *sil* is initialised by fixing *PUNC* to always correspond to the model for silence; *SPACE* is initially fixed to always have the pronunciation *skip* (i.e. not to emit any observations). After initial estimation, a new sequence of models is determined by Viterbi search, and junctures are allowed to map to either *sil* or *skip*. This model sequence in turn provides the basis for further Baum-Welch estimation of model parameters.

Pause Predictor: Classification Tree

As mentioned in Chapter 6, the positions of pauses extracted from audio are expected to be a reasonable surrogate for expert-specified phrase-break annotation. Therefore, silence segments in the letter alignment longer than 60 ms are treated as pauses. The 60 ms threshold was chosen on the basis of brief visual and auditory inspection of a few utterances. Symbols representing the presence and absence of such a pause after each word are used as the levels of a response variable for training a classification tree (see Chapter 6). Systems using feature-set A use only features relating to punctuation and positional information as predictor variables (i.e. the baseline feature set used in Experiment 2 of Chapter 6: see Section 6.3.2).

Training Data Annotation and Acoustic Model Training

The time-aligned letter annotation is supplemented with the following features:

1. The identity of the current letter and of the letters 1 and 2 places to its left and right
2. The number of letters {since, until} an orthographic word boundary
3. The number of {words, letters} {since, until} an utterance boundary
4. The length of a word in letters
5. The length of an utterance in words
6. The number of letters {since, until} a pause
7. The number of words since the last pause until the next one.

The final two types of feature refer in the training data annotation to pauses detected in the audio. The equivalent features to be used at synthesis time are based instead on the predictions of the classification tree described in the previous paragraph. This time-aligned annotation is used for training acoustic models for synthesis. As mentioned above, this training follows the procedure described in Section 2.1.3, where it is denoted HTS-2010.

8.4.2 Feature-sets B, C & D

Feature-sets B, C & D use the components described for the baseline feature-set in Section 8.4.1, and incrementally add features derived from 3 vector space models: models of letter types, word types, and utterances. The additional components that must be constructed to incorporate these features are shown in rectangular boxes in Figure 8.2 on page 150 and detailed below.

Tertiary Corpus: Text Data

Text data were collected in the three languages for the purpose of building vector space models on three orthographic levels (letter, word, sentence). Similar amounts were collected in each language, as follows.

For systems EN-{A,B,C,D}*, the same 1.2 million tokens of Wall Street Journal text used in Chapter 6 was used for building vector space models. However, that data is distributed in tokenised form. To mimic the situation where time or expertise is not available to produce an intricate, language-specific tokenisation, untokenised text was obtained by automatically reversing the Penn Treebank tokenisation of the distributed data; the reversal of the tokenisation is slightly imperfect but results in text that is very similar to text that can be obtained by web-scraping. The word count of the resulting untokenised text is c. 1 million words.

A corpus of web-scraped news texts made publicly available in the Leipzig Corpora Collection provided c. 1.2 million words of untokenised Finnish text data (Quasthoff et al., 2006).[2]

Web-scraped news texts provided c. 1 million words of untokenised Romanian text data. Care was taken to ensure that the sentences to be used to test the voices do not appear in the newstext used.

A division of the corpora into utterances is necessary for building the utterance model (but not necessary for either the word or letter models). The division

[2]Specifically, the first 1.2 million words of the corpus named *fin_web_2002_3M-text* were used.

Language	Letter tokens	Word tokens	No. utterances	Letter types	Word types
English	6,078,644	1,150,871	49,209	60	48,613
Finnish	11,293,565	1,491,571	109,006	78	262,818
Romanian	5,693,094	1,085,339	46,878	70	66,267

Table 8.4: *Details of newstext corpora used as tertiary data for experiments in Chapter 8. All counts are given in terms of the naive tokenisation used.*

provided in the English and Finnish corpora was used. For the Romanian data, boundaries indicated by HTML codes were used. Otherwise, utterances were considered to end at the symbols *.* *!* and *?*. This is very crude, and often the de facto 'utterances' created end on a full stop that actually was intended to indicate an abbreviation.

Naively tokenised newstext

Exactly the same naive tokenisation by which the tokenised speech transcription was derived from the initial transcription was followed to produce a corpus of naively tokenised newstext. Although the procedure used was identical, the different nature of the primary and tertiary text data creates a slight mismatch. That is, the primary text data (speech transcriptions) are assumed to contain no non-standard words. No such assumption is made for the tertiary data (newstext). It is expected that the vector space models will be robust enough to cope with such mismatches. Table 8.4 gives some details of the text corpora used for these experiments.

Vector Space Models

Vector space models characterising letters, words and utterances are estimated from the naively tokenised text data, as follows.

Systems B–D: Letter type model Like the vector space model built for letters in Chapter 5, the model used here for letters is a static model of unit types using context features external to those units (left and right neighbouring units, including spaces and punctuation). The same parameters as in Chapter 5 are used: n is set equal to the number of letter types (see Table 8.4), and 5 dimensions of the space resulting from SVD are kept. The vector space model is discretised uniformly along each dimension into 50 bins, as in Section 6.4.1.

Systems C & D: Word type model Like the vector space model built for words in Section 6.2, the model used here is a static model of unit types using context features external to those units (left and right neighbour words, including punctuation). The same parameters as in Section 6.2 are used: n is set equal to 250, and 50 dimensions of the space resulting from SVD are kept. The vector space model is discretised uniformly along each dimension into 50 bins, as in Section 6.4.1.

It is because of the choices involved in building these word features (using orthographic word sized units rather than subword morpheme-type units) that these features are not predicted to be useful for Finnish (see above, Sections 2.3 and 8.3).

System D: Utterance model The utterance space models used for this experiment are like the one described in Section 4.2.4 and similar to that implemented in Section 7.3. That is, they are dynamic models of utterance tokens using context features internal to utterances. All utterances of the text corpora described in Section 8.4.2 are used, and the tokens resulting from their naive tokenisation form internal contexts. All word tokens occurring more than once in a corpus are retained in the model. No stop words are excluded. Term-frequency inverse document frequency weighting is applied to the raw co-occurrence matrix, and singular value decomposition is performed. For the utterance spaces used in this chapter, 50 dimensions of the latent space are retained. The vector space is discretised uniformly along each dimension into 50 bins, as in Section 6.4.1.

It should be noted that due to the design of the corpora used in these experiments, the usefulness of utterance representations is immediately questionable. The speech corpora are designed in a conventional way to exclude the sort of variability between utterances that an utterance space is expected to be useful for modelling. For example, questions are excluded from the prompts from which the corpora are recorded. The prompts are selected from a large body of text and so recorded without any discourse context. It is in just such a discourse context that utterance function (concession, expansion, contradiction, etc.) would be expected to significantly affect the acoustic realisation of an utterance. The utterance representations are here included out of curiosity rather than an expectation that they will lead to an improvement in the quality of synthetic speech. If feature selection with ensembles of trees (see Section 8.4.4) is as effective in these experiments as in the one presented in Section 7.2, then the inclusion of these probably irrelevant features should at least not harm performance (see Hypothesis

2d above).

Pause Predictor: Classification Tree
For systems using feature-set B, the classification used to predict pauses at run time is trained in a manner identical to that of the tree used with feature set A. For systems using feature-sets C and D, extra predictor features derived from the word type vector space model are used. 50 features from the VSM features for the word preceding a juncture are used, and for the word following. The discretisation used is identical to that described in the previous paragraph.

8.4.3 Feature-set T: Topline Features

Benchmark systems were built to provide a point of reference when evaluating the naive systems. These three systems – EN-T-1, FN-T-1 and RM-T-1 – were built using database annotation derived from conventional front-ends that rely on specialised knowledge and data annotation, as detailed below. They are here considered as topline systems.

English: Systems EN-T-{1,2}
The English annotation was provided by the English front-end distributed with the Festival speech synthesis system discussed in Section 2.1.3. The time alignment is provided by the forced alignment of acoustic synthesis models – previously built on the whole of the RJS corpus – with their training data.

Finnish: Systems FN-T-{1,2}
The annotation used for the Finnish database is obtained as described in Vainio et al. (2005a,b), and consists of features relating to phonemes, phonetic categories, syllables, stress, phrasing, accent, and prominence.

Romanian: Systems RM-T-{1,2}
The annotation used for the Romanian database is that distributed with the corpus, and detailed in Stan et al. (2011). Briefly, it consists of a pronouncing dictionary, a classification tree for handling letter-to-sound conversion built from it, a dictionary for lexical stress, rules for syllabification, and a part of speech tagger.

8.4.4 Feature Selection: Systems *-*-2

For Systems *-*-2, feature selection was performed prior to the final iteration of clustering and model reestimation; this is represented by a hexagonal box in Figure 8.2. For each system, an ensemble of trees was built as described in

Section 7.2, including an artificial contrast feature for each linguistic feature used by the voice. Using bootstrap resampling of state occupancy scores for each tree in the forest, a sub-forest of 100 trees was built for each state of the models of fundamental frequency and spectral envelope, resulting in a forest of 1000 trees for each voice. The same method as in Section 7.2 was used to select features: the whole of this forest was used to select a single set of features. The Wilcoxon signed rank test was used to find features that create a total improvement in log likelihood during forest building that is significantly higher than that of the corresponding contrast feature ($\alpha = 0.05$). The single list of selected features are used for the final iteration of clustering of all parts of the voice.

8.5 Synthesis

All evaluation of the 30 voices built for this experiment was conducted using two sets of utterances. The first set (the *natural* set) consists of speech held out from the corpora during training, and annotation produced from the corresponding plain orthography transcription of those utterances. These sets were used for objective evaluation of synthesised speech, results of which are presented in Section 8.6, and for the subjective evaluation of naturalness, results of which are presented in Section 8.7. The second set of test utterances for each language was a set of Semantically Unpredictable Sentences (SUS: Benoit et al., 1996). These were for use in the subjective evaluation of intelligibility (see Section 8.7).

Synthetic speech was generated from the text prompts of the utterances (in the case of topline systems, existing annotation was used, although this had in turn been generated from the text of utterances), and evaluated as will now be described.

8.6 Objective Evaluation

8.6.1 Method

Objective evaluation was conducted using the parametric representation of synthetic speech output for the natural sets of sentences described in Section 8.5. Part of this representation – frames of Bark cepstral coefficients – was aligned with a comparable parametric representation extracted from the held-out data using Dynamic Time Warping (DTW). The warping path discovered in this way

was also used to align the generated and natural F_0 sequences. Using these time-warped parameters, Bark cepstral distortion and Root Mean Square Error of fundamental frequency were computed.

DTW was preferred to synthesis with natural durations in this case (cf. experiments in Chapters 5 and 6) as the variety of modelling units used (phonemes in the topline systems, letters in the naive systems) means that there is no single gold standard alignment that can be used for all systems to be compared. Only frames of the speech that are voiced in the reference utterances were used for this evaluation. This is a way to exclude silent segments without recourse to gold standard annotation. Silence segments excessively improve the cepstral distortion measure when they are of the correct length (spectral shape of silence is mainly unvarying and easy to synthesise), and conversely cause excessive degredation when they are wrongly inserted (in the case that DTW aligns many silent frames with speech frames).

8.6.2 Results

Without Feature Selection

The results of the objective evaluation for Systems *-*-1 are shown in Figure 8.3. A similar pattern can be seen across the three languages with regard to the scores for Bark cepstral distortion (Figures 8.3a, c and e). That is, adding features derived from the letter spaces to the basic features reduces distortion in synthesis so that part of the gap between baseline and topline is closed in all cases. However, distortion increases as word space and then utterance space features are added. For English (System EN-D-1), the result is still an improvement upon the baseline, but Finnish systems FN-D-1 and FN-A-1 score similarly, and for Romanian, RM-D-1 achieves a slightly worse score than baseline RM-A-1.

There is no such pattern shared across languages with regard to the scores for Root Mean Squared Error of F_0 (RMSE-F_0: Figures 8.3b, d and f). For English, the trend for RMSE-F_0 is similar to that for cepstral distortion for all languages: adding letter space features reduces the error, but subsequently adding word and utterance space features detracts from this initial improvement. For Finnish and Romanian, on the other hand, all VSM features worsen the RMSE-F_0, widening the performance gap between baseline and topline. For Finnish, the extent of this performance decrease lessens slightly as more features are added; for Romanian, it increases with additional features.

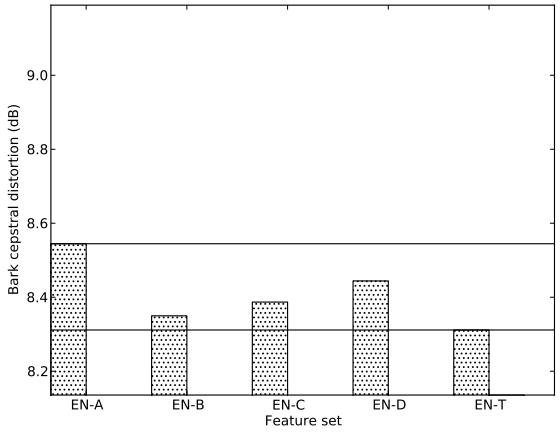
With Feature Selection

In Figure 8.4, the results shown in Figure 8.3 are supplemented with results for Systems *-*-2, that is, systems where feature selection is used. A similar pattern was seen across languages with regard to Bark cepstral distortion without the use of feature selection. When feature selection is used (Figures 8.4a, c and e), the pattern is once again mostly similar across the three languages. Feature selection harms performance considerably when baseline features are used, and slightly when the letter space model is used. When the word space model is used, feature selection has little effect on score; when the utterance space features are added, selection gives a slight improvement in all three languages. The effect of feature selection on topline systems will be discussed separately in the next section.

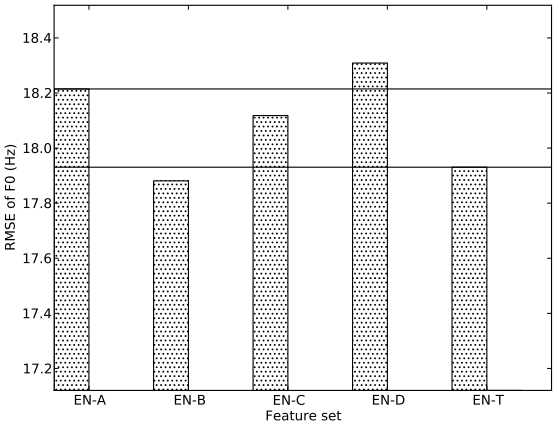
The effect of applying feature selection on RMSE-F_0 (Figures 8.4b, d and f) is erratic and varies across languages. For the Romanian system, it gives a consistent improvement, when systems with feature selection are compared to corresponding systems without it. For English and Finnish systems, the effect of feature selection is much more varied between conditions. Two points of similarity are that feature selection gives an improvement when word space features are used, and a worsening when utterance space features are added. The first point is consistent with the similar improvement seen in Section 7.2. The second shows that the prediction made in Section 8.4.2 is not borne out by the results of the experiment. The prediction was that the features of the utterance space, while probably irrelevant in the context of the current experiments, should at least not harm performance in the case that feature selection is applied. The results cast doubt on the ability of ensembles of trees – at least as they are used here – to handle the shotgun approach to feature creation described in Section 2.3. If it were working as planned, systems employing feature selection would always perform at least as well as the best-performing system to their left in the plots of Figure 8.4 (cf. Hypothesis 2d above).

Topline Systems with Additional Feature Selection

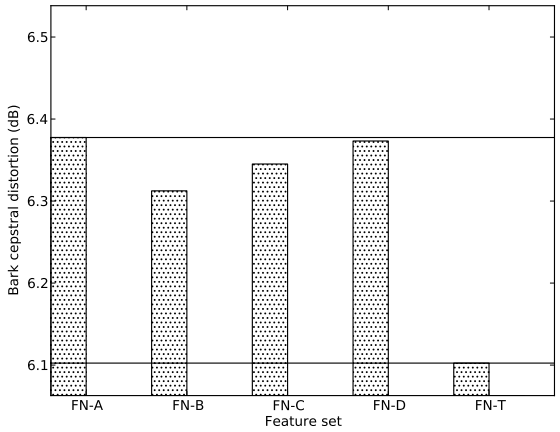
Figure 8.4 includes results for Systems *-T-2, that is, systems using conventional features where feature selection is applied. For English and Romanian, applying feature selection to the topline systems leads to an improvement in performance, both for generated cepstrum and F_0. For Finnish, the same procedure has a negative impact on the performance of the topline system, worsening performance in terms of both measures used.



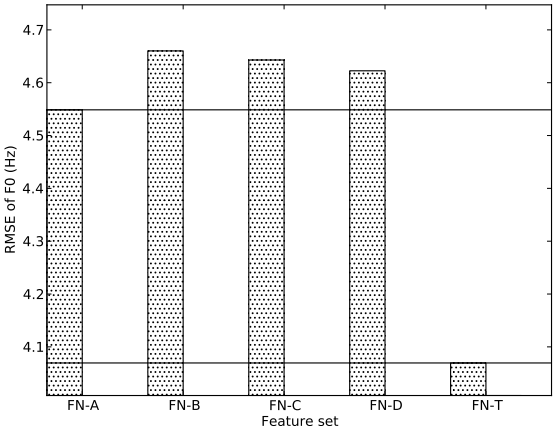
(a) *Bark cepstral distortion (English)*



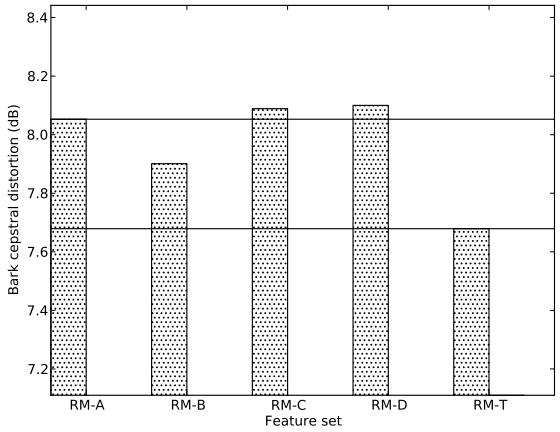
(b) *Root Mean Square Error of F_0 (English)*



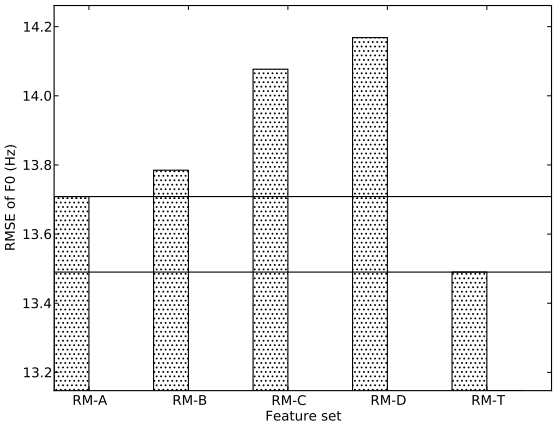
(c) *Bark cepstral distortion (Finnish)*



(d) *Root Mean Square Error of F_0 (Finnish)*

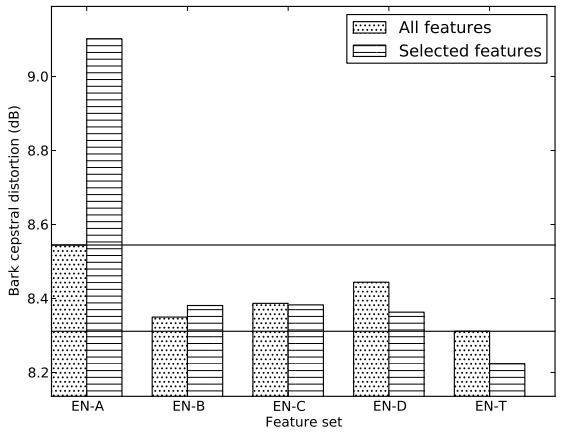


(e) *Bark cepstral distortion (Romanian)*

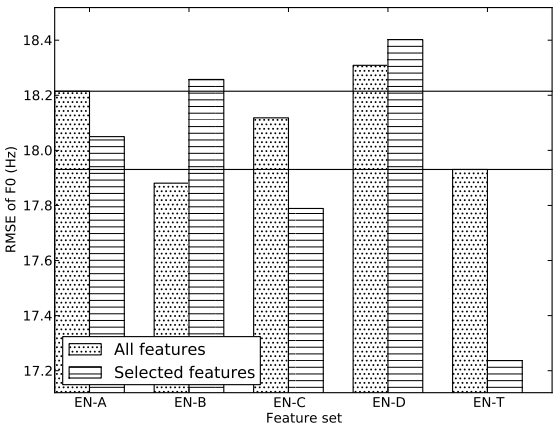


(f) *Root Mean Square Error of F_0 (Romanian)*

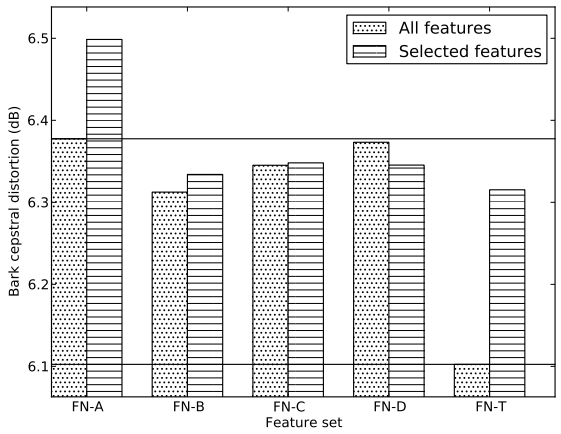
Figure 8.3: *Results of objective evaluation the 15 voices build without feature selection presented in Chapter 8. Horizontal lines indicate performance of baseline and topline systems.*



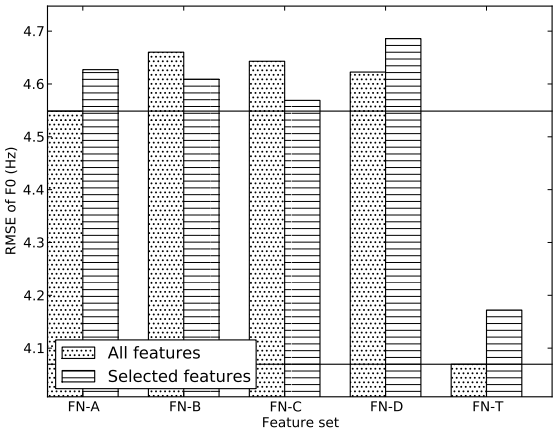
(a) *Bark cepstral distortion (English)*



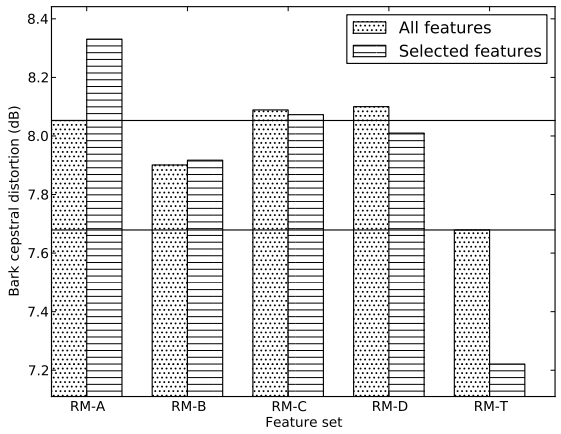
(b) *Root Mean Square Error of F_0 (English)*



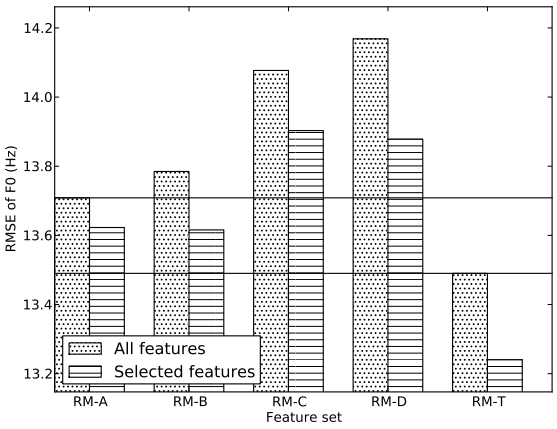
(c) *Bark cepstral distortion (Finnish)*



(d) *Root Mean Square Error of F_0 (Finnish)*



(e) *Bark cepstral distortion (Romanian)*



(f) *Root Mean Square Error of F_0 (Romanian)*

Figure 8.4: *As Figure 8.3, with additional results for the *-*-2 voices (employing feature selection)*

8.7 Subjective Evaluation

8.7.1 Selected Hypotheses

Hypothesis 1 given in Section 8.3 will be tested in this experiment. Based on the above observations of the results of objective evaluation, a single member of the subhypotheses of Hypothesis 2 formulated in Section 8.3 is selected (**Hypothesis 2a**) for testing in a subjective evaluation: that the VSM of letter types will improve system performance without feature selection needing to be performed.

In an ideal scenario, systems *-D-2 would be compared with baselines *-A-1 and toplines *-T-1. However, the improvement between systems *-A-1 and *-D-2 hypothesised above is not tested here because results of the objective evaluation suggest that feature selection – as used in these experiments – is not performing the role planned for it. Refinement of this means of feature selection is left for future work (see Chapter 9).

Results of the objective evaluation tend to support Hypothesis 3: that the method of feature selection developed in this thesis is beneficial for topline systems. However, it is not a hypothesis central to this thesis, and its subjective testing is left for separate future work.

To test the selected hypotheses, 3 systems were selected for each language to use in subjective evaluation with human listeners. Besides the baseline system *-A-1 and topline *-T-1, the experimental systems *-B-1 were used. The refined hypotheses that the listening tests conducted are designed to test are that:

> **Hypothesis 1 (part 1):** The topline system in a given language (one of systems *-T-1) will be significantly more **natural** than the corresponding baseline (one of systems *-A-1)
>
> **Hypothesis 1 (part 2):** The topline system in a given language (one of systems *-T-1) will be significantly more **intelligible** than the corresponding baseline (one of systems *-A-1)
>
> **Hypothesis 2a (part 1):** Adding features derived from a letter space model to a system (giving one of systems *-B-1) will create a significant increase in **naturalness** compared with the corresponding baseline (one of systems *-A-1)
>
> **Hypothesis 2a (part 2):** Adding features derived from a letter space model to a system (giving one of systems *-B-1) will create a significant

increase in **intelligibility** compared with the corresponding baseline (one of systems *-A-1)

In this evaluation, the difference in naturalness of two systems will be measured by listeners' stated preference for utterances generated by the systems. Intelligibility of a system will be measured by counting errors in listeners' transcriptions of utterances generated by that system.

8.7.2 Procedure

The listening test was conducted via a web browser. Conditions were different for each of the languages. The English part of the evaluation was conducted in purpose-built acoustically insulated booths using headphones. The Romanian and Finnish parts of the evaluation took place remotely, so that conditions were less controlled. However, participants were asked to use headphones and a quiet room.[3] 39 paid listeners participated in the English evaluation, and 19 and 11[4] unpaid listeners in the Romanian and Finnish evaluations, respectively. All listeners were native speakers of the relevant language, between the ages of 18 and 25 in the case of English and Romanian evaluations, and between 20-45 years in the case of the Finnish evaluation.

Part 1 consisted of three sections, in which the three system pairs (*-A-1, *-B-1), (*-A-1, *-T-1), (*-B-1, *-T-1) were compared, respectively. In each section, AB tests between the relevant systems were conducted, where listeners were asked to state a preference for one of two synthetic speech stimuli. Altogether 20 such pairs were presented in each section. Each listener heard each utterance text spoken by only one pair of systems over the course of the whole test. Furthermore, listeners were assigned to 3 equally- or similarly-sized groups,[5] and the assignment of utterance texts to sections (system pairs) was balanced across listener groups, with the effect that the different texts were heard uniformly generated by the 3 system pairs.

The ordering of systems pairs in the sections was chosen to avoid ceiling effects: the first of these comparisons (*-A-1 vs. *-B-1) is expected to involve differences

[3]Many thanks to Antti Suni and Heini Kallio of the University of Helsinki, Tuomo Raitio of Aalto University, and to Adriana Stan of the University of Cluj-Napoca for recruiting Finnish and Romanian participants.

[4]In fact, 10 Finnish listeners participated, one of whom unexpectedly participated twice. However, the records of listeners kept do not allow the duplicate responses to be identified.

[5]The 3 groups were of perfectly equal size for the English evaluation, but less perfectly balanced for Romanian and Finnish: 9–5–5 and 5–3–3, respectively.

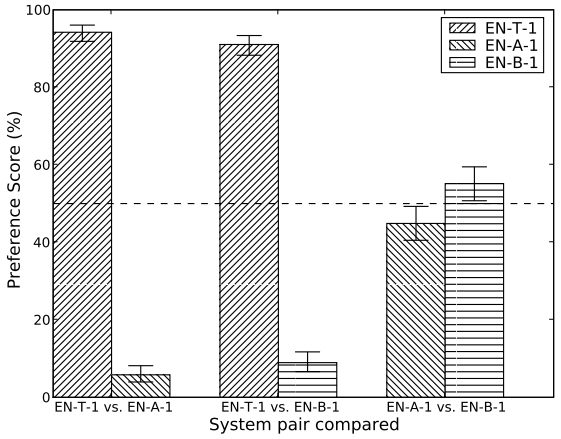
that are less obvious to listeners than later comparisons involving the topline systems. It is put first so that listeners are not led to expect as big differences to base their preference on as those that appear later in the test.

Part 2 consisted of three sections, each containing 18 tasks. Here, listeners were asked to type in a transcription of each test stimulus. Stimuli for all languages were Semantically Unpredictable Sentences. As in Part 1 of the evaluation, each listener heard each SUS text spoken by only one system over the course of the whole test. Unlike in the preference task, no ceiling effect was anticipated for the transcription task. Therefore, SUS generated by the three systems (*-A-1, *-B-1, *-T-1) were distributed evenly within each of the three sections. Transcription accuracy was measured using word error rate (WER) for English and Romanian systems, and letter error rate (LER) for the Finnish ones.

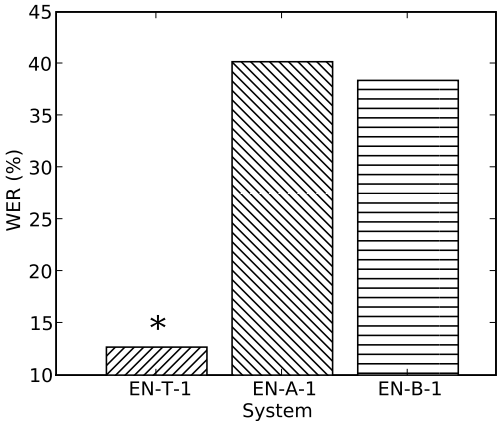
8.7.3 Results

Figures 8.5a, c and e show the results of Part 1 of the evaluations, which considers naturalness. Mean preference scores of each of the three comparisons made in each language are shown. Error bars show confidence intervals determined by a binomial test (α=0.05 with Bonferroni correction). The expected outcome under the null hypothesis (no preference for either system of a pair) is 50%, shown by a dashed line in the plots. It can be seen therefore that for the English and Finnish systems, there is a significant preference for the topline systems over the baselines, allowing acceptance of the **Hypothesis 1 (part 1)** as stated in Section 8.7.1 for Finnish and English. Also for these languages, the systems incorporating features derived from a letter space are found significantly more natural than the baselines, allowing acceptance of **Hypothesis 2a (part 1)** for those languages. It is also found for English and Finnish that listeners find the speech produced by topline systems significantly more natural than that produced by the ones incorporating the letter space features. That is, the letter space closes a significant part of the gap between baseline and topline, but the topline is still significantly better. For the Romanian systems, no system preferences are found significant, even between the topline and baseline systems. The systems using letter space features is slightly preferred over the baseline, but not significantly so. Neither **Hypothesis 1 (part 1)** nor **Hypothesis 2a (part 1)** can therefore be accepted for the Romanian systems.

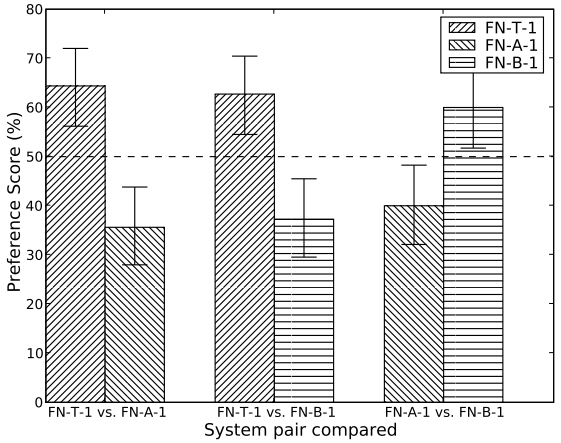
Figures 8.5b, d and f show the results of Part 2 of the evaluations, which considers intelligibility. Mean transcription error scores for each of the three sys-



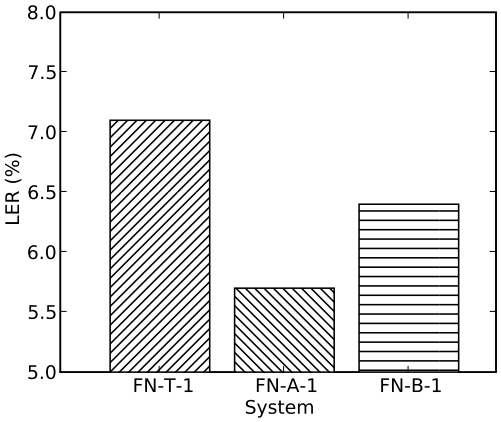
(a) *Preference scores (English)*



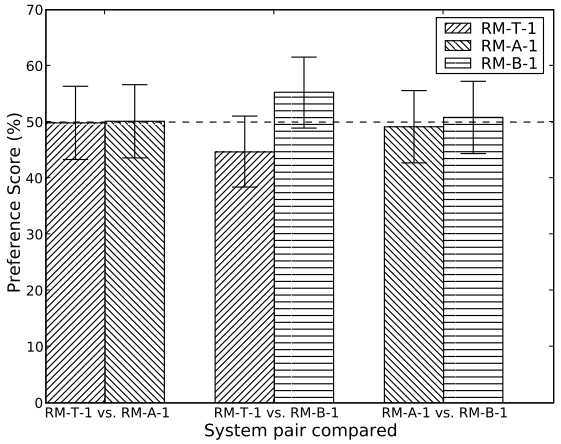
(b) *Word error rates (English)*



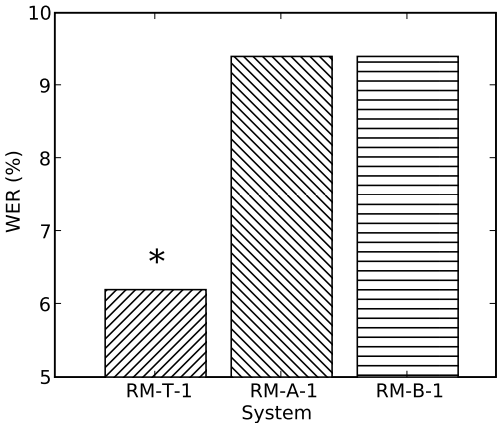
(c) *Preference scores (Finnish)*



(d) *Letter error rates (Finnish)*



(e) *Preference scores (Romanian)*



(f) *Word error rates (Romanian)*

Figure 8.5: *Preference and intelligibility scores.*

tems in each language are shown: word error rates for English and Romanian, and letter error rates for Finnish. Differences between system error rates were compared in a pairwise fashion using the bootstrap procedure of Bisani and Ney (2004): bootstrap-t confidence intervals were calculated over system differences. The systems marked with an asterisk in these plots were found significantly more intelligible than *both* competing systems (with $\alpha = 0.05$ and Bonferroni correction). Otherwise, no significant differences were found.

It can be seen that the topline systems in English and Romanian are significantly more intelligible than the corresponding baselines, allowing acceptance of **Hypothesis 1 (part 2)** as stated in Section 8.7.1 for these two languages. However, there is no significant difference between the intelligibility of the baseline and letter space systems for any language, meaning that **Hypothesis 2a (part 2)** cannot be accepted for any language. For Finnish, the trend is the opposite of what might be expected: the topline systems is less intelligible than the baseline. However, this difference is not found to be significant, and **Hypothesis 1 (part 2)** cannot be accepted for Finnish.

8.8 Conclusions

This chapter has presented the construction of 30 TTS systems in three languages. 24 experimental systems were built, to allow the evaluation of three automatically-induced feature sets, with and without the application of feature selection based on ensembles of trees. Three topline systems provided a benchmark in each language, and the same feature selection procedure was applied additionally to them.

The objective evaluation, conducted over all systems, suggested that the feature selection procedure does not perform as planned and as the results of the experiment of Section 7.2 suggest. Under the ideal operation of this procedure, irrelevant features would be discarded, meaning that adding arbitrary new noisy features would not harm performance. This is not what the results of objective evaluation suggest is actually happening.

Three selected systems from each language were compared in evaluations with human listeners. The automatically-induced features tested in these evaluations – derived from vector space models of letter types – never harm scores for naturalness or intelligibility. In two of the languages used in the experiment, these features produced a significant preference among native listeners for the system

in which they are used, compared with systems from which they are absent.

Chapter 9

Conclusions and Future Work

9.1 Contributions

The primary contribution of the work presented in this thesis is a methodology for the construction of TTS front-ends using representations of subword-, word- and utterance-level units which are learned in an unsupervised fashion. The approach is novel: this is the first published presentation where unsupervised learning is applied to the front-end of a TTS system in a unified way across several levels of analysis.

It has been shown that the distributional–acoustic method developed enables the unsupervised acquisition and use of representations for textual and linguistic objects that can replace conventional representations of the same objects derived from expert knowledge in TTS systems. Unsupervised induction of representations incurs only a fraction of the cost associated with manually annotating data or encoding knowledge in look-up tables, and can close much of the gap in performance between systems using conventional representations and ones which use no representations at all, as the following selected results show:

- In Section 5.3.1 a baseline TTS system using phonemes but incorporating no knowledge of phonetic categories was built; mean Bark cepstral distortion per non-silent frame between held-out speech and speech synthesised by the system is 5.54dB. Incorporating expert-derived phonetic categories brings this distortion down to 5.41dB. Incorporating features obtained without expert knowledge, using distributional analysis, closes most of this gap in performance, giving a score of 5.44dB. The full set of phonetic categories can be approximated with a reduced set, such as a simple division of phonemes into vowels and consonants, which could be trivially speci-

fied by a non-expert system-builder. These two categories close part of the same performance gap when added to the system, but not so much as the unsupervised representations (score: 5.48dB).

- In Section 6.3.3, the performance of several phrase-break predictors was measured objectively using the mean F scores of phrase-breaks predicted by 10 systems sharing the same configuration. A baseline system – using only basic features such as distance until punctuation – attained a mean F score of 69.6%, and a conventional topline system which incorporates part of speech tags output by a state-of-the-art tagger attained a mean score of 78.8%. Once again, the incorporation of features obtained without expert knowledge, using distributional analysis, closes most of this gap in performance, giving a mean F score of 77.7%. The word representations used in this experiment are learned from the same 1.2 million words of data as the tagger used for the topline system, but pretend that no part of speech annotation exists for those data.

- The fact that no manual annotation is required to learn such representations means that it is trivial to increase the amount of data on which they are learned; in Section 7.1, similar representations are learned on 25 million words of text, and applied to the same phrase-break task, giving a mean F score of 79.1%, thus entirely closing the gap between mean performance of the systems. It is not doubted that training a part of speech tagger on a comparably larger corpus would improve topline performance. However, if it is assumed that tagged corpora do not already exist in the target language (the case for the majority of the world's languages), this probable gain in performance comes with the considerable expense of manually annotating an extra 24 million words of text.

- Sections 6.4 and 7.2 presented attempts to apply the learned word representations directly to HMM state-tying. In order to successfully exploit the representations for this task, a method of feature selection using ensembles of trees was devised. This is the first published application of tree-ensembles to feature selection for acoustic modelling in TTS. The F_0 trajectories generated by a baseline system making no use of word representations attain a correlation of 0.585 with the F_0 of natural held-out samples. Using part of speech as word representation increases this to 0.592. Using an automatically selected set of features derived from distributionally-acquired word

representations surpasses this topline performance, achieving a correlation of 0.613.

- Although no strong claims are made for the language-independence of the techniques developed, they are linguistically naive enough that they can be applied with little effort to any language making use of an alphabetic script with orthographically-marked word boundaries (Section 2.3). This was done in Chapter 8, where topline systems in English, Finnish and Romanian – including expert-derived speech annotation – were compared with systems in those languages built with minimal reliance on human intervention, and which depend on distributional analysis to characterise text objects. A subset of these systems were chosen for subjective evaluation, and for both Finnish and English, listeners significantly preferred systems incorporating distributional representations of letters to baseline letter-based systems (Section 8.7).

9.2 Future work

The framework developed for the experiments presented in this thesis enables the automated building of entire systems from data with minimal expert supervision. This framework forms the necessary foundation for ongoing work, which is carried out in the context of the *Simple4All* project which has the aim of developing:

> [...] speech synthesis technology that learns from data with little or no expert supervision.[1]

The work of this thesis provides a valuable starting-point for the work of this project. In turn, the project provides opportunities to build upon and extend the techniques developed in this thesis. Most obviously, the techniques developed in this thesis will be tested in languages other than the three used in the experiments of Chapter 8. Even within these languages, it is unclear to what extent the differences between results obtained for the languages are truly indicative of language differences, and to what extent the comparison between languages is confounded by other characteristics of the databases used (e.g., speaker-specific characteristics). Use of multiple speakers to build multiple voices within each target language would reveal these confounding effects.

[1] `http://simple4all.org/`

Incorporation of an unsupervised morphological segmentation module is expected to improve performance on languages with rich morphology. Due to the similarity between morphological and word segmentation tasks (Creutz and Lagus, 2007), it is also envisaged that a generalised segmentation module could also be of use for languages such as Mon, Thai, Tibetan, Burmese, Khmer and Lao,[2] which use alphabetic-syllabic scripts but which do not mark word boundaries orthographically.

It was noted in Section 2.3 that this thesis deliberately avoids the issue of text normalisation, considering it too language-specific to be handled in a wholly unsupervised way. Future work will focus on incorporating non-expert native-speaker knowledge in an active learning approach to this task.

The systems developed in this thesis embody a framework which allows the induction and incorporation of representations of text units of arbitrary size. The framework therefore allows features above the utterance level to be used, which is becoming increasingly important with recent trends in using 'found', continuously-recorded speech for system training instead of traditional TTS databases (Braunschweiler and Buchholz, 2011; Székely et al., 2011). Use of such data is planned for future work, and features at the utterance level and beyond, extracted and exploited in the distributional–acoustic framework, are expected to be an indispensable element in systems that make use of the natural variability in such data.

The method of feature selection used in this thesis is novel in the context of TTS, and provides improvements in the performance of conventional systems for two of the three target languages in Chapter 8 (in the objective evaluation). This is a serendipitous finding, as improvement to conventional systems is not the concern of this thesis. As such, this is not the place for extensive investigation of this finding. However, such investigation could form the basis of useful future work. On the other hand, the feature selection method did not provide the expected benefits in the experimental systems for which it was primarily intended (Section 8.8).

However, avoiding the use of tree-structured models might render the feature selection method developed unnecessary altogether. As mentioned in Section 4.2.5, the acoustic phase of the distributional–acoustic method advocated by this thesis was implemented with decision trees for pragmatic reasons, as doing so meant that state-of-the-art systems for acoustic model building could be used.

[2] These are the six languages that are counted in Table 2.2.

However, use of decision trees is an implementational choice: there is nothing in the general framework proposed which binds it to this choice of method for its acoustic phase. Indeed, the limitations of the divide-and-conquer approach used in decision trees is well known, as has been pointed out at various points in the course of this work (e.g. in Section 6.4.1). An attractive approach is to integrate continuous representations of textual units more closely with acoustic models, and perhaps to update them explicitly in light of the acoustics (an approach suggested by work such as Collobert and Weston, 2008; Collobert et al., 2011). This would involve fundamental revision of the acoustic-modelling framework used which is clearly beyond the scope of this thesis, but is planned for future work.

Bibliography

G. Anumanchipalli, K. Prahallad, and A. Black. Significance of early tagged contextual graphemes in grapheme based speech synthesis and recognition systems. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 4645–4648, 31 2008-April 4 2008.

G. K. Anumanchipalli and A. W. Black. Adaptation techniques for speech synthesis in under-resoured languages. In *Spoken Language Technologies for Under-resoured languages*, Penang, Malaysia, 2010.

A. Arvaniti and M. Baltazani. Greek ToBI: A system for the annotation of Greek speech corpora. In *Proceedings Second International Conference on Language Resources and Evaluation (LREC 2000)*, volume 2, pages 555–562, 2000.

J. Augerot. Romanian. In E. in Chief: Keith Brown, editor, *Encyclopedia of Language & Linguistics (Second Edition)*, pages 660–663. Elsevier, Oxford, second edition edition, 2006.

M. P. Aylett, S. King, and J. Yamagishi. Speech synthesis without a phone inventory. In *Interspeech*, pages 2087–2090, 2009.

F. B. Baker. Information retrieval based upon latent class analysis. *J. ACM*, 9: 512–521, October 1962.

J. Bellegarda, J. Butzberger, Y.-L. Chow, N. Coccaro, and D. Naik. A novel word clustering algorithm based on latent semantic analysis. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 172 –175 vol. 1, may 1996.

E. M. Bender. Linguistically naïve != language independent: why NLP needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, ILCL '09, pages 26–32, 2009.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

C. Benoit, M. Grice, and V. Hazan. The SUS test: A method for the assessment of text-to-speech synthesis intelligibility using Semantically Unpredictable Sentences. *Speech Communication*, 18(4):381 – 392, 1996.

K. Beulen and H. Ney. Automatic question generation for decision tree based state tying. In *ICASSP '98*, volume 2, pages 805–808 vol.2, May 1998.

C. Biemann. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the 21st International Conference on computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, COLING ACL '06, pages 7–12, 2006.

M. Bisani and H. Ney. Bootstrap estimates for confidence intervals in ASR performance evaluation. In *Proc. ICASSP '04*, volume 1, pages 409–12, 2004.

A. Black and A. Font Llitjos. Unit selection without a phoneme set. In *IEEE TTS Workshop 2002*, 2002.

A. Black, P. Taylor, and Caley. *The Festival Speech Synthesis System*, 1999.

A. W. Black. Multilingual speech synthesis. In *Multilingual Speech Processing*, pages 207 – 231. Academic Press, Burlington, 2006.

A. W. Black and K. A. Lenzo. Multilingual text-to-speech synthesis. In *In: Proceedings of the ICASSP 2004*, pages 761–764, 2004.

A. W. Black, K. Lenzo, and V. Pagel. Issues in building general letter to sound rules. In *Proc. of the 3rd ESCA Workshop on Speech Synthesis*, pages 77–80, 1998.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

BNC. *Reference Guide for the British National Corpus (XML Edition)*, 2007. URL `http://www.natcorp.ox.ac.uk/XMLedition/URG/`.

A. Bonafonte and P. D. Agüero. Phrase break prediction using a finite state transducer. In *11th International Workshop on Advances in Speech Technology*, 2004.

H. Borko and M. Bernick. Automatic document classification. *J. ACM*, 10: 151–162, April 1963.

A. Borovsky and J. Elman. Language input and semantic categories: a relation between cognition and early word learning. *Journal of Child Language*, 33(04): 759–790, 2006.

M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20 – 30, 2006.

T. Brants. TnT: a statistical part-of-speech tagger. In *Proc. 6th Conf. Applied Natural Language Processing*, pages 224–231, 2000.

N. Braunschweiler and S. Buchholz. Automatic sentence selection from speech corpora including diverse speech for improved HMM-TTS synthesis quality. In *Proc. Interspeech*, pages 1821–1824, Florence, Italy, Aug. 2011.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall, 1993.

C. Brierley and E. Atwell. ProPOSEC: A prosody and PoS annotated Spoken English Corpus. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010.

E. Brill. A simple rule-based part of speech tagger. In *Proc. 3rd Conf. Applied Natural Language Processing*, pages 152–155, 1992.

P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, 1992.

H. Bunt, J. Alexandersson, J. Carletta, J.-W. Choe, A. Chengyu Fang, K. Hasida, K. Lee, V. Petukhova, A. Popescu-Belis, L. Romary, C. Soria, and D. Traum. Towards an ISO Standard for Dialogue Act Annotation. In *Seventh conference on International Language Resources and Evaluation (LREC'10)*, La Valette, Malta, 2010.

B. Busser, W. Daelemans, and A. van den Bosch. Predicting phrase breaks with memory-based learning. In *Proc. 4th ISCA Speech Synthesis Workshop*, 2001.

M. Candito and B. Crabbé. Improving generative statistical parsing with semi-supervised word clustering. In *IWPT '09: Proceedings of the 11th International Conference on Parsing Technologies*, pages 138–141, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

E. Cantu-Paz and C. Kamath. Inducing oblique decision trees with evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 7(1):54–68, feb 2003.

J. Carletta, S. Isard, G. Doherty-Sneddon, A. Isard, J. C. Kowtko, and A. H. Anderson. The reliability of a dialogue structure coding scheme. *Comput. Linguist.*, 23:13–31, Mar. 1997.

C. Chelba and R. Morton. Mutual information phone clustering for decision tree induction. In *Proc. Int. Conf. on Spoken Language Processing 2002*, 2002.

C. Christodoulopoulos, S. Goldwater, and M. Steedman. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, Cambridge, MA, October 2010. Association for Computational Linguistics.

C. Christodoulopoulos, S. Goldwater, and M. Steedman. A Bayesian mixture model for part-of-speech induction using multiple features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.

G. Chrupała. Efficient induction of probabilistic word classes with LDA. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, Chiang Mai, Thailand, November 2011.

R. A. J. Clark, K. Richmond, and S. King. Multisyn: Open-domain unit selection for the Festival speech synthesis system. *Speech Communication*, 49(4):317–330, 2007.

CMU. *The Carnegie Mellon University Pronouncing Dictionary*. URL `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`.

F. Coenen, P. Leng, R. Sanderson, and Y. J. Wang. Statistical identification of key phrases for text classification. In *Proceedings of the 5th international conference on Machine Learning and Data Mining in Pattern Recognition*, MLDM '07, pages 838–853, 2007.

R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*, 2008.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398, 2011.

B. Comrie. Writing systems. In M. S. Dryer and M. Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Digital Library, Munich, 2011.

M. Creutz and K. Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3:1–3:34, Feb. 2007.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

J. Dijkstra and L. C. W. Pols. Frisian TTS, an example of bootstrapping TTS for minority languages. In *In: Proc. 5th ISCA Speech Synthesis Workshop*, pages 97–102, 2004.

A. D'Ulizia, F. Ferri, and P. Grifoni. A survey of grammatical inference methods for natural language learning. *Artificial Intelligence Review*, 36:1–27, 2011. 10.1007/s10462-010-9199-1.

S. Finch and N. Chater. Bootstrapping syntactic categories using statistical methods. In *Proceedings of the 1st SHOE Workshop on Statistical Methods in Natural Language*, pages 229–235, 1992.

S. Fitt and S. Isard. Synthesis of regional English using a keyword lexicon. In *Proc. Eurospeech 1999*, volume 2, pages 823–826, Sept. 1999.

M. Gales and S. Young. The application of hidden Markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, Jan. 2007.

J. Gao and M. Johnson. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proceedings of the Conference on Em-*

pirical Methods in Natural Language Processing, EMNLP '08, pages 344–352, 2008.

J. Goldsmith and A. Xanthos. Learning phonological categories. *Language*, 85 (1):4–38, 2009.

S. Goldwater and T. Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June 2007.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *PNAS*, 101(suppl. 1): 5228–5235, 2004.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, Mar. 2003.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, Second Edition*. Springer, 2009.

D. Hirst. Form and function in the representation of speech prosody. *Speech Communication*, 46:334 – 347, 2005. Quantitative Prosody Modelling for Natural Speech Description and Generation.

D. Hirst and R. Espesser. Automatic modelling of fundamental frequency using a quadratic spline function. *Travaux de l'Institut de Phonétique d'Aix*, 15:71–85, 1993.

T. K. Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832 –844, Aug. 1998.

T. Honkela, A. Hyvärinen, and J. J. Väyrynen. WordICA – emergence of linguistic representations for words by independent component analysis. *Natural Language Engineering*, 16:277–308, 2010.

D. Horn and I. Axel. Novel clustering algorithm for microarray expression data in a truncated SVD space. *Bioinformatics*, 19(9):1110–1115, 2003.

X. Hu, Z. Cai, D. Franceschetti, P. Penumatsa, and A. C. Graesser. LSA: The first dimension and dimensional weighting. In R. Altermann and D. Hirsh,

editors, *Proceedings of the 25th Meeting of the Cognitive Science Society*, pages 1–6, Boston, 2003. Cognitive Science Society.

C. L. Isbell, Jr. and P. Viola. Restructuring sparse high dimensional data for effective retrieval. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 480–486, Cambridge, MA, USA, 1998. MIT Press.

K. J. Jensen and S. Riis. Self-organizing letter code-book for text-to-phoneme neural network model. In *INTERSPEECH*, pages 318–321, 2000.

G. H. John. *Enhancements to the data mining process*. PhD thesis, Stanford University, 1997.

O. Karaali, G. Corrigan, N. Massey, C. Miller, O. Schnurr, and A. Mackie. A high quality text-to-speech system composed of multiple neural networks. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, volume 2, pages 1237–1240, 1998.

F. Karlsson. Finnish. In E. in Chief: Keith Brown, editor, *Encyclopedia of Language & Linguistics (Second Edition)*, pages 474–475. Elsevier, Oxford, 2nd edition, 2006.

S. Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of IJCNN'98, International Joint Conference on Neural Networks*, pages 413–418, Piscataway, NJ.

H. Kawahara, I. Masuda-Katsuse, and A. Cheveigné. Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: possible role of a repetitive structure in sounds. *Speech Communication*, 27:187–207, 1999.

M. Killer, S. Stüker, and T. Schultz. Grapheme based speech recognition. In *Proc. Eurospeech*, pages 3141–3144, 2003.

S. King and V. Karaiskos. The Blizzard Challenge 2010. In *Proc. Blizzard Challenge Workshop 2010*, Sept. 2010.

G. Knowles, A. Wichmann, and P. Alderson. *Working with Speech: Perspectives on Research into the Lancaster/IBM Spoken English Corpus*. Longman, 1996a.

G. Knowles, B. Williams, and L. Taylor. *A Corpus of Formal British English Speech: The Lancaster/IBM Spoken English Corpus*. Longman, 1996b.

J. Kominek. *TTS From Zero: Building Synthetic Voices for New Languages*. PhD thesis, Carnegie Mellon University, 2009.

J. Kominek and A. Black. The CMU Arctic speech databases. In *Proc. 5th ISCA speech synthesis workshop*, pages 223–224, Pittsburgh, USA, June 2004.

J. Kominek, T. Schultz, and A. W. Black. Voice building from insufficient data – classroom experiences with web-based language development tools. In *Proc. 6th ISCA Speech Synthesis Workshop*, pages 322–327, Bonn, Germany, 2007.

T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *In Proc. ACL/HLT*, 2008.

J. K. Korpela. *Unicode Explained*. O'Reilly Media, 2006.

C. H. A. Koster and M. Seutter. Taming wild phrases. In *Proceedings of the 25th European conference on IR research*, ECIR'03, pages 161–176, 2003.

K. Lagus, M. Creutz, and S. Virpioja. Latent linguistic codes for morphemes using Independent Component Analysis. In A. Cangelosi, G. Bugmann, and R. Borisyuk, editors, *Modeling language, cognition and action: Proceedings of the Ninth Neural Computation and Psychology Workshop (NCPW9)*, September 2005.

M. Lamar, Y. Maron, M. Johnson, and E. Bienenstock. SVD and clustering for unsupervised POS tagging. In *ACL (Short Papers)'10*, pages 215–219, 2010.

T. K. Landauer and S. T. Dumais. A solution to Plato's problem: the Latent Semantic Analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240, 1997.

J. Latorre, K. Iwano, and S. Furui. New approach to the polyglot speech generation by means of an HMM-based speaker adaptable synthesizer. *Speech Communication*, 48(10):1227–1242, 2006.

A. Lenci. Distributional semantics in linguistic and cognitive research. *Italian Journal of Linguistics*, 20(1):1–31, 2008.

W. Li and A. McCallum. Semi-supervised sequence modeling with syntactic topic models. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 2*, pages 813–818, 2005.

K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods*, 28:203–208, 1996.

K. Lund, C. Burgess, and R. A. Atchley. Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pages 660–665, 1995.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, 2011.

K. Maekawa, H. Kikuchi, Y. Igarashi, and J. Venditti. X-JToBI: an extended J-ToBI for spontaneous speech. In *ICSLP-2002*, pages 1545–1548, 2002.

W. C. Mann and S. A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: the Penn Treebank. *Comput. Linguist.*, 19:313–330, June 1993.

B. Merialdo. Tagging English text with a probabilistic model. *Comput. Linguist.*, 20(2):155–171, June 1994.

P. Mermelstein. Automatic segmentation of speech into syllabic units. *The Journal of the Acoustical Society of America*, 58(4):880–883, 1975.

S. Miller, J. Guinness, and A. Zamanian. Name tagging with word clusters and discriminative training. In *Proceedings of HLT*, pages 337–342, 2004.

G. Murray, M. Taboada, and S. Renals. Prosodic correlates of rhetorical relations. In *Proceedings of the HLT-NAACL 2006 Workshop on Analyzing Conversations in Text and Speech*, ACTS '09, pages 1–7, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

E. Navas, I. Hernez, and I. Sainz. Evaluation of automatic break insertion for an agglutinative and inflected language. *Speech Communication*, 50(11-12):888 – 899, 2008. Iberian Languages.

C. Odé. ToRI, A Transcription of Russian Intonation. An Interactive Research Tool and Learning Module on the Internet. In *Dutch Contributions to the Fourteenth International Congress of Slavists*, pages 431–449, 2008.

J. J. Odell. *The Use of Context in Large Vocabulary Speech Recognition*. PhD thesis, University of Cambridge, 1995.

M. Ostendorf, P. J. Price, and S. Shattuck-Hufnagel. The Boston University Radio News Corpus. Technical report, Boston University, Mar. 1995.

A. Parlikar and A. W. Black. A grammar based approach to style specific phrase prediction. In *Interspeech*, pages 2149–2152, Florence, Italy, August 2011.

K. Prahallad and A. W. Black. Handling large audio files in audio books for building synthetic voices. In *Proc. Speech Synthesis Workshop 2010*, pages 148–153, Nara, Japan, Sept. 2010.

U. Quasthoff, M. Richter, and C. Biemann. Corpus portal for search in monolingual corpora. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC 2006)*, pages 1799–1802, 2006.

E. Raghavendra and K. Prahallad. A multilingual screen reader in indian languages. In *Communications (NCC), 2010 National Conference on*, pages 1 –5, jan. 2010.

A. Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proc. Conf. Empirical Methods in NLP*, May 1996.

I. Read and S. Cox. Stochastic and syntactic techniques for predicting phrase breaks. *Computer Speech and Language*, 21(3):519 – 542, 2007.

M. Redington, N. Chater, and S. Finch. Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, 22(4):425–469, 1998.

R. Řehůřek. Fast and faster: A comparison of two streamed matrix decomposition algorithms. *CoRR*, abs/1102.5597, 2011.

R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

P. Roach, G. Knowles, T. Varadi, and S. Arnfield. Marsec: A machine-readable spoken english corpus. *Journal of the International Phonetic Association*, 23 (2):47–54, 1993.

T. T. Rogers, M. A. L. Ralph, P. Garrard, S. Bozeat, J. L. McClelland, J. R. Hodges, and K. Patterson. Structure and deterioration of semantic memory: A neuropsychological and computational investigation. *Psychological Review*, 111(1):205–235, 2004.

RPART. *Package 'rpart': Recursive partitioning and regression trees.* URL `http://cran.r-project.org/web/packages/rpart/rpart.pdf`.

M. Sahlgren. *The Word-Space Model: Using Distributional Analysis to Represent Syntagmatic and Paradigmatic Relations between Words in High-Dimensional Vector Spaces.* PhD thesis, Stockholm University, Stockholm, Sweden, 2006.

G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, November 1975.

H. Schmid and M. Atterer. New statistical methods for phrase break prediction. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

H. Schütze. Dimensions of meaning. In *Supercomputing '92: Proceedings of the 1992 ACM/IEEE conference on Supercomputing*, pages 787–796, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.

H. Schütze. Word space. In *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan Kaufmann, 1993.

H. Schütze. Distributional part-of-speech tagging. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 141–148, 1995.

H. Schwenk and J.-L. Gauvain. Connectionist language modeling for large vocabulary continuous speech recognition. In *In International Conference on Acoustics, Speech and Signal Processing*, pages 765–768, 2002.

I. Shafran and M. Ostendorf. Acoustic model clustering based on syllable structure. *Computer Speech & Language*, 17(4):311–328, 2003.

K. Shalonova and R. Tucker. South asian languages in multilingual tts-related database. In *EACL Workshop on Computational Linguistics for the Languages of South Asia*, pages 57–63, Budapest, April 2003.

K. Shinoda and T. Watanabe. MDL-based context-dependent subword modeling for speech recognition. *Acoustical Science and Technology*, 21(2):79–86, 2000.

K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirschberg. ToBI: A standard for labeling English prosody. In *International Conf. on Spoken Language Processing*, volume 2, pages 867–870, Banff, 1992. International Conf. on Spoken Language Processing.

R. Singh, B. Raj, and R. Stern. Automatic clustering and generation of contextual questions for tied states in hidden Markov models. In *Proc. ICASSP '99*, volume 1, pages 117–120, Mar 1999.

H. Somers, G. Evans, and M. Zeinab. Developing speech synthesis for under-resourced languages by "Faking it": An experiment with Somali. In *5th International Conference on Language Resources and Evaluation*, pages 2578–2581, Genoa, Italy, May 2006.

R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. Normalization of non-standard words. *Computer Speech & Language*, 15(3): 287 – 333, 2001.

A. Stan, J. Yamagishi, S. King, and M. Aylett. The Romanian speech synthesis (RSS) corpus: Building a high quality HMM-based speech synthesis system using a high sampling rate. *Speech Communication*, 53(3):442 – 450, 2011.

M. Steyvers and T. Griffiths. Probabilistic topic models. In T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning.* Laurence Erlbaum, 2006.

A. Stolcke and E. Shriberg. Automatic linguistic segmentation of conversational speech. In *Proc. ICSLP*, pages 1005–1008, 1996.

X. Sun and T. H. Applebaum. Intonational phrase break prediction using decision tree and n-gram model. In *Proc. Eurospeech 2001*, pages 537–540, Sept. 2001.

D. Surendran and G.-A. Levow. Dialog act tagging with support vector machines and hidden Markov models. In *Proc. Interspeech*, pages 1950–1953, 2006.

É. Székely, J. P. Cabral, P. Cahill, and J. Carson-Berndsen. Clustering expressive speech styles in audiobooks using glottal source parameters. In *Proc. Interspeech*, pages 1821–1824, Florence, Italy, Aug. 2011.

P. Taylor and A. W. Black. Assigning phrase breaks from part-of-speech sequences. *Computer Speech & Language*, 12(2):99 – 117, 1998.

P. Taylor, A. W. Black, and R. Caley. Heterogeneous relation graphs as a formalism for representing linguistic information. *Speech Communication*, 33(1–2): 153–174, 2001.

M. I. Tietze, A. Winterboer, and J. D. Moore. The effect of linguistic devices in information presentation messages on comprehension and recall. In *ENLG '09: Proceedings of the 12th European Workshop on Natural Language Generation*, pages 114–117, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. Speech parameter generation algorithms for hmm-based speech synthesis. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1315 –1318 vol.3, 2000.

R. Tucker and K. Shalonova. Supporting the creation of TTS for local language voice information systems. In *Proc. Interspeech 2005*, pages 453–456, Lisbon, Portugal, Sept. 2005.

J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA, 2010.

E. Tuv, A. Borisov, and K. Torkkola. Feature selection using ensemble based ranking against artificial contrasts. In *International Joint Conference on Neural Networks*, pages 2181–2186, 2006.

E. Tuv, A. Borisov, G. Runger, and K. Torkkola. Feature selection with ensembles, artificial variables, and redundancy elimination. *J. Mach. Learn. Res.*, 10:1341–1366, December 2009.

M. Twain. *A tramp abroad*. American Pub. Co. ; Chatto & Windus, Hartford, Conn. : London :, 1880.

A. Ushioda. Hierarchical clustering of words. In *Proceedings of the 16th conference on Computational linguistics - Volume 2*, COLING '96, pages 1159–1162, 1996.

M. Vainio, A. Suni, and P. Sirjola. Accent and prominence in finnish speech synthesis. In *Proceedings of the 10th International Conference on Speech and Computer (Specom 2005)*, pages 309–312, Oct. 2005a.

M. Vainio, A. Suni, and P. Sirjola. Developing a finnish concept-to-speech system. In *Proceedings of the Second Baltic Conference on Human Language Technologies*, pages 201–206, April 2005b.

J. Černocký. *Speech Processing Using Automatically Derived Segmental Units: Applications to Very Low Rate Coding and Speaker Verification*. PhD thesis, Universite Paris-Sud, Dec 1998.

M. Q. Wang and J. Hirschberg. Automatic classification of intonational phrase boundaries. *Computer Speech & Language*, 6(2):175 – 196, 1992.

O. Watts, J. Yamagishi, and S. King. The role of higher-level linguistic features in HMM-based speech synthesis. In *Proc. Interspeech*, pages 841–844, Makuhari, Japan, Sept. 2010a.

O. Watts, J. Yamagishi, and S. King. Letter-based speech synthesis. In *Proc. Speech Synthesis Workshop 2010*, pages 317–322, Nara, Japan, Sept. 2010b.

O. Watts, J. Yamagishi, and S. King. Unsupervised continuous-valued word features for phrase-break prediction without a part-of-speech tagger. In *Proc. Interspeech*, Florence, Italy, Aug. 2011.

D. Widdows. *Geometry and Meaning*. Center for the Study of Language and Information, Stanford, CA, 2004.

F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

B. Williams. The formulation of an intonation transciption system for british english. In G. Knowles, B. Williams, and L. Taylor, editors, *A Corpus of Formal British English Speech: The Lancaster/IBM Spoken English Corpus*, pages 38–57. Longman, 1996.

Z. Xie and P. Niyogi. Robust acoustic-based syllable detection. In *Proceedings of the ICSLP, International Conference on Spoken Language Processing*, 2006.

J. Yamagishi and O. Watts. The CSTR/EMIME HTS System for Blizzard Challenge. In *Proc. Blizzard Challenge 2010*, Sept. 2010.

T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Duration modeling for hmm-based speech synthesis. In *ICSLP*, 1998.

H. Zen. Speaker and language adaptive training for hmm-based polyglot speech synthesis. In *Proc. Interspeech 2010*, pages 410–413, 2010.

H. Zen, T. Toda, M. Nakamura, and K. Tokuda. Details of Nitech HMM-based speech synthesis system for the Blizzard Challenge 2005. *IEICE Trans. Inf. & Syst.*, E90-D(1):325–333, Jan. 2007.