University of Washington

Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Ivan Bulyko

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:

_____

Mari Ostendorf

Reading Committee:

_____

Mari Ostendorf

_____

Alex Acero

_____

Jeffrey Bilmes

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Bell and Howell Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

Flexible Speech Synthesis
Using Weighted Finite State Transducers

by Ivan Bulyko

Chair of Supervisory Committee:

Professor Mari Ostendorf
Electrical Engineering

The main focus of this thesis is on improving the quality of concatenative speech synthesis by taking advantage of the natural (allowable) variability in spoken language, namely, the fact that there are multiple ways of uttering a given sentence and there are several word sequences that can represent a given concept. An architecture for speech generation for constrained domain applications is proposed that tightly integrates language generation and speech synthesis, allowing the choice of words and desired intonation in the system's response to be optimized jointly with the speech output quality. Experiments with a travel planning dialog system have demonstrated that by expanding the space of candidate responses and possible prosodic realizations we achieve higher quality speech output.

The additional flexibility in terms of word sequences, prosodic realizations and pronunciations increases the search space and, consequently, the computational cost of the synthesis system. To address this problem this thesis also offers improvements to the popular unit selection approach for more accurately constraining or pruning the search space at the acoustic level. In particular, we describe a variation to the

cluster-based unit database design aimed at constraining the set of candidate units, and we introduce splicing costs into the unit search criterion as a measure to indicate which unit boundaries are particularly good or poor join points, augmenting existing concatenation measures for better pruning of the search space. As a byproduct, the new splicing costs also lead to improvements in speech quality.

Finally, we introduce a modular speech synthesis system architecture where each component is represented with weighted finite-state transducers (WFSTs), and we describe specific WFST implementations of prosody prediction and unit selection modules. Such an architecture provides an efficient representation of flexible targets and allows the steps in the synthesis process to be performed with operations available in a general purpose toolbox.

TABLE OF CONTENTS

List of Figures **iv**

List of Tables **vii**

Chapter 1: Introduction **1**

1.1 Motivation 1

1.2 Proposed Approach 4

1.3 Thesis Outline 5

Chapter 2: Background **8**

2.1 Speech Synthesis 8

2.2 Prosody Prediction 10

2.2.1 Symbolic Prosody Markers 12

2.2.2 Continuous Acoustic Parameters 14

2.3 Waveform Generation 15

2.4 Unit Selection 17

2.4.1 Synthesis Unit 17

2.4.2 Selecting Units from Multiple Candidates 19

2.5 Waveform Modification 22

2.6 Integrating Prosody in Unit Selection 23

2.7 Tying Language Generation and Speech Synthesis in Dialog Systems 25

2.8 WFSTs for Speech Processing 26

Chapter 3: Corpora **32**

3.1 Communicator 32

3.2 Radio News 35

3.3 Prosodic Labeling 36

Chapter 4: Integrated Response Generation **38**

4.1 Response Generation Components 38

4.1.1 Overall Architecture 38

4.1.2 Language Generation 40

4.1.3 Prosody Expansion 41

4.1.4 Waveform Generation 44

4.2 Integrating Generation and Synthesis with WFSTs 46

4.2.1 Prosody Prediction 46

4.2.2 Unit Selection 48

4.2.3 WFST Composition 49

4.3 Experiments 51

4.3.1 Prosody Prediction 51

4.3.2 Synthesizing with Multiple Prosodic Targets 53

4.3.3 Flexible Text Generation and Prosody Prediction 55

4.4 Summary 57

Chapter 5: Search Space Reduction for Efficient Unit Selection **59**

5.1 Computational Aspects of Unit Selection 59

5.2 Clustering Similar Units 61

5.2.1 Computing Acoustic Distance 62

5.2.2 Building Decision Trees 64

5.2.3 Adjusting the Clusters 69

5.3 Models of Spectral Dynamics 71
5.3.1 Dynamic Features and HMMs 71
5.3.2 Dynamic Model: BMM 73
5.3.3 Dynamic Models for Unit Selection 75
5.4 Unit Selection with Splicing Costs 76
5.4.1 Splicing Costs as the Inverse of Spectral Change 77
5.4.2 Computing Splicing Costs with BMMs 79
5.5 Redefining Join Points 81
5.6 Summary 86

Chapter 6: System Implementation with WFSTs 87
6.1 Flexible Target Specification 89
6.1.1 Prosodic Variability 89
6.1.2 Using Alternative Pronunciations 91
6.2 Efficient Unit Selection 93
6.2.1 Modeling Context Dependency with WFSTs 93
6.2.2 Unit Selection by WFST Composition 95
6.3 Performance Considerations 98

Chapter 7: Conclusion 104
7.1 Summary of Results and Contributions 104
7.2 Open Questions 106
7.3 Use of Hand-Labeled Data 107
7.4 Opportunities for Further Advances 108

Bibliography 110

LIST OF FIGURES

2.1 Block diagram of a speech synthesis system. 9

2.2 Speech waveform with prosodic labels and F0 contour 11

2.3 Concatenative approach to waveform generation. 16

2.4 Cost functions for selecting units from multiple candidates: C^t is the target cost, and C^c is the concatenation cost. 20

2.5 An example of a weighted finite-state transducer W_d. 28

2.6 An example of a non-minimal and non-deterministic WFST W_n. . . . 28

2.7 Input sequence $red - green$ represented as an FSA I. 29

2.8 The result of the composition $I \circ W_d$. 30

4.1 Communication strategy between the language generator and speech synthesizer. 39

4.2 An example of a word and phrase lattice representing six alternative wordings of a prompt. 42

4.3 Representing prosodic variability in the form of a network of prosodic templates. In brackets are ToBI labels for accent and boundary tones. 43

4.4 Frames of spectral features used in computing the concatenation cost between units U_i and U_j. 45

4.5 Modular structure of our prosody model, where $\oplus$ indicates union and $\circ$ indicates composition operations on WFSTs. 47

4.6 A simple decision tree and its WFST representation, where F is a prediction variable and $\{s, t\}$ are the possible class labels. 48

4.7 An example of a finite-state network of targets containing multiple word sequences and various prosodic specifications from a template prosody WFST. Values in boxes describe prosody, i.e breaks, accents and tones. Empty boxes correspond to an absence of any prosodic labels. 50

5.1 Aligning frames in order to compute the distance between two units U and V. 62

5.2 Radio News corpus 67

5.3 A histogram itemized for each individual feature (Radio News corpus). 68

5.4 The average cluster impurity as a function of the number of units added to the cluster (Radio News corpus). Pruning ensures that the cluster size remains the same. 72

5.5 A histogram of BMM dependencies according to their relative position (in frames). 80

5.6 Feature dependencies used for computing splicing costs for unit U_i. Dotted lines show frame boundaries of LSF vectors. Circles are individual features. 81

5.7 Average cost per unit in the best path, as a function of the number of boundaries made unavailable for making a splice. 83

5.8 Comparison of off-line and on-line search pruning in terms of the average cost per unit in the final path (Radio News). 85

6.1 Internal structure of our subword-unit-based synthesizer. Symbols denote individual WFST components. 88

6.2 Input (I) phrase "to Boston" annotated with labels α_i. 90

6.3 An example of prosody-annotated network of words (Π). 91

6.4 A fragment of the network Φ containing a phonetic representation of the word "to". 91
6.5 A fragment of the lexicon Λ, containing words "to" and "Boston". The output labels S_i are hypothetical indices to syllable structure tags. Empty string ε is used as input in order to insert additional output labels. 92
6.6 High-level context information transducer H. 94
6.7 Context-dependency transducer G. 94
6.8 WFST implementation of the unit selection module U. Shown are two unit clusters and a VQ concatenation network of size 4. Costs are labeled as follows: $T(u)$ is the target cost for unit u; $C(i, j)$ is the concatenation cost between vector quantized frames i and j; $C(\#, i)$ and $C(i, \#)$ costs of transitioning from silence to VQ frame i and vice versa; $S\_left(u)$ and $S\_right(u)$ are left and right splicing costs for unit u. 97
6.9 Deterministic context-dependency transducer N_d for combining high level context C_i and phonetic context to obtain decision tree clusters ω_i, where ω is the target phone. 102

LIST OF TABLES

3.1 The speech corpora. 35

3.2 Relative frequency of various prosodic labels in the Communicator (C) and Radio News (R) corpora. DS stands for downstepped, bd. stands for boundary. 37

4.1 Prosody prediction results, where "chance" performance is computed by always predicting the most frequent label. Labeler agreement is based on results reported in [69]. For pitch accents, labeler agreement (72.4%) was computed using three categories: none, non-downstepped and downstepped. 53

4.2 Perceptual experiment results: synthesizing with multiple prosodic targets. Shown are pairwise comparisons: A_1 vs. B_1 and B_1 vs. C_1, counting only those cases where the utterances were different. Significance of the results is measured by the probability P of sample medians not being significantly different. 54

4.3 Differences in synthesis methods used to produce the versions evaluated in the perceptual experiments. 56

4.4 Perceptual experiment results: flexible text generation and prosody prediction. 57

5.1 Features used for clustering. 66

6.1 Notation used to define sizes of certain WFSTs. 99

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to Mari Ostendorf, my adviser and teacher, for giving me the opportunity to work in the area of speech synthesis and for setting high standards and guiding me toward the goal that at times seemed light years away.

I would like to thank the members of my committee. Jeff Bilmes, for sharing his expertise in graphical models and motivating new and exciting directions of research. Alex Acero, who helped me achieve more in my research in terms of making the technology useful in a wide range of applications.

Many thanks to all of the SSLI lab members and especially those who helped with the transition from Boston University. Becky Bates, Ozgur Cetin, Randy Fish, Katrin Kirchhoff, Scott Otterson, David Palmer, Izhak Shafran, and other colleagues.

I would like to acknowledge Bryan Pellom for his invaluable help with the Communicator, and Lesley Carmichael for prosodically labeling corpora.

I am infinitely thankful to my parents for being so brave to support their only son's decision to move eleven time zones away.

And my special thanks to Melissa for her love, support and understanding.

This work was supported in part by the National Science Foundation, grant No. (IIS-9528990).

Chapter 1

INTRODUCTION

1.1 Motivation

Improvements in automatic speech recognition (ASR) have led to many new deployments of speech-enabled computer interfaces. These interfaces are successful, in part, because they focus on a limited domain of application, e.g. travel planning, weather information, baggage lost-and-found, stock quotations, etc. Such systems require high quality speech output for user acceptance in telephone-based applications, where speech is the only medium used for communication. While applications with very limited capabilities can make use of pre-recorded voice prompts, many applications require generation of a variety of responses which can only be handled by synthesizing speech. The simple approach is to use a general text-to-speech synthesis system, but this may not be good enough for some applications largely due to its lack of naturalness. Higher quality synthesized speech can be obtained using a system tailored for the specific domain. Depending on the scope of the domain, designing such a system may involve hours of recordings in order to sufficiently represent the variety of potential responses. Moreover, minor modifications to the text generator may require adjustments to the synthesizer and additional recordings from the target speaker.

This thesis will focus on improving the quality of speech synthesis by making it more flexible and capable of benefiting from the variability that natural languages possess, namely the fact that it is possible to convey essentially the same meaning with two acoustically different utterances. There are three main contributions of this

work: 1) the introduction of flexible targets (via a network representation) and a tight integration (joint optimization) of language generation and speech synthesis for improved speech responses in a dialog system, 2) improvements to the widely used unit selection approach to speech synthesis, and 3) a modular system architecture accommodating flexible generation and providing efficient implementation of the synthesis component via a general purpose weighted finite-state transducer (WFST) toolbox.

One of the key problems with current systems is poor *prosody* prediction, where prosody includes the phrase, accent and tone structure of speech, which helps to convey the meaning of an utterance to the listener by marking its information structure and drawing attention to specific lexical items. Phrase structure corresponds to perceived groupings of words in an utterance and is related to syntactic structure. Pitch accents indicate relative strength of different words, indicating such things as novelty and contrast. Tones mark different fundamental frequency movements, which can give cues to discourse structure. Prosodic patterns are difficult to predict in part because they depend on high level factors, such as syntax and discourse, that have been less well studied than phoneme segments in terms of their acoustic consequences, and in part because this information is difficult to extract from unrestricted text. The acoustic correlates of prosody, including fundamental frequency (F0) and duration (or timing), are also difficult to model because the same cue can be affected by factors at multiple time scales: phrase, syllable, segment.

Recently, very high quality concatenative synthesis has been achieved by splicing together segments of speech for cases where the application domain is very constrained, essentially ignoring the problem of prosody prediction. In less constrained text-to-speech applications, however, finding long continuous segments of recorded speech to match input text becomes less probable, thus smaller segments must be concatenated to form the output speech. Furthermore, finding segments with appropriate prosody is also less probable; therefore, before these segments can be combined into a continuous waveform, they must undergo signal modifications in order to match

target F0 and duration specifications. Substantial changes in fundamental frequency can compromise sound quality of speech output, which raises the importance of selecting segments (also called *units*) that match the target well.

Automatic generation of language in dialog systems provides an opportunity to pass more information to the prosody prediction module than would be available in unrestricted text-to-speech synthesis. One example of such additional information that has often been cited is the annotation of automatically generated syntactic and semantic structure as well as dialog context for purposes of improved prosody prediction. While using this information is important for prosody prediction, our focus is on a different approach that takes advantage of advances in concatenative speech synthesis: that is, to provide flexibility to the synthesizer in terms of possible responses.

The flexibility in system responses and, hence, the opportunity to select units from a larger pool of candidates can potentially benefit the output quality as there is a better chance of finding a sequence that would match the target specification and have minimal distortion associated with transitions from one unit to the next. A larger search space, however, increases the computational cost and potentially the implementation complexity. Real-time performance required from speech synthesis systems makes the implications of the speech model for the unit selection search very important. In this work, we introduce changes to the concatenation model to address the efficiency of unit selection and propose a system architecture that supports language flexibility represented by alternative pronunciations and prosodic sequences.

While this work largely focuses on limited domain applications, many of the contributions can generalize to unrestricted text-to-speech. In particular, unit clustering and unit selection techniques as well as the system implementation with WFSTs are designed for concatenative synthesis in general.

1.2 Proposed Approach

Our approach to improving the quality of limited domain speech synthesis systems is to provide flexibility to the synthesizer in terms of possible responses. In particular, we take advantage of two main areas of flexibility: choice of wording and prosodic realization of an utterance. The key idea is that there is more than one acceptable response at any point in a dialog. Instead of passing a single text string (or prosodically annotated text string) to a synthesizer, we pass an annotated network. Thus, the search for wording, prosody prediction and speech units is jointly optimized. In other words, instead of predicting a specific word sequence and prosodic realization first and then searching for units to match that target, our approach effectively makes a "soft" decision about the target words and prosody and evaluates alternative realizations of a given utterance. In this work we propose a general architecture for speech generation, based on: 1) representing a response in terms of an annotated word network rather than a single word sequence, and 2) using a symbolic representation of prosodic structure for annotation. We also describe a specific implementation using a template-based language generator and a variable-length unit selection synthesis system.

Since providing more flexibility increases the search space, it also increases the computational cost and potentially the implementation complexity. Thus, a key to making this approach practical is an efficient implementation. We use weighted finite-state transducers to represent data structures at each stage in the synthesis process. Each step of network expansion is then followed by minimization, and both can be implemented using a general purpose toolbox. The computational aspects of the system are especially important as we addresses the challenges of less constrained domains and unrestricted text-to-speech, which require selecting units from a large inventory of sub-word segments. Our focus is on efficient implementation of unit selection for concatenative speech synthesis. We develop a methodology for automatically estimating

the suitability of a given unit to have a splice made at its boundary prior to performing the unit selection search process, which reduces the computation at run time and achieves smoother concatenations. It also opens opportunities for new unit inventory and cost design techniques. Our system implementation is based on the WFST formalism and designed to accommodate flexibility (in terms of alternative wordings, pronunciations and prosodic targets) and efficiency (in terms of computational cost).

This work makes use of several techniques borrowed from the speech recognition technology. In particular, we use various speech signal models, decision tree clustering techniques and dynamic search and pruning strategies, that can be found in many speech recognition applications. The use of WFSTs for joint search has also been applied to speech recognition. While there are fundamental differences between speech synthesis and speech recognition, these two fields overlap significantly and can benefit from each other by sharing technology (see [41] for a review of both fields).

1.3 Thesis Outline

As described earlier, the focus of this thesis is on improving the quality of speech synthesis by taking advantage of the flexibility (phonetic, prosodic and wording choices) that natural languages offer. We describe a new architecture for integrated language and speech generation in limited domain applications, extensions to the standard concatenative approach and an efficient implementation architecture with weighted finite-state transducers that allows us to exploit the flexibility of language without a dramatic increase in computational complexity of the system. In this section we provide an overview of the thesis outlining the presentation of these contributions and the infrastructure that they build on.

We start with background material in Chapter 2 by explaining some of the challenges of concatenative speech synthesis and providing a review of previous work in this area. In particular, we describe individual components of a general purpose syn-

thesizer: prosody prediction, unit selection and waveform modification. We also talk about the issues related to language generation and speech synthesis in dialog systems. We then review some applications of weighted finite-state transducers in speech processing, since our system implementation relies on the WFST methodology.

In Chapter 3, we describe the two corpora used in our experiments. The first speech database contains recordings of system responses from a travel planning dialog system. The other corpus is comprised of news articles spoken by a professional radio announcer.

Chapter 4 describes how language generation and speech synthesis for spoken dialog systems can be efficiently integrated under a weighted finite-state transducer architecture. Taking advantage of this efficiency, we show that introducing flexible targets in generation and prosody prediction leads to more natural sounding synthesis.

In Chapter 5, we focus on reducing the computational complexity of unit selection in concatenative speech synthesis. In particular, we describe a variation to the cluster-based unit database design aimed to constrain the search space, and we introduce splicing costs into the unit search criterion as a measure to indicate which unit boundaries are particularly good or poor join points. Splicing costs help reduce search space at run time via on-line pruning and/or constraining the space of potential splicing points during database construction.

In Chapter 6, we explain how a speech synthesis system can be efficiently implemented using weighted finite-state transducers. We develop a modular system architecture by representing each component as a WFST. This work is the first to offer a complete system design that can support a large database of subword units in the context of limited domain synthesis. Further, when combined with a text analysis module, the same architecture can be used for full unrestricted text-to-speech capability. We also improve the efficiency and extend the capabilities of the unit selection module, over what has previously been proposed in [111]. The main aspects of the system this chapter covers in detail are: generation of flexible prosodic and phonetic

targets, and unit selection which involves identifying candidate clusters and finding the optimal unit sequence.

Chapter 7 concludes this thesis. First, we summarize the main contributions of this work. Then we list several questions that were raised but remained unanswered throughout the thesis. Finally, we outline some future directions for research in dialog systems and concatenative speech synthesis.

Chapter 2

BACKGROUND

2.1 Speech Synthesis

Speech synthesis is typically performed in four phases (Figure 2.1). The first phase is the analysis or generation of input text. In unrestricted text-to-speech synthesis, input text can come from any source and may contain ambiguous words. Text analysis addresses numerous issues that include but are not limited to: word/sentence segmentation, abbreviation and numeral expansion, word pronunciation, homograph disambiguation, part-of-speech tagging, syntactic parsing and simple discourse analysis. For example, in sentence "She lives at 123 St. Mary's St.", we have an address that contains a number pronounced as *one twenty three* and two abbreviations "St.", where the first stands for *saint* and the second stands for *street*. Moreover the synthesizer must know that "lives" is a verb (vs. a plural noun) in order to pronounce it correctly. In human-computer dialog systems computer responses are typically produced by a text generation module. The text generator can provide information relevant to the meaning of the utterance, rendering text analysis unnecessary. In our work we will assume access to text analysis or generation components developed by others.

The next step in the speech synthesis process is symbolic parameter generation. It involves conversion of words into a sequence of phonemes with prosodic structure specification. Determining the appropriate phoneme sequence and specifying the prosodic structure are often treated as separate problems. First, a canonical pronunciation is found by either a dictionary lookup or by applying default letter-to-sound

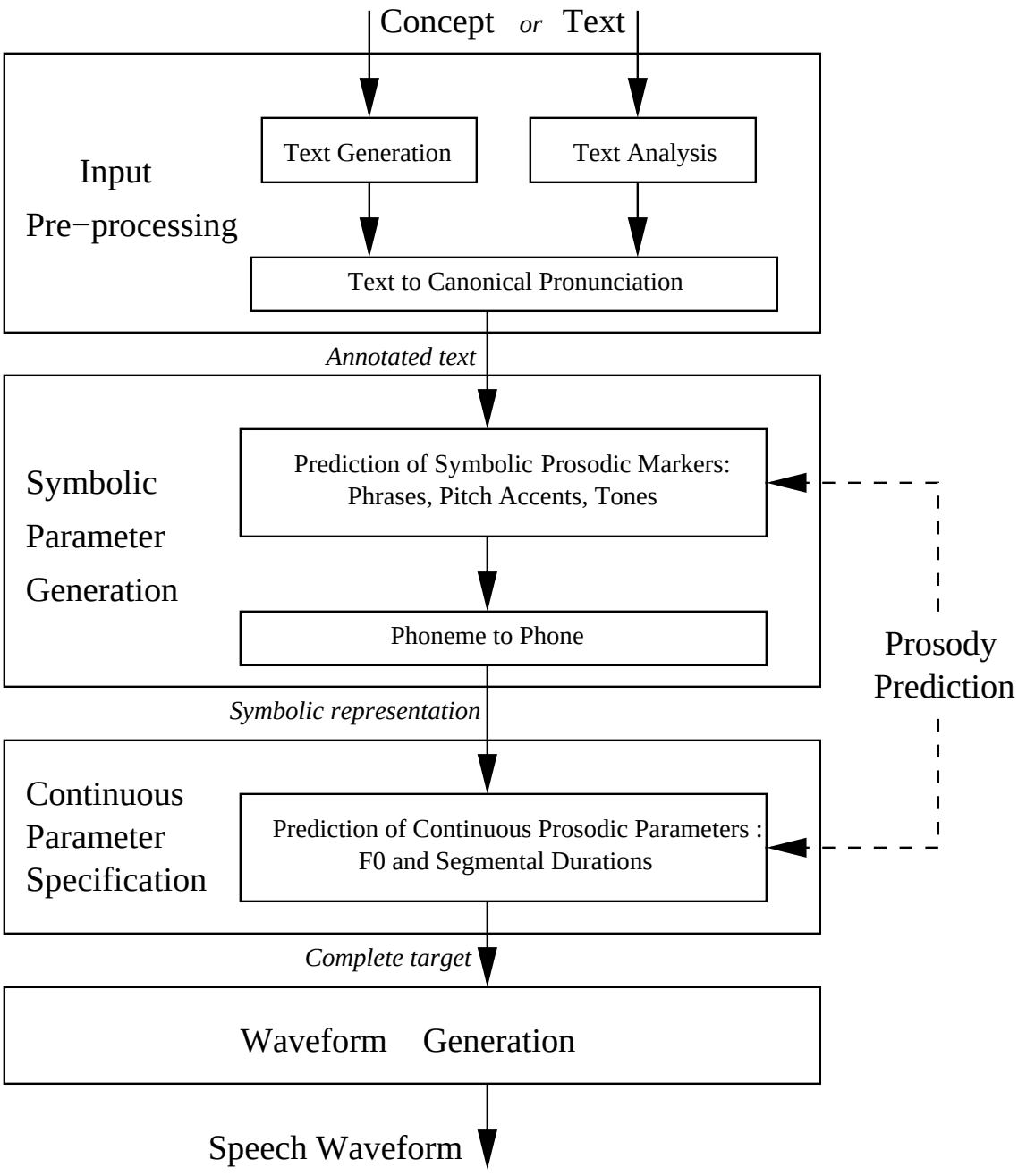

Figure 2.1: Block diagram of a speech synthesis system.

rules, when the word is not found in the dictionary. Later, the canonical pronunciation might be adjusted depending on prosody. In our work we will assume that all necessary word transcriptions are available via a dictionary lookup. The prosodic structure predicted at this stage consists of symbolic markers indicating emphasis and phrasing, as discussed further in Section 2.2.

The third phase of speech generation involves predicting continuous acoustic parameters such as segmental duration and the F0 contour, based on the symbolic prosodic markers and phonemic transcription. Then in the last phase, spectral and source parameters (or speech segments) are chosen based primarily on the phonetic labels, and the speech waveform is produced in accordance to the acoustic specification. As shown in Figure 2.1, prosody prediction spans both the second and the third phases, as we first predict symbolic and then continuous prosodic parameters. The problems of prosody prediction and waveform generation are discussed in more detail in the next two sections.

2.2 Prosody Prediction

Prosody includes the phrase, prominence and tonal structure of speech. It marks the information structure of an utterance, from syntax to topic structure, and also helps to convey speaker intentions. As pointed out in the previous section, prosody is described at both an abstract symbolic level and in terms of continuously varying parameters. Figure 2.2 illustrates these aspects. The top window in Figure 2.2 shows a speech waveform that corresponds to the utterance "Wanted: Chief Justice of the Massachusetts Supreme Court." The other two windows are time-aligned with the waveform. Symbolic prosodic labels, i.e. tones and breaks, marked in accordance to the ToBI [87] system of prosodic transcriptions, along with the word labels are displayed in the middle window. The tones and breaks were labeled by a human labeler, but that still leaves room for uncertainty. Note that the tone marked "*?" corre-

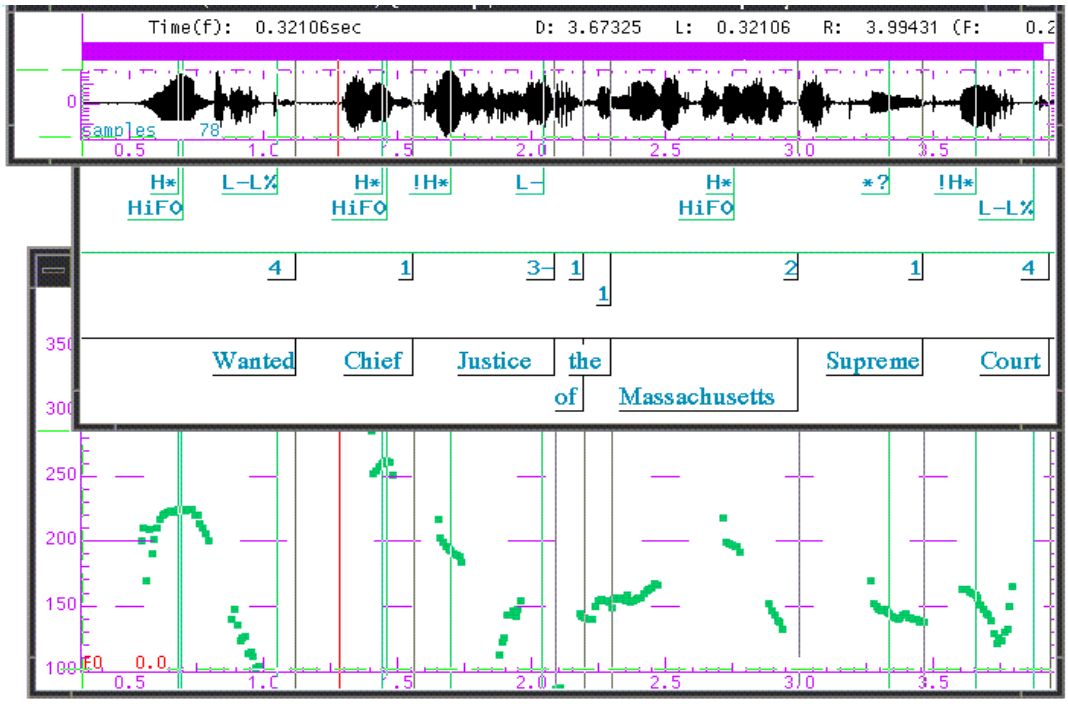

Figure 2.2: Speech waveform with prosodic labels and F0 contour

sponds to the labeler being unsure about the presence of a pitch accent. The bottom window shows the F0 contour measured in Hertz. The F0 values were extracted automatically by Entropic pitch tracker. The F0 points are continuous-valued, but do not comprise a continuous contour in time, because reliable measurements are only possible in voiced regions of speech, plus automatic pitch tracking is far from being error-proof. F0 events at major phrase boundaries, marked by boundary tones, can give cues to discourse properties of the utterance, i.e. a decline in F0 contour is often observed at the end of a statement, while a final F0 rise is characteristic of a question.

The symbolic and continuous aspects of prosody are usually predicted in two steps: 1) prediction of symbolic markers to indicate emphasis (or, pitch accents) and phrasing, and 2) generation of continuous acoustic control parameters, such as

segmental duration and the F0 contour. The focus of this work will be on the symbolic aspect of prosody, though both aspects are briefly reviewed next.

2.2.1 Symbolic Prosody Markers

There are two main types of prosodic events: phrase breaks and pitch accents. Prosodic phrases comprise a hierarchical structure. Different break labels (0 through 4 in ToBI [69] notation) correspond to different levels of phrasing, where 4 marks a major phrase boundary, 3 corresponds to an intermediate (minor) prosodic phrase, 1 and 2 typically reflect word boundaries, and 0 marks an absence of a break, i.e. two words being cliticized. In our example, the utterance is split into two major prosodic phrases, and the second major phrase consists of two intermediate phrases, as illustrated by the brackets below.

[[Wanted:]]	[[Chief	Justice]	[of	the	Massachusetts	Supreme	Court.]]
4	1	3	1	1	2	1	4

ToBI reliability studies [87] have indicated that labelers agreed on the same break index 69% of the time, and the index labels were different by at most 1 level 94% of the time. In our work, we will only use 3 levels (4, 3, other) to reduce the system complexity associated with allowing multiple prosodic labels in prosody prediction.

Major (4) and minor (3) phrases are associated with edge marking tones, indicated by "-" and "%" in the tone window. Phrasal prominence, sometimes called sentence stress, is typically associated with a pitch accent, i.e. tones marked with "*" using the ToBI system illustrated in Figure 2.2. Tones labeled "H*" and "!H*" mark pitch accents that are typically associated with high F0 values; "L*" is associated with a low F0. Each accent however has a different F0 associated with it, some being significantly lower than others. When an accent has F0 lower than that of the

preceding accent in the same prosodic phrase, it is marked as *downstepped* (exclamation mark in ToBI notation). In our example, a lower "Justice" preceded by a higher "Chief" is marked downstepped. Downstep operates within a minor phrase. Hence, when we look at non-adjacent pitch accents or those belonging to different phrases, downstepped accents do not always have to be lower than non-downstepped (compare "Justice" and "Massachusetts"). The "HiF0" labels mark accents that correspond to the highest F0 point in a given minor prosodic phrase (excluding pitch tracking errors and segmental effects). The reliability studies [87] of the ToBI system of prosodic transcriptions have shown a significant disagreement (36%) among human labelers when assigning specific tone labels to pitch accents, while only 14% disagreed on accent location (19% reported in [69] on different corpora). Marking accents with downstep labels appeared to be somewhat more reliable than all tones, resulting in 21% disagreement (28% reported in [69]). In our work, to alleviate the data sparsity and to lessen the effects of labeling inconsistency we plan to group all pitch accents into three categories: low accents, high accents and downstepped high accents.

Accent prediction algorithms base their decision on lexical stress of the syllable, the part-of-speech class of a given word [78], structure of complex noun phrases [61, 88], and discourse properties such as focus and given/new status [36, 19], and reflection of theme/rheme [94]. *Prosodic phrase* prediction algorithms, besides relying on punctuation, utilize information about phrase length [4, 65] as well as syntactic structure, varying from simple part-of-speech labels of target and neighboring words [104] to full syntactic parses [4]. Accent and phrase structures are interrelated, and different algorithms have used one to predict the other [36, 33]. When both accents and phrases are unknown, predicting phrases first and then using them to predict accents yields better results than the reverse order [33]. *Tone* label prediction is typically done after accent and phrase break locations are specified. Boundary tones are correlated with the intentional properties of the utterance, i.e. whether it is a statement or a question. The type of pitch accent was shown to be related to the previous accent type, position

of phrase in sentence, and break index after the accented word [78].

It has been demonstrated in the literature [104, 36, 79, 65], that phrases, accents and tones can be predicted using decision trees. Advantages of decision trees in general are that they are automatically trainable, well suited for categorical features, and that they capture dependencies between different prediction variables. These dependencies become evident from analysis of how a given tree partitions the training data. In addition to predicting the value of a given categorical variable, the decision tree can estimate the conditional probability distribution for a range of values. These probabilities can be used as costs in a WFST representation of the prosodic alternatives, as we plan to do in the proposed work. Other rule-based alternatives also exist for prosody prediction including automatic learning approaches based on TRBL [33]. We will not use these approaches primarily because we want the probability functions for costs or making "soft decisions".

Natural prosody is not unique for any given text. An evaluation of this allowable prosodic variability was reported in [79], where the authors conducted an empirical study of the differences among pitch accents and tones, comparing the different read versions of each test story. The results show that for both accents and tones the variability across versions was much larger than can be accounted for in terms of labeler inconsistency or disagreement. In the proposed work we want to take advantage of this allowable prosodic variability by incorporating it into the cost metric of our finite-state prosody model.

2.2.2 Continuous Acoustic Parameters

Symbolic prosodic markers and phonemic transcription are typically used to predict continuous acoustic parameters such as segmental duration and the F0 contour. Other acoustic correlates to prosody might include source parameters and energy contours.

Predicting durations of phonetic segments involves a number of contextual factors that include identities of surrounding phonemes, syllabic stress, word emphasis, effects

of phrase boundaries and more. These factors interact, which increases the difficulty of duration control. Nevertheless, duration is relatively well understood compared to intonation. In the particular approach to synthesis we will take, concatenative synthesis with dynamic unit selection, it has been noted that the default duration of the chosen units is acceptable without modification [26, 109, 71, 2], so we will not address this problem.

Most models of intonation represent the F0 contour at two or three time scales: phrase, accent, and optionally segment. Phrase level F0 variation is typically associated with pitch range [86, 102] and possibly baseline declination, and depends primarily on discourse [36, 19] and syntactic [4, 104] structure. Accent level F0 variation occurs at the syllable time scale. It is related to the tones and possibly also to a separate prediction of gradient syllable prominence [16]. Segmental effects (sometimes referred to as microprosody) include the influence of neighboring consonants, vowel height and placement of F0 peaks [86]. In this work, we will assume that F0 details associated with segmental effects and choice of tone labels will be captured automatically via appropriate unit selection.

Factors such as energy, spectral tilt, and other cues related to glottal source changes are less well studied and beyond the scope of this thesis.

2.3 Waveform Generation

Waveform generation is the process of transforming the symbolic representation into a speech waveform. There are three main methods used for waveform generation: articulatory [12], formant [3] and concatenative [62]. *Articulatory* synthesis involves modeling static and dynamic characteristics of the human articulators, i.e. the vocal chords, tongue, velum, jaws and lips. Generally, *formant* synthesizers use a set of rules to compute resonant frequencies of the vocal tract, or formants, which are then used to produce speech waveform. *Concatenative* synthesis, being popular for

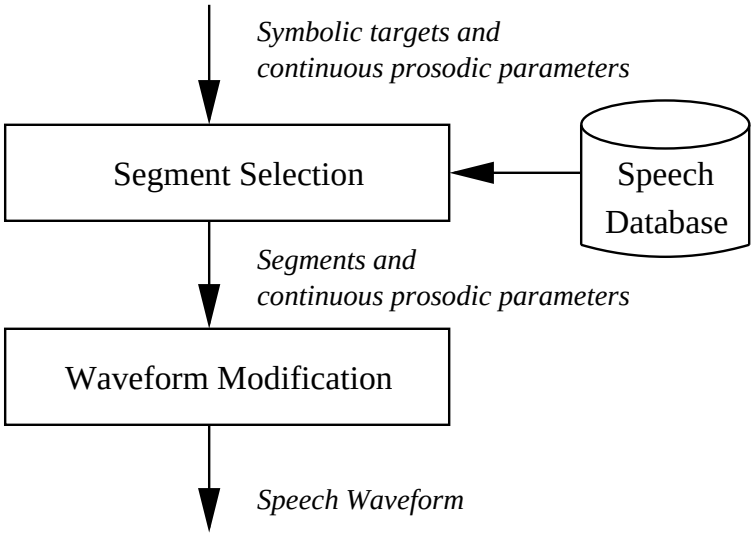

Figure 2.3: Concatenative approach to waveform generation.

its superior naturalness, particularly in constrained domain tasks, relies on extracting model parameters from speech data and concatenating fragments of speech to create new utterances. There are two main stages in concatenative synthesis: segment selection and waveform modification (Fig. 2.3). The segments can be carefully selected diphones or automatically clustered phone-sized or sub-phonetic units. Segment selection involves defining the inventory of segments as well as selecting the appropriate segment for a given phonetic (and prosodic) context. In one version of concatenative synthesis, referred to as "unit selection", the segments are selected dynamically and vary in length. Waveform modification is the process of combining the waveform pieces and modifying them to have the desired duration and pitch. The following two sections (2.4 and 2.5) will provide details about the concatenative approach to waveform generation.

The recent focus on corpus-based methods in speech synthesis has encouraged researchers to adapt techniques, such as decision tree clustering and Hidden Markov Models (HMM), that are commonly used in speech recognition. One approach to

waveform generation, described in [101], is based on the estimation of spectral parameters (such as mel-frequency cepstral coefficients, MFCC) from continuous mixture HMMs which include dynamic features, i.e. delta and delta-delta parameters of speech. In [32] the authors examine the suitability of stochastic Markov graphs instead of HMMs to improve the accuracy of the output cepstra. The cepstra can then be transformed into a speech waveform by using the mel log spectral approximation (MLSA) filter [34]. While these approaches provide a compact speech database representation (only model parameters, not actual speech waveforms need to be stored) and offer flexibility in terms of model adaptation, the quality of speech is not yet good enough for general applications. Synthetic speech of higher quality is obtained when statistical modeling techniques are applied to the concatenative approach. Decision trees have been used for unit selection [30, 11, 38, 31], while applications of HMMs include automatic speech segmentation [100, 55, 38] and smoothing waveforms at the concatenation points [70].

2.4 Unit Selection

2.4.1 Synthesis Unit

Choosing the inventory of units is a subject of ongoing research. *Diphone*-based systems have been offered for many years [51]. A diphone database contains the transitions between all pairs of phones that can exist in a given language. While English has approximately 50 phones the total number of diphones ranges between 1500 and 2000, as some combinations never occur. Diphone-based systems can produce very intelligible synthetic speech, however, in order to represent contextual diversity in synthetic speech, diphone units must undergo substantial modifications, which compromises output quality.

More recently, researchers addressed the challenge of selecting units from a large inventory of context-dependent *phone*-sized segments [42, 9, 38]. Systems such as

CHATR [9] are capable of synthesizing very natural speech, but fail to do so consistently. One of the reasons for such inconsistent performance is that phone-based systems demand phone boundary concatenations even for the cases where diphone boundaries may produce smoother joins. For example, there is evidence that stops and fricatives have minimal coarticulation effects [89] and therefore are less likely to have perceived discontinuities at joins at the phone boundaries. Vowels, on the other hand, are found to have smoother concatenations in the middle of the phone. Furthermore, different vowels have different degrees of perceived discontinuity when spliced in the middle of the phone [50].

The AT&T Next-Gen TTS system [6] overcomes this limitation by using *half-phones* for the unit inventory, which allows concatenations to be made at phone boundaries as well as mid-phones. The system allows scaling the concatenation costs at phone boundaries relative to the mid-phone. This makes it possible to tune the system's behavior, gradually changing it between phone and mid-phone concatenation. In our work we will take this capability a step further by providing controls for quantifying the potential perceptual discontinuity at a given boundary, separately from the spectral mismatch between the candidate units. This extension allows more flexibility and efficiency.

Borrowing from the speech recognition community, several researchers used context-dependent HMM state alignments to define the unit inventory. For example in [30, 40], tri-state HMMs are used to produce three units (called *senones*) for each phone in the speech database. While senones can usually model spectral variation at a fine sub-phonetic level, they introduce many junctions during concatenation which can potentially result in speech quality degradation and increased search costs.

Maintaining a phone or a subphone unit for every possible context is not practical. Representing all possibilities for just the immediate left and right phonetic context already requires more than 50,000 units for English, and this number would grow if we take into account higher level information, such as syllable structure and prosody.

Decision tree clustering techniques, widely used in speech recognition, allow for a flexible compromise between the unit database size and the acoustic variety of the units by automatically clustering units of the same phone (or HMM state) class based on their phonetic context and/or prosodic parameters (lexical stress, F0, duration) [26, 11, 38].

Regardless of the unit size, in order to be able to synthesize any text the database must have at least one instance of each unit in its inventory. For example, a diphone-based system must have all possible phone transitions. One advantage of a *single-instance* system is its compactness. Also, selecting the units during synthesis entails a simple table lookup. The disadvantage is that the instance chosen needs to be robust for all possible concatenations and various prosodic modifications. A mismatch between the unit and the target would require heavy signal processing, which in turn may lead to speech quality degradation. A *multiple-instance* system overcomes this limitation by maintaining several units of each type in its database. Having a larger number of units to choose from during synthesis makes it possible to find units that are more suitable for a given phonetic and prosodic context, hence reducing the amount of additional signal processing. The unit selection process, however, becomes more complex as it requires a dynamic search.

2.4.2 Selecting Units from Multiple Candidates

Selection of variable size units from large single-speaker speech databases [82, 42, 39, 11, 24, 38, 6, 31] is typically based on minimizing acoustic distortion introduced when selected units are modified and concatenated. This distortion is represented in terms of two cost functions (Figure 2.4): 1) *target* cost $C^t(u_i, t_i)$ which is an estimate of the difference between the database unit u_i and the target t_i, and 2) *concatenation* cost $C^c(u_{i-1}, u_i)$ which is an estimate of the quality of concatenation of two units u_{i-1} and u_i.

The target specification typically includes phonetic labels, indication of stress,

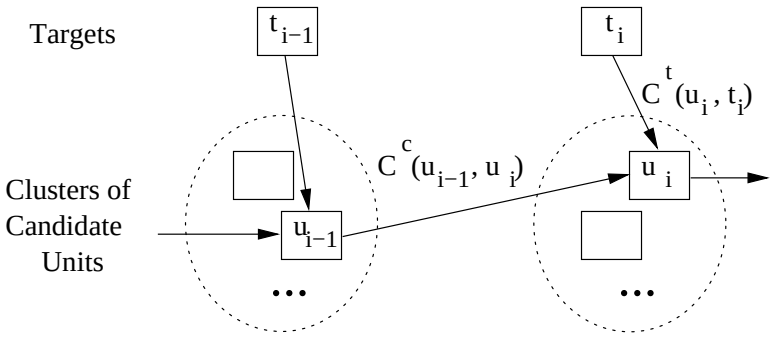

Figure 2.4: Cost functions for selecting units from multiple candidates: C^t is the target cost, and C^c is the concatenation cost.

phone duration and desired F0. Use of both symbolic and continuous features in the target makes computing the distance between a unit and the target (i.e. the target cost) non-trivial. In the case of symbolic distances, a simple binary match function is not an adequate measure and weights are typically assigned to allow several degrees of precision. These weights may be computed via an iterative training process [42], or may be linguistically motivated and set by hand [22]. Distances between the continuous parameters may be absolute difference functions (appropriate for pitch for example) or lop-sided functions (appropriate for duration, for example, in which duration shortening is penalized more than duration lengthening) [22]. When decision tree clustering is used to group similar units according to some objective function, such as minimum distance (of mel-warped cepstral coefficients, for example), then target cost may be computed by finding the distance between the unit and the cluster center [11]. An alternative objective is maximum likelihood of Gaussian models, in which case the likelihood of the unit cepstra normalized by the number of frames may be used for the target cost [40, 38].

The concatenation cost is often derived from the spectral discontinuity at the splice point. Various forms of spectral representation and associated distance functions have been studied in terms of their correlation with human perception. While there is no

consensus on the best approach, it has been demonstrated that some useful distance measures are: MFCC-based Euclidean and Mahalanobis distance [108], formant-based Kullback-Leibler distance [49, 50], and line spectral frequency (LSF) based distance [38, 106]. It has been shown that the distances can be improved by taking into account phonetic context of a given splice [28]. The absolute differences in log power and F0 have also been used in computing concatenation costs [42].

The units in the synthesis database can be treated as states in a finite state network with the state occupancy costs given by the target cost and the state transition cost given by the concatenation cost. The task of unit selection is to find database units that minimize the sum of the target and concatenation costs. When the unit database is represented in the form of a finite-state network, unit selection involves finding the optimal path through the database network, which can be done by means of a Viterbi-style (dynamic programming) search [42]. As the computational complexity of the unit selection search can grow quadratically (depending on the implementation) with the number of candidate units per target, some measures must be taken to make this approach practical. One way of reducing the computation is to make the unit database smaller by pruning units from the candidate clusters. Different pruning techniques have been investigated: some focused on eliminating the "outliers" as well as overly "common" units in terms of their target costs [11]; some concerned with diversifying the spectral characteristics at unit boundaries, which has the potential for reducing concatenation costs [38]; and others optimized the above criteria jointly [27]. Alternatively, since computing concatenation costs is the slowest operation, one can precompute and cache concatenation costs between the most frequently used pairs of units [7], or vector-quantize the space of units and store a complete distance table between groups of units [6, 22].

2.5 Waveform Modification

Once the choice of units is made, pre-recorded segments of speech are concatenated and unit waveforms are modified according to prediction of continuously varying F0 and duration parameters. At this stage speech signals are typically encoded using models that ensure that the concatenation of selected units results in a smooth transition and that the target unit pitch and duration are achieved without perceptual degradation.

There are various methods of representation and modification of speech segments. The Pitch Synchronous Overlap and Add (PSOLA) algorithm [62] is a time-domain modification method that performs pitch-synchronous analysis and synthesis of speech. PSOLA algorithm operates by extracting segments of speech that are two pitch periods long. These segments then can be replicated, scaled and added together by overlapping some parts of the neighboring segments. Even though this approach can perform high quality prosodic modifications, it does not do any spectral adjustments which are often necessary to smooth out discontinuities at unit boundaries, because it operates in the time domain.

Another popular approach uses a source-filter model, where an LPC filter is estimated from the speech unit, and the excitation is a parameterized pulse generator [91] which can be white Gaussian noise for unvoiced sounds and an impulse train for voiced sounds. Higher output quality and better resemblance with the original speaker can be achieved when the excitation is derived directly from the speech signal in the form of a residual [52, 1]. This approach, called residual-excited LPC, allows for prosodic modification of the residual and spectral smoothing across unit boundaries by manipulating the LPC coefficients or equivalent parameters such as reflection coefficients and Line Spectral Frequencies (LSF) [43]. LSF parameters, which we shall use here, roughly indicate formant location and bandwidth associated with complex poles. They are popular in speech coding mainly because quantization error

is localized to the frequency of the coefficients and results in low spectral distortion – characteristics that do not occur with other representations [41].

Sinusoidal models have also been proposed for synthesis [56]. The Harmonic Plus Noise Model (HNM) [95] is part of the family of sinusoidal models, which represents speech signals as having a time-varying harmonic component plus a modulated noise component. The decomposition of the signal into these two components allows for more natural-sounding modifications of the signal and provides a straightforward way of smoothing discontinuities of acoustic units around concatenation points by linear interpolation of the model parameters.

PSOLA, residual-excited LPC and HNM are very popular waveform modification methods and have been used successfully in concatenative speech synthesis. However, these algorithms are still limited in their ability to make significant prosodic modifications, which raises the importance of selecting units that match the target well.

2.6 Integrating Prosody in Unit Selection

One unsolved problem in the concatenative approach is how to incorporate prosodic factors in unit selection to reduce the signal modification requirements. The major disadvantage of predicting the F0 contour and phone durations for the target specification is that prediction methods are prone to errors, often making target specification inaccurate. Moreover, such specification does not take into account allowable prosodic variability of natural speech.

A compromise between the naturalness of unmodified speech and the desirability of suitable prosody was sought by P. Taylor [98], where he suggested using a weighted combination of the unit's original prosody and prosody specified in the target, so that the degree of required prosody modification would take into account the closeness of phonological match, the known accuracy of the target specification and the amount of

distortion that the signal processing will produce. In another compromise approach, an objective criterion for determining which units require prosody modification was implemented for Japanese language in CHATR [25], where the authors measured the F0 slope in accented regions of candidate units, and performed pitch modification when the units' F0 contour had a slope outside the acceptable range. This range was set manually as a function of the accent position in the phrase and the phrase position in sentence.

Some researchers attempted to make prosody specification in the target more general by representing it in the form of tree-like phonological structures [14, 98]. It has been suggested that ToBI prosodic structures may be a reasonable representation [20], but no results were reported. ToBI structures are appealing from the computational modeling prospective, since the ToBI label hierarchy is fixed and may not grow to arbitrarily long depth. In constrained domain synthesis, ToBI structures are simple enough to extract long (i.e. word or phrase sized) continuous segments from the database that match the target. On the other hand, when the domain is not restricted, the chance of finding a large generic phonological structure to match the target is small, resulting in poor phrase-level matches or short-time segments, while we may still find a suitable ToBI structure to represent the target.

The drawback of these approaches is that the "compromise" is determined *after* the prosodic target has been specified. We propose a more dynamic solution, where the target prosody contains a range of values specified in terms of a weighted distribution, and the best matching units within the range are found. We use symbolic markers to specify prosodic targets, while the continuous F0 parameters are used in a distance function for clustering and for computing concatenation costs (as described in Chapter 5).

2.7 Tying Language Generation and Speech Synthesis in Dialog Systems

The simplistic view of response generation treats natural language generation and speech synthesis separately. Decisions about text planning and sentence realization are made ignoring the capabilities of speech synthesis. Decisions about prosodic structure in speech synthesis are based on simplistic (error prone) text analysis, ignoring any syntactic information that might be available from the process of language generation. Clearly this strict separation is inefficient, and researchers have long considered the idea of more tight integration of these modules [112], often referred to as concept-to-speech (CTS) synthesis. In recent years, the increasing feasibility of human-computer dialog systems due to improved ASR performance has prompted the need for better response generation. As a result, there is growing research activity in the general area of CTS, particularly for limited domain applications.

An important motivation for integrating language generation and speech synthesis is improved prosody prediction. The basic idea is to augment the text output from a generator with additional information that is a by-product of the generation process. The key research problem is to determine what types of information – including semantic, syntactic and/or discourse information – that actually leads to improved synthesized speech quality. Several studies have been conducted [73, 94, 66, 67, 37], though there is still much to be learned about what linguistic factors are most important for prosody prediction. An important aspect of these studies is that they leverage generators with sophisticated linguistic internal representations. However, few such generators are used in speech dialog systems; most are hand-crafted template-based generators.

Template-based language generation [85, 68] involves mapping a semantic frame in a given dialog act directly into a template utterance assigned to this dialog act and then filling in required values from the semantic frame. The sequence of words

in the template is predetermined by the system developer. This approach has the advantage of implementational simplicity, greater control for the developer, and predictable output quality. It has the disadvantage that there is little reuse of tools from one application to another, and there is little (if any) explicit representation of linguistic structure that might be useful for prosody prediction.

Seneff and Polifroni [85] overcome the limitation of communicating with prosody prediction by "tuning" the generator for improved synthesis performance. Instead of using plain text, the generator communicates with the synthesizer via a special mark-up language that can include additional information, such as prosodic characteristics of the template, and synthesis "shortcuts" which allow the developer to bypass the search when desired and explicitly represent waveform segments. Although this approach requires greater effort from the system developer, it has a potential for producing higher quality speech output.

Improvements in speech synthesis for response generation have mainly involved designing unit-selection concatenative systems that are tailored to a given domain, taking advantage of the fact that a specific application is limited in scope. It has been a common practice in developing limited-domain synthesis systems to record a large number of domain-specific prompts. High quality synthetic speech is achieved by concatenating whole phrases and words with sub-word units for infrequent or new words [110, 29, 10]. This requires having a speech corpus that sufficiently covers the generator's vocabulary and provides recordings of frequently used words and phrases in the appropriate prosodic context.

2.8 WFSTs for Speech Processing

Many areas of language and speech processing have adopted the weighted finite-state transducer (WFST) formalism [46, 77, 89, 59], because it supports a complete representation of regular relations and provides efficient mechanisms for performing

various operations on them. A finite-state transducer encodes a regular relation, i.e. a set of pairs of strings. The elements in each pair correspond to the transducer's input and output strings respectively. Every pair in the relation is represented in the transducer by a path, a sequence of arcs from the start state to a final state. In a weighted finite-state transducer (WFST) arcs have weights (costs) assigned to them. Given these weights, the application of the WFST entails finding the best path (i.e. path with the least cost) through the network. This task involves dynamic programming which is typically done by applying the Viterbi algorithm.

A simple example of a WFST W_d is given in Figure 2.5. It translates a sequence of zero or more "red" followed by one "green" into "GO". In the form of a regular relation W_d can be defined as:

$$W_d \models (red^*) \cdot green \rightarrow GO,$$

where "$*$" denotes *Kleene closure*, i.e. a sequence of zero or more elements. Here the input alphabet consists of words "red" and "green" and the output alphabet contains only "GO". W_d is capable of transducing all sequences in the form

$$\{(green), (red - green), (red - red - green), \ldots (red - \ldots - red - green), \ldots\}.$$

Note that "red" is being mapped into ε which corresponds to an empty string. Both arcs and the final state (denoted with a double circle) have costs associated with them (after "/"). In general, transducers have one start state, but can have more than one final state. Moreover, a final state may coincide with the start state.

It is easy to see that a given regular relation can be represented by more than one WFST. For example, transducers in Figures 2.5 and 2.6 encode the same regular relation. However, the WFST W_n in Figure 2.6 has more states and arcs and therefore is not minimal. The operation of reducing the number of states and arcs in a WFST without changing its functionality is called *minimization*. Minimizing a WFST makes it more compact and easier to traverse.

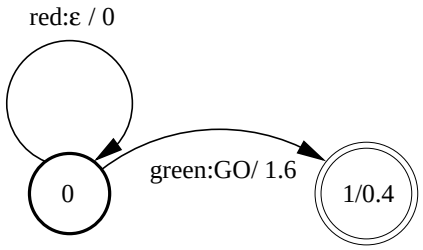

Figure 2.5: An example of a weighted finite-state transducer W_d.

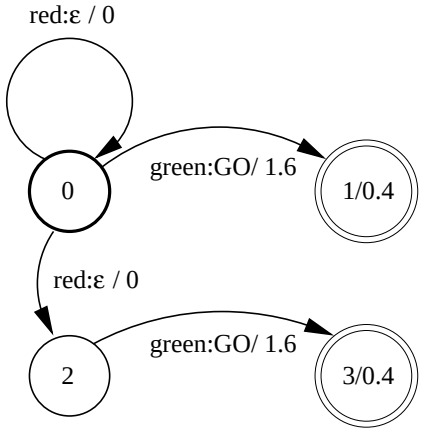

Figure 2.6: An example of a non-minimal and non-deterministic WFST W_n.

In addition to being non-minimal W_n is also *non-deterministic*, because there are two arcs with the same input label "red" leaving state 0. Transducers are called *deterministic* if at each state all outgoing arcs carry unique input labels (such as in Figure 2.5). Deterministic transducers are more efficient to perform operations on because there is no ambiguity at any step during graph traversal. Algorithms for automatic determinization of certain types of WFSTs have been developed [57, 58], but in general not all WFSTs are determinizable.

Transducers can be cascaded by means of composition and/or their functionalities can be combined by the union operation. If a transducer T_1 has input alphabet A and output alphabet B, while another transducer T_2 uses B and C for its input and output

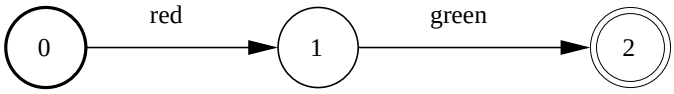

Figure 2.7: Input sequence $red - green$ represented as an FSA I.

alphabets respectively, then the output of T_1 can be sent as input to T_2. Instead of having a sequential application of first T_1 and then T_2 one can combine these two transducers by means of *composition*, producing a single transducer $\hat{T} = T_1 \circ T_2$ with A and C as its input and output alphabets respectively. By eliminating the intermediate step, $\hat{T}$ is more efficient than the sequential application of T_1 and T_2.

When two transducers T_1 and T_2 have the same input alphabet A and the same output alphabet B, their functionalities can be combined with the *union* operation. The resulting transducer $\hat{T}$ will have capabilities of both T_1 and T_2, and yet will be faster than trying to apply T_1 and T_2 separately and then comparing the results.

The application of WFSTs to a given input can be performed with the composition if the input is also represented as a WFST. For example, if we want to apply W_d to the input sequence $red - green$ we can represent this input sequence as a finite-state acceptor (FSA) I (shown in Figure 2.7), and the transduction is then accomplished via the composition $I \circ W_d$. Unlike WFSTs that encode regular relations, FSAs have only one label assigned to each arc and hence they encode regular expressions. The arcs and exit states may also have weights in which case we have weighted finite-state acceptors or WFSAs. For the purposes of composition $I \circ W_d$, which requires both arguments to be transducers, acceptor I can be treated as a WFST whose input and output labels are identical.

The result of the composition is also a WFST shown in Figure 2.8. However, it may be that only the sequence of output labels is of interest to us, in which case we take the *right projection* of the result $\pi_2(I \circ W_d)$, obtaining an acceptor that represents the sequence $\varepsilon \cdot GO$. The ε-transitions that correspond to empty strings

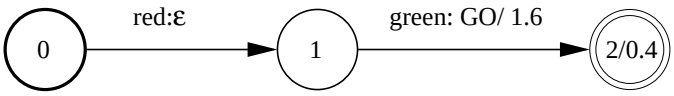

Figure 2.8: The result of the composition $I \circ W_d$.

can be removed with ρ_ε operation yielding the final output, which is simply GO. In summary, the application of WFST W_d to input sequence I is accomplished with the following operation:

$$\rho_\varepsilon[\pi_2(I \circ W_d)].$$

While determinization and minimization are not applicable to some WFSTs, all FSAs can be determinized and minimized for efficiency.

Composition is non-commutative (i.e. $I \circ W_d \neq W_d \circ I$), and transitive (i.e. $(I \circ W_1) \circ W_2 = I \circ (W_1 \circ W_2)$). The order in which pairs of transducers are composed is a design issue. When W_1 and W_2 are predefined transducers, then their composition $W = W_1 \circ W_2$ can be computed in advance hence making the input transduction $I \circ W$ faster. In some cases, however, application of W_1 and W_2 to the input I in a sequence is more appropriate. In particular, if W_1 is designed to account for a large number of possible input sequences and W_2 expands the output of W_1 with even more new options, then composing W_1 and W_2 may produce a very large transducer. But if I is first composed with W_1 then the number of options is restricted to those defined by I. This method of splitting the functionality among several transducers is often referred to as *factoring*.

Recent applications of WFSTs range from the construction of lexical analyzers [45] and the compilation of morphological and phonological rules [44, 47], to text and speech processing [60] and speech recognition [59]. In speech synthesis, WFSTs were used for text analysis [76, 89, 90], prosodic phrasing [89], and unit selection [42, 111, 17]. The prosodic phrase prediction model described in [89] is based on decision trees. The author uses the algorithm reported in [92] to compile the decision

tree into a WFST, which then can be applied to the overall WFST-based text analysis module.

The unit selection database can be efficiently represented in the form of a WFST, as was suggested by [42] and implemented in [111, 17]. The basic idea is to have units represented in the transducer by states, and the arcs carry out the transduction from input target labels into unit indices with weights derived from the combination of target and concatenation costs. Yi and Glass [111] use phones as the fundamental synthesis unit. In order to constrain the number of links in the WFST, they introduce a series of intermediate layers of states, where all possible unit transitions are mapped into more general classes based on phonetic categories, and transition costs between each pair of classes are computed. The system also incorporates word-to-phoneme conversion in a WFST module. This module is then composed with the unit selection WFST, allowing conversion of input words directly into a sequence of database units.

Chapter 3

CORPORA

This work makes use of two corpora: *Communicator* and *Radio News*. The choice of corpora was dictated by the types of applications studied in this work: limited domain and unrestricted text-to-speech. Limited domain synthesis systems have a relatively small vocabulary and can take advantage of speech recordings containing frequently used words and phrases. Unrestricted text-to-speech systems must be able to synthesize any word sequence and therefore they rely on corpora with a wide range of phonetic and prosodic structures.

3.1 Communicator

The Communicator corpus, which we used for our work in limited domain speech generation, contains recordings of system responses from a travel planning dialog system developed at the University of Colorado [68]. The domain represented by the corpus is fairly limited. The system is capable of synthesizing 120 different types of responses represented by templates such as the following:

> Would you like a rental car in CITY_DEST?
> Is CITY_DEST your final destination?
> I have you going to CITY_DEST on DEPARTURE_DATE.

The dynamic information that is inserted into these templates includes names of cities, airports, airlines, hotels, car rental companies, dates, time of the day, flight numbers and costs. The variety of the dynamic information (especially numerical data) makes

the number of possible output word sequences astronomical (order of 10^{25}). Clearly, such a system cannot use prerecorded prompts.

All of dynamic information is represented with approximately two thousand *units*, i.e. words and proper names. Several types of these units, in particular, city names and airports, are recorded in more than one prosodic contexts to reflect their use in the templates. Nevertheless the corpus oversimplifies prosodic variability encoded in these templates. For example, city and airport names can occur in 45 various prompts. Locations of city and airport names in these prompts can be grouped according to the prosodic context as follows:

1. last in a statement (e.g. "I have you going to CITY.");
2. last in a yes-no question (e.g. "Would you like a rental car in CITY?");
3. last in a question other than yes-no (e.g. "What day do you want to leave CITY?");
4. middle of sentence, not phrase final (e.g. "Leaving CITY on Friday.");
5. middle of sentence, appears in a list of items (e.g. "I know of 3 cities in Arizona: CITY1, ...");
6. middle of sentence, major phrase final (e.g. "I have you going to CITY, is this correct?").

Using the carrier phrases above, it requires 25 seconds to record a city in all 6 prosodic contexts. For a list of 850 city and airport names we will need almost 6 hours of speech. When we consider various prosodic contexts for other units (i.e. numbers, dates, etc.) the amount of recordings will exceed 14 hours. With a larger number of units and more degrees of freedom in prosody (for example, if we consider various types of pitch accents) and/or if we try to account for various phonetic contexts, the amount of recordings becomes prohibitive.

Dialog systems for the travel planning domain have been developed at several sites [68, 81, 84, 93, 72, 115, 113, 114]. While approaches to designing synthesis components in these systems varied from a general purpose text-to-speech engine [6] to a limited domain speech synthesizer [10], a formal evaluation [103] showed that systems taking the limited domain approach to synthesis were rated as most natural by the users. The baseline for our synthesizer is the University of Colorado (CU) system [68] which was rated among the best based on the quality of speech output. CU and other sites [80, 110] used similar strategy when collecting speech corpora for their limited domain synthesizers, i.e. record prompts, proper names, dates, numbers and other words used by the system. However, prosodic characteristics, such as boundary tones and pitch accents, were ignored or oversimplified.

The size of the corpora in terms of their length, number of words, number of phones and size in megabytes (uncompressed) is summarized in Table 3.1. The Communicator corpus contains approximately 2 hours of speech from a female speaker who did not have any special training in diction. The entire corpus was automatically segmented with subsequent hand-correction applied to the word boundaries. A trained linguist annotated a subset of the corpus (17 minutes) with ToBI prosodic labels [87]. This prosodically labeled subset of the data (2752 words) was randomly split into training and test sets: three quarters of the data was used for training and the rest was used for testing.

The Communicator corpus was digitized at 16kHz sampling rate in 16-bit format. F0 contours and epochs (i.e. instances of the glottal closure) were automatically extracted for all waveforms by means of the Entropic signal processing tools [96, 97]. All of the transcriptions were automatically labeled with part-of-speech tags [74] and syntactic structure [21]. In concept-to-speech generation, you would not typically need a parser or a tagger since the generator would provide that information, but the generator used in this implementation of the Communicator system was not designed with such an interface in mind. Information available from the generator included

Table 3.1: The speech corpora.

	Communicator corpus			Radio News corpus		
	Training		Test	Training *"radio"*		Test
	Total	Prosodically labeled		Total	Hand-corrected segmentation	*"labnews"*
Length	2 hours	13 min	4 min	1 hour	35 min	12 min
Words	16239	2064	688	9091	5193	2112
Phones	66807	8509	2704	45560	25488	10091
Size (MB)	230	25	8	115	67	24

semantic classes (city, number).

3.2 Radio News

The Radio News corpus [64] is comprised of local news stories spoken by a professional radio announcer. It offers a wide range of phonetic and prosodic structures and has been used in other studies for unrestricted text-to-speech synthesis [42, 11], though it was not explicitly designed for this application. In particular, noise level and other channel effects changing between recording sessions result in non-uniform quality of synthesized speech.

The Radio News corpus consists of a one-hour training set (called "radio") and a twelve-minute test set (called "labnews") spoken by a female professional radio news announcer. The training set was recorded at the radio studio during broadcast, while the test set was collected in a laboratory. The entire corpus was automatically segmented, and the phonetic segmentations were corrected by hand in a part of the training set (35 minutes) which we used in our synthesis experiments, and in the

entire test set. The corpus was also annotated with ToBI prosodic labels.

As for the Communicator corpus, the Radio News recordings were digitized at 16kHz sampling rate in 16-bit format and used to automatically extract F0 contours and epochs. The corpus was also automatically labeled with part-of-speech tags and syntactic structure.

3.3 Prosodic Labeling

The main types of prosodic events are pitch accents and phrase breaks with associated tones (see Section 2.2.1). The ToBI (Tones and Break Indices) transcription system [87] was developed for annotating these events with a set of symbolic markers. ToBI phrase structures are hierarchical, hence a major prosodic phrase can contain more than one minor phrase. Pitch accent labels describe various shapes of the pitch contour. Pitch accents with high F0 excursion are typically labeled with high (H*, L+H*) or downstepped (!H*, L+!H*, H+!H*) markers, where downstep indicates that the accent has F0 lower than that of the preceding accent in the same prosodic phrase. Accents with low F0 excursion are usually marked with low tones (L*, L*+H).

The reliability studies [87] of the ToBI system have shown a significant disagreement (36%) among human labelers when assigning specific labels to pitch accents, while only 14% disagreed on accent location. Marking accents with downstep labels appeared to be somewhat more reliable than all tones, resulting in 21% disagreement. In our work, to alleviate the data sparsity and to lessen the effects of labeling inconsistency we group all 7 types of pitch accents in both corpora into three categories: low accents, and two types of high pitch accents: downstepped and non-downstepped.

Table 3.2 shows relative frequency of prosodic labels in the corpora. While the overall frequency of accented words and words with boundary tones is similar in both corpora, there are differences among specific types of labels. Low accents have much more significance in the Communicator corpus than in the Radio News corpus. This

Table 3.2: Relative frequency of various prosodic labels in the Communicator (C) and Radio News (R) corpora. DS stands for downstepped, bd. stands for boundary.

	Accented	Among all accents			Words with	Among all boundary tones			
	words	High	Low	DS	bd. tones	L-L%	L-H%	H-L%	H-H%
C	52%	49%	22%	29%	21%	71%	16%	3%	10%
R	49%	75%	$< 1\%$	25%	23%	65%	33%	2%	$< 1\%$

may be attributed to the differences in genres as well as in individual characteristics of the speakers. We observed more H-H% tones in the Communicator corpus because yes-no questions are common for a dialog system while hardly occur in a news broadcast. A higher rate of L-H% tones (and lower rate of L-L% tones) in the Radio News corpus can be the result of using longer sentences composed of many major prosodic phrases.

While there is a strong linguistic motivation for using ToBI labels to represent prosody symbolically, annotating corpora with ToBI is a very expensive and time consuming process which requires hand labeling to be performed by a trained linguist. The current focus in speech synthesis research is on automatic, corpus-based methods, and our long-term goal is an automatically trainable system requiring little or no hand labeling. Use of ToBI labels in this work provides us with a baseline to assess the usefulness of intermediate symbolic labels in the unit selection process.

Chapter 4

INTEGRATED RESPONSE GENERATION

4.1 Response Generation Components

Our response generation system is part of a mixed-initiative human-computer dialog system that is designed to help people solve travel planning problems, where users can get information about flights, hotels and rental cars. The complete dialog system is based on the University of Colorado (CU) Communicator system [68], with changes only to the response generation components. The CU system uses a client-server architecture, developed in the DARPA Communicator program with the specific goals of promoting resource sharing and plug-and-play capability across multiple sites [83]. Under this architecture the system is composed of a number of servers that interact with each other through the Hub. The behavior of the Hub is controlled by a simple script. In this chapter we will focus on the aspects involving communication between the language generation and speech synthesis servers.

4.1.1 Overall Architecture

There may be several modules involved in response generation, from dialog management and planning to waveform generation. However, because our system currently makes only very limited use of user preferences, the planning process is relatively simplistic and is handled by the dialog manager. Thus, there are three main modules involved in this system: sentence realization, prosody prediction and waveform generation. (These are implemented as two servers in the current system, with prosody prediction and waveform generation combined, but this is a choice of convenience

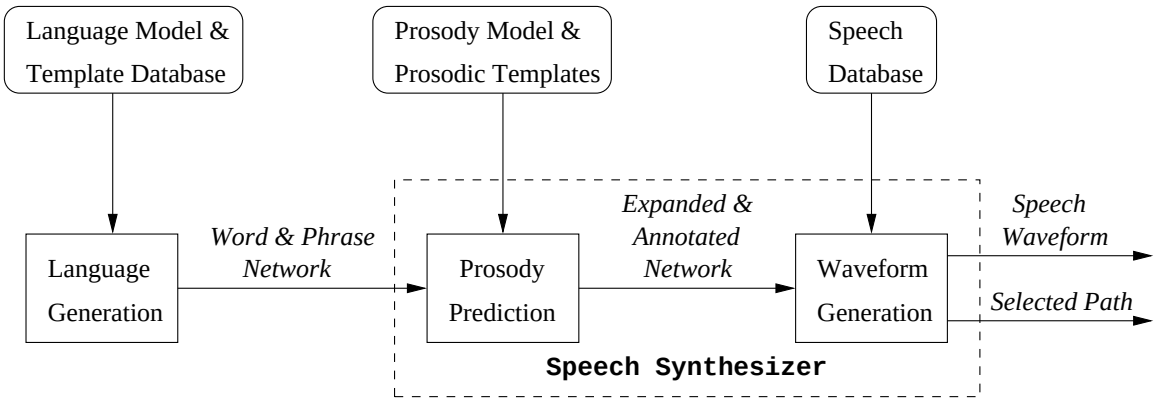

Figure 4.1: Communication strategy between the language generator and speech synthesizer.

rather than necessity.) As illustrated in Fig. 4.1, the modules communicate by passing annotated graphs or lattices that get progressively expanded until the final search process determines the best path through the network. Since the choice of response may have an effect on the future dialog strategy, the synthesizer needs to inform the dialog manager and possibly other modules about the final output. (For example, the knowledge of the specific words in the generated response may be useful for detecting future error corrections by the user, as observed in [48].) Thus, the final outputs are the synthesized waveform for playback to the user and the best path through the network.

The lattices passed from the generator to the synthesizer contain arcs that are annotated with an index and optionally attributes, as well as weights associated with a cost for that particular arc, if any. At each stage the lattice may be expanded and/or further annotated. The representation in terms of a weighted network fits nicely into the WFST framework, as will be described further in Section 4.2. Each module has access to a database for interpreting and further annotating the arc indices and attributes, and each adds new weights in any further expansion of the network.

The flexibility of the system is improved by having the generator and prosody

prediction module output more than one candidate response. This gives more freedom to the synthesizer for selecting an utterance that will sound most natural. Generation of multiple candidate responses also frees the developer from the burden of redesigning the synthesizer database every time there is a minor modification to the generator, since the dynamic search over many alternatives makes it possible to automatically optimize the system response given a sufficiently rich speech database and inventory of templates. Moreover, it achieves a tight integration without sacrificing overall modularity of the system since the generator and the synthesizer are not internally coupled.

4.1.2 Language Generation

The architecture described above can support any generation mechanism that can provide weighted alternatives, including the stochastic and hybrid approaches. In previous work an n-gram language model trained on transcribed human-human speech has been used to generate candidate responses that are subsequently filtered to eliminate grammatical errors [63], and alternatively to score candidate responses for "quality" [53, 35]. Ratnaparkhi [75] proposed a statistical learning approach to produce natural language text directly from a semantic representation that could lead to a more useful means of scoring candidate sentences. Work by [5] offers the potential to introduce scores at the sentence planning level. These developments are important for the work reported here, because they involve generation of multiple responses associated with a score, which our work can take advantage of. However, because many current dialog systems are based on template generators, we chose to initially work with the template paradigm where wording alternatives are unweighted.

Here, we build on a system that uses a template generator, which has a set of predetermined responses for different dialog states. Depending on the response, there are empty slots that are filled in with values from the semantic frame provided by the understanding module. Example responses include:

Will you return to CITY-ORIG from CITY-DEST?

I have you going from CITY-ORIG to CITY-DEST. Is that correct?

Do you have an airline preference?

What time do you want to leave CITY-ORIG?

As an aside, we note that while the template generator is relatively limited and most current work seeks to go beyond this framework, it does have the advantage of simplifying the implementation of multi-lingual systems.

For the template generator, the lattices passed from the generator to the synthesizer contain arcs that correspond to words, phrases or templates, so the annotation on an arc could be a specific word, a phrase index, or a template index with attached words or values to fill the associated slots. A message from the generator may also include some dialog state information that may be relevant for prosody prediction.

The baseline generator provides a single text string to the synthesizer. Our modifications involved introducing multiple alternatives for a subset of the prompts and changing the interface from a text string to weighted, indexed lattices. An example of the alternatives for one prompt is illustrated in Fig. 4.2. We modified 7 prompts, and the resulting networks characterized 2.7 sentence alternatives on average.

The framework provides for the possibility of weighting the different text alternatives. In a template generator, which is hand-crafted, it is not clear how to specify the weights, so we simply gave all alternatives equal (zero) costs. With a stochastic generator, one could use grammar probabilities or n-gram scores to determine weights.

4.1.3 Prosody Expansion

In addition to generating multiple candidate templates our system supports alternative prosodic targets for each word string. While it is certainly not the case that any prosodic realization will be acceptable, there is some degree of variability that is perceived as natural and appropriate for the context. Anecdotally, it is easy to notice

Will you	return	to	CITY_ORIG	from	CITY_DEST?
Will you	return	from	CITY_DEST	to	CITY_ORIG?
Would you like to	return	to	CITY_ORIG	from	CITY_DEST?
Would you like to	return	from	CITY_DEST	to	CITY_ORIG?
Do you want to	return	to	CITY_ORIG	from	CITY_DEST?
Do you want to	return	from	CITY_DEST	to	CITY_ORIG?

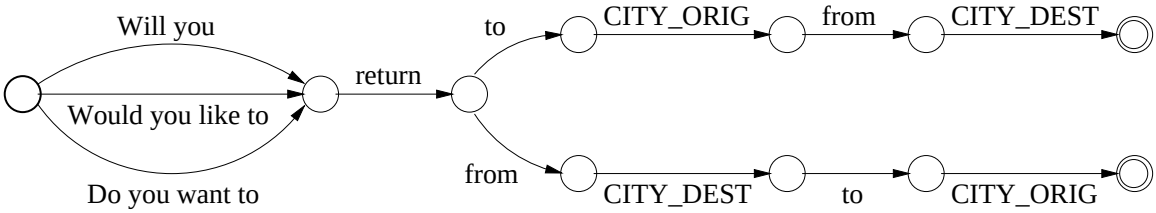

Figure 4.2: An example of a word and phrase lattice representing six alternative wordings of a prompt.

differences in story readings or dramatic performances. More formal evidence is provided in [79], where it was found that multiple readings of the same text (news stories) resulted in more than one symbolic representation for the prosodic structure of most utterances. In our prosodically labeled synthesis database, we find the same phenomenon, but in this case for a single speaker. For example, the utterance "Will you return from Boston to Seattle?" had two prosodic realizations, illustrated in Figure 4.3. This variety does not appear to be correlated with location in the dialog because most utterances were not read in the context of a complete dialog, so presumably the different prosodic renditions are used by the speaker for variety. Style-related variety is characterized in a non-deterministic algorithm for prosody prediction proposed by [19]. Here, we use a statistical model to weight alternatives, but the joint search means that the overall algorithm is deterministic.

As we are building a constrained domain synthesizer, we can expect much of the input to resemble the utterances recorded when collecting data. Therefore, we

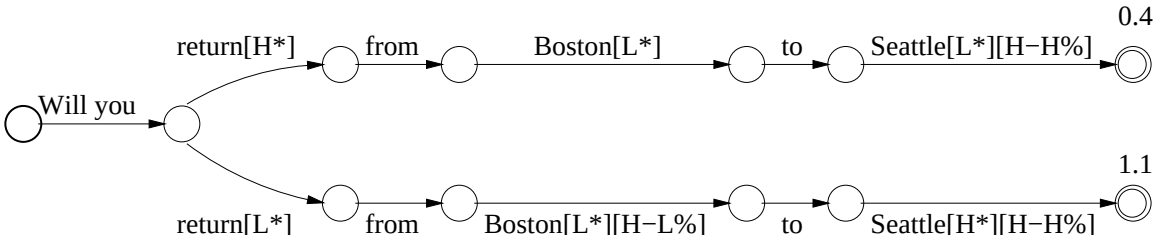

Figure 4.3: Representing prosodic variability in the form of a network of prosodic templates. In brackets are ToBI labels for accent and boundary tones.

want our model to precisely describe prosodic structures represented frequently in the training data in terms of so-called prosodic "templates". The "templates" are not stored speech waveforms, but rather a symbolic sequence of words and word "slots" that are associated with symbolic prosodic labels describing pitch accents and phrase boundary markers. Thus, the templates may or may not correspond to actual phrases in the synthesis database, depending on the particular words chosen. The candidate prosody templates are weighted according to their relative frequency in the training data, using negative log probability as the "cost" of choosing a particular template.[1]

It is conceivable that none of the prosodic templates has a precise match in the unit database, either because of the particular choice of words for filling a slot or because there has been some change to the generator. In this situation, prosody is predicted for each word using a generic prosody prediction module. The new predictor is effectively used as a back-off mechanism, though it differs somewhat from the "back-off prosody prediction" suggested by [98] in that the implementation is integrated (because of the WFST implementation) rather than a two stage process.

The "generic" prosody prediction component uses decision trees, following the method proposed in [104, 36] and used in many studies since then. The features

[1]We observed that longer city names tend to have major phrase breaks assigned to them more often. Potentially, it may be appropriate to include the length of the city name as a factor when computing the costs of prosodic sequences, however, more training data is necessary to make this statement statistically conclusive.

predicted include major prosodic phrase breaks and associated boundary tones and three types of pitch accents – the same labels as used in the prosody templates. The features used in prosody prediction were extracted at the word level, i.e. for each word in the corpus we produced one vector of features describing the word's part-of-speech, a function word indicator, and features describing the word's position in the syntactic tree[2] and within the prosodic phrase. Given that our corpus is characteristic of the travel planning domain, we also used two features from the generator indicating if the word is a city name or a number. The predictor was trained and evaluated on prosodically labeled synthesis data, as described in Section 4.3. We also designed predictors on a standard Radio News corpus [64], confirming that the accuracy was similar to previously reported results [79]. In our application, unlike most prosody prediction work for speech synthesis, the predicted values are not used directly. At the leaf of the decision tree is a distribution of possible alternatives for a given context. All cases with non-zero probability are allowed as possible arcs, weighted according to their negative log probability.

4.1.4 Waveform Generation

The waveform generation module is a constrained-domain concatenative synthesis system that uses a unit selection framework to incorporate variable size units, from single words to phrases. The module takes as input the prosodically annotated network, expands it into target phoneme sequences (annotated with lexical stress and prosodic markers), and then searches for the lowest cost unit sequence from a synthesis database that corresponds to a path in the network. The search linearly combines the standard target and concatenation costs used in unit selection with the other costs included in the network from previous modules.

[2]In this work we obtained syntactic features automatically from text using a parser, as described in Chapter 3, but conceivably these may be supplied by the language generator with greater accuracy.

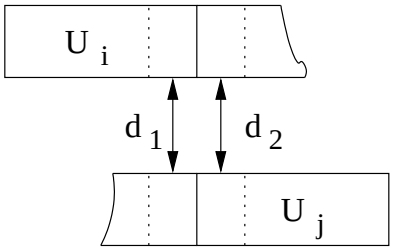

Figure 4.4: Frames of spectral features used in computing the concatenation cost between units U_i and U_j.

In the system implemented for these experiments, the minimal size units are words, so target costs are zero for a matching word and infinite otherwise. The concatenation cost between units U_i and U_j is the maximum distance[3] between overlapping frames of features: $\max(d_1, d_2)$, where d_1 is the distance between the last frame in unit U_i and the frame in unit U_{j-1} which precedes unit U_j (as the database was naturally recorded), and, similarly, d_2 is computed between the first frame in unit U_j and the first frame in unit U_{i+1} which follows unit U_i, as illustrated in Fig. 4.4. This approach is more robust than computing a distance between two consecutive frames, because it does not imply continuity at join points. Motivated by work described in [107], we used a line spectral frequencies (LSF) representation of frames at unit boundaries. Squared differences between individual LSF vector components were weighted by the inverse of the distance between the adjacent spectral lines. We also included F0 and energy for computing the concatenation costs, using the same max framework and normalizing the contribution of these terms so that the average was similar to the LSF average.

Because this is a constrained-domain system, it is possible to get relatively good quality speech without any waveform modification, and the implementation does not include it.

---

[3]The idea of using the maximum (vs. the average) was borrowed from [22], but the specific distances used here differ somewhat.

4.2 Integrating Generation and Synthesis with WFSTs

A WFST architecture provides a framework for a flexible and efficient implementation of the response generation components. The flexibility of WFSTs accommodates the use of variable size units and different forms of prosody and text generation. The computational efficiency of WFST composition and finding the best path allows real-time synthesis, particularly for constrained domain applications. This section gives details about our approach to integrating language generation and speech synthesis under a common framework of WFSTs.

It should be clear that a word network can be represented as a finite-state machine, which is a particular class of finite-state transducers. The process of building prosody prediction and unit selection WFSTs used for speech synthesis will be described in Sections 4.2.1 and 4.2.2, respectively. Then Section 4.2.3 will explain how the steps of template generation, prosody prediction and unit selection are tightly coupled by composing the corresponding WFSTs.

4.2.1 Prosody Prediction

The overall modular structure of our finite-state prosody model is summarized in Fig. 4.5, including two types of WFST models: *template prosody* which describes specific prosodic structures in the training data, and *general prosody* prediction which predicts prosody for unseen cases. Both prosody prediction models are generated at two levels: utterance and phrase. At each level the prosody prediction WFST may itself be produced by composing individual transducers, as in the case of accent and tone prediction at the phrase level. Then, the resulting models at each level (i.e. utterance level and phrase level) are composed to form the overall prosody prediction WFST. The order of terms in the composition corresponds to the order of steps during prosody prediction, i.e. utterance-level prosody is generated first and then used as a predictor for the phrase-level prosody. Finally, the template and general prosody

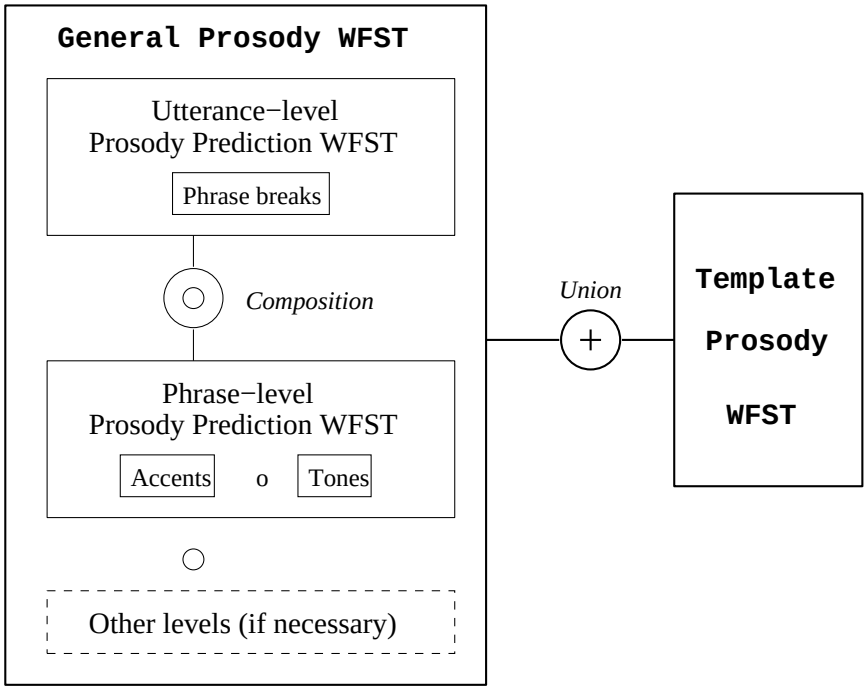

Figure 4.5: Modular structure of our prosody model, where $\oplus$ indicates union and $\circ$ indicates composition operations on WFSTs.

prediction WFSTs can be combined into a single transducer by means of the union operation. The modular structure allows other levels of prosodic structures (such as word or paragraph levels) to be easily added if desired.

The general prosody prediction module is based on decision trees, which can be efficiently compiled into weighted finite-state transducers. A simple decision tree can be represented by a WFST with just two states (a start and an end state) and the number of arcs equal to the number of leaves in the tree times the number of different values that the output variable can take (as illustrated in Fig. 4.6). The costs in the resulting WFST should reflect the conditional probability distributions that can be computed at each leaf when training the tree. As suggested in [92, 89], we used $c(x) = -\log(p(x|leaf))$ as the cost function in our experiments.

For cases where there is more than one prosody "template", no prosodic target

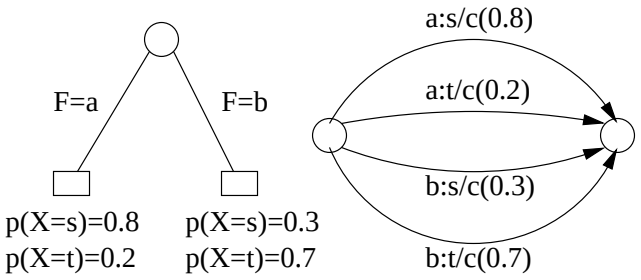

Figure 4.6: A simple decision tree and its WFST representation, where F is a prediction variable and $\{s, t\}$ are the possible class labels.

will be given zero cost. In this case, it may happen that the template with the higher prosody cost has zero concatenation cost (recorded in the database), while the template with the lower prosody cost has some non-zero concatenation cost since the template is based on "slots" and not actual words. In most cases, we want the total cost to favor the version that is a complete phrase in the database, hence template prosody costs are scaled so that the concatenation costs dominate the total cost in such situations. In addition, the costs in the template prosody transducer are scaled so that they are (on average) lower than the cost of the general decision tree-based prosody prediction, since the templates (when applicable) can presumably model prosody with greater accuracy than the decision tree. Costs for likely prosodic sequences from the general model were scaled so that on average they were close to the average concatenation cost.

4.2.2 Unit Selection

The units in the synthesis database can be treated as states in the finite state transducer with the state transition cost given by the concatenation cost. The units can be of arbitrary size. It is, however, important to match the unit inventory to the output of the prosody prediction module in order to satisfy necessary conditions for composing the prosody prediction and the unit selection WFSTs, i.e. the set of output

strings from the prosody prediction WFST must be acceptable as input to the unit selection transducer. In the case of limited domain synthesis, many of the responses that a text generator produces are likely to contain phrases that were recorded during data collection. Both words and phrases are indexed directly and treated as units in the database in the implementation here. For instance in Fig. 4.7, a phrase *"will you return"* is mapped into a single arc, and hence as a whole would have an instance in the unit database. Sub-word elements of these words and phrases can also be used as units when it is necessary to be able to synthesize new words.

4.2.3 WFST Composition

WFSTs provide a convenient representation of the different types of targets used in response generation: text, prosodic structure, and speech units. As explained earlier, the language generator outputs a network of templates to the synthesizer. Each of these templates is then mapped into one or more prosodic specifications producing a network of targets. For example, as illustrated in Fig. 4.2, the system response can be either one of

"Will you return to Seattle from Boston?"
"Will you return from Boston to Seattle?"

and each of these word sequences can have various prosodic realizations as shown in Fig. 4.3. At run time, the word and phrase lattice L is expanded by composing it with the network of prosodic templates T and the resulting target, shown in Fig. 4.7, is then combined with the decision-tree-based word-level prosody prediction module P by the union operation. The resulting network is composed with the unit selection WFST U and the best path through the final transducer determines the output sequence of units: $\pi_2(BestPath([(L \circ T) \oplus P] \circ U))$, where π_2 is the projection operation that chooses the output labels. The composition operation is better than a sequential application of the two transducers because it allows for a wider search space by not

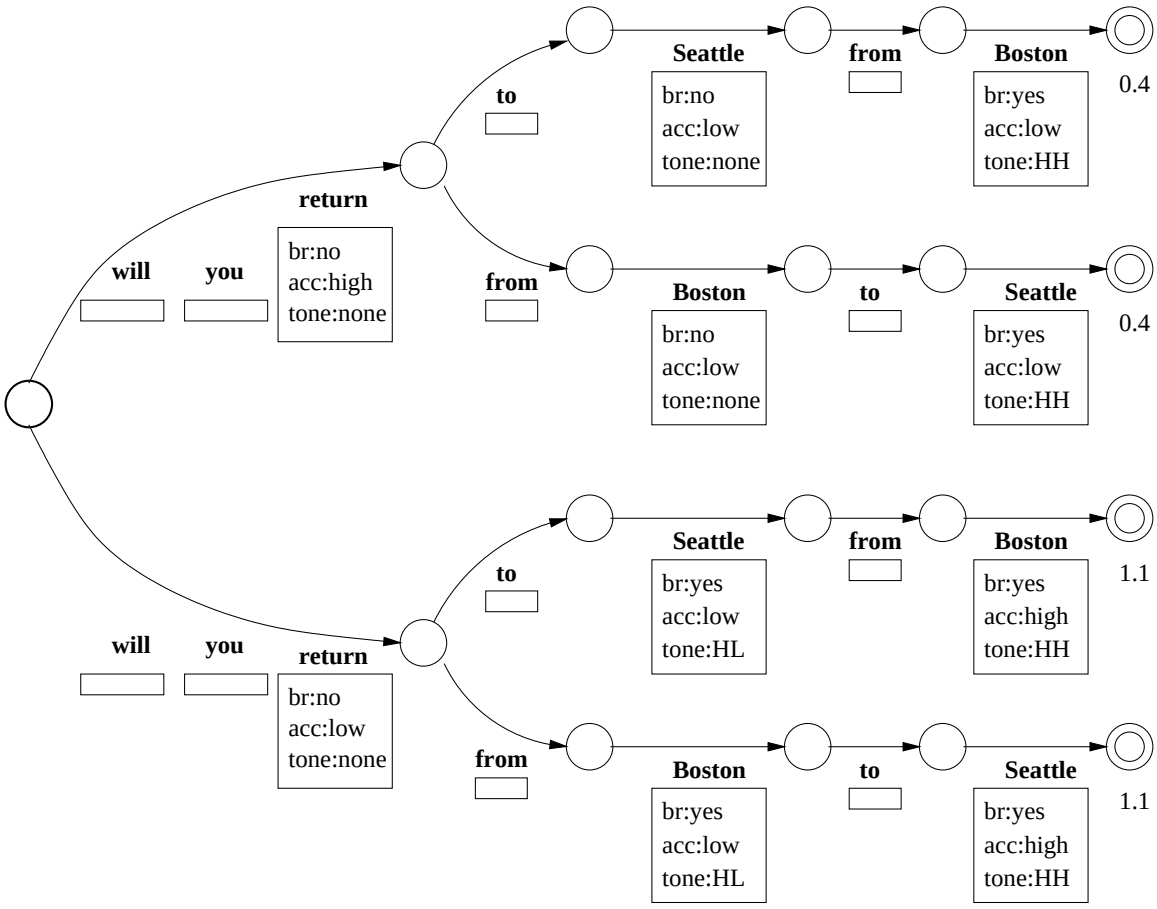

Figure 4.7: An example of a finite-state network of targets containing multiple word sequences and various prosodic specifications from a template prosody WFST. Values in boxes describe prosody, i.e breaks, accents and tones. Empty boxes correspond to an absence of any prosodic labels.

making a hard decision when predicting prosody. Prior to performing the union and composition operations we minimize the transducers for efficiency.

Each alternative path through the network is weighted according to its relative frequency in the training data. The weights (negative log probabilities) are linearly combined in the composition step and assigned to the exit states, represented by double circles in Fig. 4.7. The relative scaling of costs across these two WFSTs can be tuned according to a given task. We used informal listening tests in order to obtain

relative scaling factors for each type of cost. However, perceptual experiments may be useful to better optimize the scaling factors.

The modular WFST architecture makes it easy to add new components to the synthesizer. For example, one can design a letter-to-sound WFST that models different pronunciations or simply store a short list of alternatives in a dictionary (actually, a separate WFST), which would expand the space of candidate units even further, provided that the unit selection WFST supports subword units.

4.3 Experiments

This section provides details on our experiments. First, Section 4.3.1 describes the prosody prediction results. Then, Sections 4.3.2 and 4.3.3 cover our perceptual experiments demonstrating the benefits of synthesizing from multiple prosodic targets and multiple word sequences respectively.

4.3.1 Prosody Prediction

The entire prosodically labeled part of the Communicator corpus (see Chapter 3) (2752 words) was randomly split into training and test sets. Three quarters of the data was used for training the decision trees, and the rest was used for testing. To alleviate the data sparsity and to lessen the effects of labeling inconsistency we have converted the ToBI labels into a simplified representation, where pitch accents were compressed into three categories: high (H*, L+H*), downstepped (!H*, L+!H*, H+!H*), and low (L*, L*+H). Four possible types of boundary tones were used (L-L%, L-H%, H-L%, H-H%), but only major prosodic boundaries (break index 4) were annotated. This representation focuses on maximizing the inter-transcriber agreement [69] and is different from previously proposed ToBI simplification in [105], where the authors grouped various types of ToBI pitch accents into two categories: bi-tonal and other, according to the perceptual prominence ratings submitted by four labelers. Our view

is that prominence should be represented as a different dimension than tone, and that it is perceptually odd to group L* and H* together.

Prediction results for the symbolic prosodic labels described earlier and a reduced class of accented vs. unaccented words are summarized in Table 4.1. Chance performance was computed by always predicting the most frequent label; all predictors did significantly better than chance ($P < 10^{-3}$). The column indicating labeler agreement provides an upper bound on performance for each category. This bound, however, is only approximate in that it is based on results reported in [69] while only one labeler marked the Communicator corpus used in our experiment.

While we were able to achieve high accuracy with breaks and boundary tones, accent tone prediction was not very accurate. This is consistent with the findings in [79] and the view that there is more variability allowed in accent tone type. Even if we compress all types of accent labels into one category (± presence of a pitch accent, labeled "accents-binary" in the table), the accuracy is low (74.2%) compared to the 80-85% accuracy reported for decision tree labeling of binary accents on other corpora [36, 79].[4] However, we verified that accuracy of the decision trees on a Radio News corpus is similar to other results on that corpus [79]. The low accuracy on accent tones provides further motivation for allowing multiple targets rather than a single one.

By examining the questions used in the trees we found that use of features describing a word's position in the prosodic phrase did not improve performance on the test set. Syntactic features were especially useful for predicting phrase breaks, while the part-of-speech labels, function word indicator, city name and number flags substantially benefited accent and tone prediction. We also found that break labels were somewhat useful for predicting accents, and accents as a feature, in turn, improved boundary tone prediction.

---

[4]Note that the results here are reported at the word level (as opposed to the syllable level in [36]).

Table 4.1: Prosody prediction results, where "chance" performance is computed by always predicting the most frequent label. Labeler agreement is based on results reported in [69]. For pitch accents, labeler agreement (72.4%) was computed using three categories: none, non-downstepped and downstepped.

Type of prosody and prosodic labels	Chance performance	Prediction accuracy	Labeler agreement
Breaks (none, major break)	81.0%	92.2%	93.4%
Accents (none, high, low, downstepped)	46.6%	59.4%	$< 72.4\%$
Accents-binary (none, accent)	53.4%	74.2%	80.6%
Boundary Tones (LL, LH, HL, HH)	69.9%	86.4%	90.9%

4.3.2 Synthesizing with Multiple Prosodic Targets

We constructed prosodic templates (as described in Section 4.2.1) for several types of target sentences common to the text generator, each comprised of a sequence of the compressed ToBI labels. We limited our focus to several types of sentences containing city names in various prosodic contexts. Through informal interactions with the dialog system we found that these types of utterances often had incorrect prosody and could potentially benefit from better prosody prediction.

Fourteen target sentences were synthesized by three different methods (see Table 4.3): A_1) no prosody prediction, with unit selection based entirely on the cepstral concatenation costs; B_1) only one zero-cost prosodic target in the template (the most frequent), with all other alternatives having very high and equal costs; and C_1) a prosody template that allows alternative paths weighted according to their relative frequency (unobserved events are assigned a fixed and significantly higher cost). The target sentences were chosen so that they do not match any single continuously recorded utterance in the database in its entirety.

We conducted a perceptual experiment, where subjects ranked versions A_1, B_1

Table 4.2: Perceptual experiment results: synthesizing with multiple prosodic targets. Shown are pairwise comparisons: A_1 vs. B_1 and B_1 vs. C_1, counting only those cases where the utterances were different. Significance of the results is measured by the probability P of sample medians not being significantly different.

A_1 vs. B_1				B_1 vs. C_1			
Wins		Ties	Significance	Wins		Ties	Significance
A_1	B_1		test, P	B_1	C_1		test, P
25	18	1	0.360	1	55	0	1.58×10^{-15}
56.8%	40.9%	2.3%		1.8%	98.2%	0%	

and C_1 (relatively) based on their naturalness. There were five subjects, all native speakers of American English, and each ranked all three versions of 14 responses. The subjects were allowed to play the three versions of each response any number of times in any order. Scores of equal ranks were allowed in case a subject cannot perceive the difference between given samples (some are identical) or considers them equally natural. The order of sentences and of the three different versions for each was randomized. The subjects were speech researchers but were naive with respect to the system implementation and corpus. The rankings submitted by one subject were excluded from the final results because they exhibited a much larger variance than that of other subjects. Excluding this subject, there were 56 trials in total. However, in several cases waveforms were identical in versions A_1 and B_1, so there were in effect only 44 trials for the A-B comparison.

The results of our perceptual experiments (as illustrated in Table 4.2) show that version C_1 was rated the most natural very consistently. It was rated the best of all three versions in 50 trials (or 89.3% of the time), and was at least as good as either A_1 or B_1 in 53 trials (or 94.6% of the time). Versions A_1 and B_1 did not show consistent preference differences, though A_1 was rated above B_1 somewhat more often, probably

because it tended to have smoother (and fewer) concatenations. Two (from the total of fourteen) sentences happened to be the most favorable to version A_1, probably for this reason. Excluding these sentences brings the winning rate of C_1 over A_1 and B_1 up to 98%, suggesting that some attention to signal processing aspects of concatenation would lead to an increased importance of prosodic match.

4.3.3 Flexible Text Generation and Prosody Prediction

From the results described in the previous section it is evident that the use of multiple prosodic templates can significantly benefit the output quality. The experiments, however were limited by a relatively small number of trials and a little variety in the types of utterances used. We conducted another perceptual experiment designed to investigate the effects of allowing alternative sequences of words as well as multiple prosodic targets in the template, as it was explained in Section 4.2. The results are based on a larger number of trials and a greater variety of targets than those described in Section 4.3.2. The experiment design here is also somewhat different. First, more subjects participated in this experiment (sixteen as opposed to five), and most of the subjects this time were not speech researchers. In our current experiment, we used hand-corrected word boundaries and LSF-based (as opposed to MFCC-based) concatenation costs, which improved the overall speech quality for all synthesized versions. This experiment also makes use of the word-level prosody prediction based on the decision trees reported in Section 4.3.1.

Twenty target sentences were synthesized by three different methods: A_2, B_2, C_2; outlined in Table 4.3. Version A_2 made no use of prosody, and hence, the synthesis was driven entirely by the concatenation costs. In version B_2, each template had only one word sequence represented by a single zero-cost prosodic target; all other prosodic alternatives were weighted according to the word-level prosody prediction performed by the decision trees as described in the previous section. Finally, in version C_2, we made use of alternative word sequences as well as multiple (weighted) prosodic targets.

Table 4.3: Differences in synthesis methods used to produce the versions evaluated in the perceptual experiments.

Version	Word sequences in the template	Prosodic targets in the template	Word-level prosody prediction
$A_1 = A_2$	single	none	no
B_1	single	single	no
C_1	single	multiple	no
B_2	single	single	yes
C_2	multiple	multiple	yes

The target sentences were chosen so that they do not match any single continuously recorded utterance in the database in its entirety.

The procedure was similar to that of the experiment described in the previous section. This time we had sixteen subjects, most of whom were not speech researchers. Again the rankings submitted by one subject were excluded from the final results because they exhibited a much larger variance than that of other subjects. Excluding this subject, there were 300 trials in total. However, in several cases waveforms were identical in versions A_2 and B_2, so there were in effect only 194 trials for the $A_2 - B_2$ comparison.

The results of our perceptual experiments are illustrated in Table 4.4. A pairwise comparison of the three versions shows that: 1) on average, version B_2 was rated more natural than version A_2, indicating that even simple prosody controls can be helpful; and 2) version C_2 was preferred by the listeners more often than version B_2, which points out the benefit of using multiple word sequences and prosodic targets in the template. Version C_2 was rated the best of all three versions in 202 trials (or 67.3% of the time), and was at least as good as either A_2 or B_2 in 223 trials (or 74.3% of the time).

Table 4.4: Perceptual experiment results: flexible text generation and prosody prediction.

A_2 vs. B_2				B_2 vs. C_2			
Wins		Ties	Significance	Wins		Ties	Significance
A_2	B_2		test, P	B_2	C_2		test, P
59	90	45	0.0122	70	212	18	4.60×10^{-17}
30.4%	46.4%	23.2%		23.3%	70.7%	6.0%	

These results support the main conclusion in the previous experiment, which was that it is useful to allow prosodic alternatives in the unit selection search, even though the experiment design was somewhat different. The fact that we observed improved performance of version B_2 over version A_2, which was not observed for the case of a single prosodic prediction in version B_1 in the previous experiment, is probably due to the use of the general prosody prediction module as a back-off mechanism.

It is interesting to note that several subjects reported that their scoring was influenced by the alternative word sequences that version C_2 made use of. Sometimes they strongly preferred one wording over the other hence biasing their scores toward one word sequence. The latter aspect emphasizes the importance of being able to weigh alternative word sequences according to user preferences.

4.4 Summary

In summary, we have demonstrated that by expanding the space of candidate responses in a dialog system we can achieve higher quality speech output. Specifically, alternative word sequences in addition to multiple prosodic targets were used to diversify the output of the language generator, taking advantage of the natural (allowable) variability in spoken language. Instead of specifying a single desired utterance, we

make a "soft" decision about the word sequence and the prosodic target and let the synthesizer select the utterance that will sound most natural. Results of perceptual experiments show that by integrating the steps of language generation and speech synthesis we are able to achieve improved naturalness of synthetic speech compared to the sequential implementation.

The WFST architecture offers a convenient representation of multiple targets for all components in the system. The standard operations of union, composition and finding the best path on WFSTs allow efficient ways of combining the network of targets, the template and prosody prediction WFSTs, and the unit database at run time, offering the real-time performance needed for interactive dialog applications.

Generation of multiple candidate responses allows the system to automatically optimize the output for the best quality, when provided with a rich inventory of templates. This takes the burden of tuning the language generator to the speech synthesizer away from the system developer, providing for better plug-and-play capability of the individual modules. Furthermore, the flexibility of targets simplifies the process of extending the system with new types of responses since it would require less (or much less) additional data from the target speaker.

Chapter 5

SEARCH SPACE REDUCTION FOR EFFICIENT UNIT SELECTION

5.1 Computational Aspects of Unit Selection

Selecting subword units from a large inventory poses a challenge from the computational performance perspective, since speech synthesis applications generally require real-time performance. Comparing all possible unit sequences in terms of expected output quality is impractical due to a very large number of such sequences. The approaches to limiting the unit selection search space in such a way that minimizes potential quality degradation are divided into two main groups based on whether the search space reduction is performed before or during synthesis. *On-line* methods are applied at run-time and typically imply pruning the search tree according to some threshold value of the combined cost and/or controlling the search tree expansion rate by setting a limit on the branching factor. *Off-line* methods take place during the database construction and range from unit space reduction to precomputing and storing some of the run-time parameters. In this chapter we present several novel methods, both on-line and off-line, that allow unit selection to be performed more efficiently. Developing these methods we also aim to improve the output quality by using models of speech dynamics, some of which had not been applied to speech synthesis previously.

Finding the best unit sequence involves a Viterbi search, because the concatenation cost, being a part of the optimization criteria, is a function of two consecutive candidate units (see Section 2.4). As a result, the search complexity grows quadrati-

cally with the number of candidate units considered for each phone (or subphone) in the target sequence. Controlling the number of candidates to be considered for each target can be done off-line by clustering similar units according to their phonetic and prosodic context. Then at run-time a cluster of candidates is selected that matches the context of the target. Clustering is typically accomplished by means of decision trees, where clustering criteria is based on an objective function such as a difference in acoustic features or likelihood of cepstral vectors, and the tree is formed by asking questions about the unit's phonetic and prosodic context. This chapter will explain our approach to decision tree clustering, which includes some specifics of computing the objective distance function, the choice of features used for building the tree, and post-training adjustments to the clusters that overcome some of the limitations of decision trees.

On-line pruning of the search tree is typically done by limiting the number of candidates based on the target cost and/or selecting up to a certain number of candidates that result in the smallest concatenation cost with units from the previous step. While taking the concatenation cost into account when pruning the search space makes it possible to find a better overall match, it also requires more computation as all pairs of units must be considered. In this work we improve the flexibility of the unit selection framework by introducing *splicing* costs that provide controls for quantifying the potential perceptual discontinuity at a given boundary, separately from the spectral mismatch between the candidate units. Splicing costs make it possible to decide which boundaries are potentially bad for splicing and prune the search tree prior to evaluating all possible concatenations.

Another approach to reducing the candidate space is to prune the unit database by permanently removing outliers or overly common units. This method is often combined with decision tree clustering so that each cluster is pruned individually according to unit's relative position within the cluster. We take this approach a step farther and, in addition to eliminating units, prune the space of potential join points

by disallowing a splice for entire classes of boundaries. Here we develop a methodology for automatically classifying unit boundaries according to their potential for making smooth concatenations.

Since computing concatenation costs requires floating-point operations with vectors of high dimensionality a substantial part of run-time computation can be reduced if these operations are performed off-line. However, computing and storing concatenation costs for all possible pairs of units is impractical for large databases. One solution is to precompute and cache concatenation costs between the most frequently used pairs of units [7]. A different approach uses vector-quantization to encode the space of units and store a complete distance table between groups of units indexed by the VQ codebook entries [6, 22]. In our experiments we take the vector quantization approach to computing concatenation costs, as will be described in Chapter 6.

5.2 Clustering Similar Units

As mentioned above, clustering is used to reduce the search space by reducing the number of candidate units for a given target. Our approach to clustering builds on the previous work by Black and Taylor [11]. We used the same parametric representation of the signal and the same set of software tools [99]. The main differences between our approach and that of [11] are in: 1) acoustic feature normalization and handling of F0 when computing the acoustic distance between pairs of units; 2) the choice of features used for clustering (i.e. for forming the decision tree questions); and 3) adjustments made to the clusters after constructing the tree. Of these, the key new contribution is cluster adjustment which allows units to be shared among several clusters in cases where the acoustic spaces of these clusters overlap. Such a clustering strategy increases the possibilities for finding an optimal unit sequence during unit selection. The increase in the set of candidate units is offset by pruning outliers.

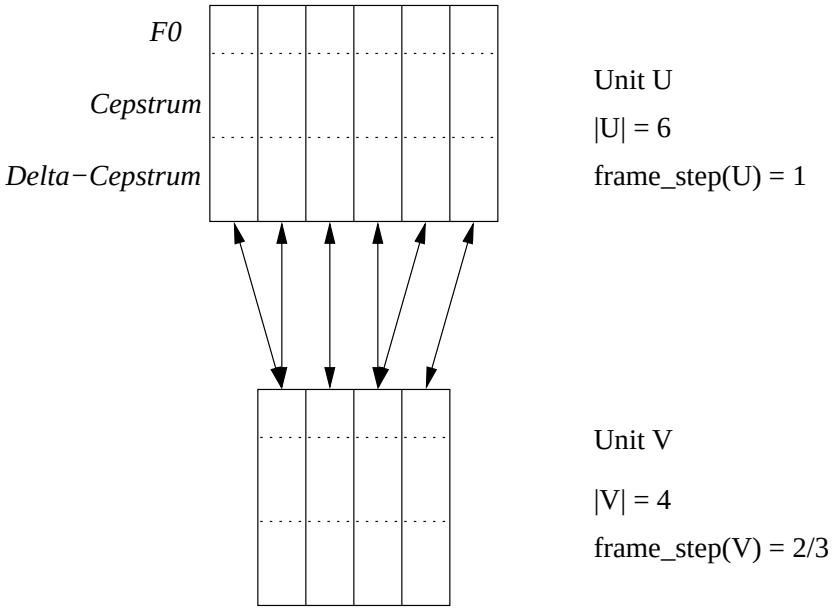

Figure 5.1: Aligning frames in order to compute the distance between two units U and V.

5.2.1 Computing Acoustic Distance

We used MFCC coefficients of order 12, MFCC delta coefficients, and F0 to represent the acoustics of unit waveforms. Feature vectors were extracted at a 5 msec analysis step. We used these feature vectors (called frames) to compute the objective distance that provided a criteria for clustering, i.e. units were clustered in such a way as to minimize the average distance between units in each cluster. The distance between two units U and V is computed by aligning their frames and stepping through the corresponding pairs of frames, as shown in Fig. 5.1. We compensate for differences in unit durations by scaling[1] the step with which we progress through the shorter unit. For example, in Figure 5.1 unit V is shorter and hence we step through its frames by increments of 2/3.

[1]While a linear alignment is suitable for short segments such as half-phones (as we use in this work), a different distance metric (e.g. HMM alignment) is preferable for units longer than a phone.

Formally, the distance $D(U,V)$ between two units U and V (assuming U is longer than V) is computed as follows

$$D(U,V)_{|U|>|V|} = w_D d_D + \frac{1}{|U|}\sum_{i=1}^{|U|}\left[w_C d_C(u_i, v_{j(i)}) + w_{F0} d_{F0}(u_i, v_{j(i)})\right] \ , \qquad (5.1)$$

where d_D is the penalty for the difference in unit durations; $d_C(u_i, v_j)$ and $d_{F0}(u_i, v_j)$ are the cepstral and F0 distances between the i-th frame in unit U and j-th frame in unit V; and $j(i) = \left[i\frac{|V|}{|U|}\right]$ rounded to the nearest integer. The quantities w_D, w_C, w_{F0} are duration, cepstra, and F0 weights assigned to the corresponding distances. By manipulating these weights we can control the relative contributions of duration, cepstra and F0 differences to the total distance. In [11] the authors used duration penalty d_D as a multiplicative factor, i.e. the overall cost was scaled by d_D. We chose for duration penalty to be an additive component to the overall distance, and the frame-based distances d_C and d_{F0} to be normalized by the number of frames, since such metric offers better controls for scaling the individual distances.

The duration distance d_D is simply

$$d_D = \frac{|U|}{|V|} - 1 \ ,$$

where $|U|$ and $|V|$ are unit durations in frames and $|U| > |V|$.

The cepstral distance d_C between two frames u_i and v_j is the normalized Mahalanobis distance between the vectors of MFCC coefficients:

$$d_C(u_i, v_j) = \frac{1}{N}\sum_{k=1}^{N}\frac{\left(f_k(u_i) - f_k(v_j)\right)^2}{\sigma_k^2} \ ,$$

where N is the length of cepstral vector (including the delta coefficients) and f_k are the individual feature components. The variances σ_k^2 are computed across all units of the given type (e.g. all instances of left half of phoneme "aa") prior to clustering.

Unlike in [11], when computing the F0 distance d_{F0} we take into account voicing of speech frames and we normalize all distances by the appropriate variance. The F0

distance consists of two components – the difference in absolute F0 values $d_{|F0|}$, and the difference in the rate of F0 change $d_{\Delta F0}$:

$$d_{F0}(u_i, v_j) = d_{|F0|}(u_i, v_j) + d_{\Delta F0}(u_i, v_j) \ .$$

The value of $d_{|F0|}$ depends on voicing of the frames u_i and v_j, and is computed as follows

$$d_{|F0|}(u_i, v_j) = \begin{cases} \frac{(F0(u_i) - F0(v_j))^2}{\sigma^2_{F0}}, & \text{if both frames are voiced} \\ 0, & \text{if both frames are unvoiced} \\ 1, & \text{otherwise} \end{cases}$$

where σ^2_{F0} is the variance of F0 in voiced frames observed for the given type of unit.

The value of $d_{\Delta F0}$ is also the Mahalanobis distance between the corresponding features $\Delta F0$

$$d_{\Delta F0}(u_i, v_j) = \frac{(\Delta F0(u_i) - \Delta F0(v_j))^2}{\sigma^2_{\Delta F0}}$$

where $\Delta F0$ is the difference in F0 between two consecutive voiced frames:

$$\Delta F0(u_i) = \begin{cases} F0(u_i) - F0(u_{i-1}), & \text{if both frames are voiced} \\ 0, & \text{otherwise} \end{cases}$$

and $\sigma^2_{\Delta F0}$ is computed over the voiced frames observed for the given type of unit.

5.2.2 Building Decision Trees

Using the distance measure formulated above we computed distances between all pairs of units that belong to the same category (i.e. a specific type of half-phone). So, for instance, all left halves of phone "ax" were grouped together, and clustered separately from the right halves of the same phone or any other phones. This acoustic measure is used to compute the *impurity* of a cluster of units, which is defined as the mean distance between all pairs of units in the cluster. The objective is to split the cluster in such a way that the impurity of its subsets is minimized. Splitting can be repeated

recursively until the cluster size reaches some minimal threshold or the impurity can no longer be significantly reduced.

Splitting of clusters was done by means of the CART method [15], i.e. by building a decision tree. Each non-final node of the tree contains a yes/no question about the phonetic, lexical and/or phonological context of a given unit (e.g. "is the right neighbor a nasal?", "is the syllable stressed?", "does the syllable have a pitch accent?"). The CART algorithm chooses these questions automatically by performing a greedy search over the set of phonetic, lexical and phonological features provided. At each step the algorithm selects a question that offers the best (in terms of impurity) split of the cluster.

Features that we used for constructing the trees are listed in Table 5.1. They consist of phonetic features describing unit's left and right neighbors, features related to unit's position within the syllable, and several features containing higher level information such as the indication of stress, syllable's position in the word and in the prosodic phrase, and ToBI accents and tones. Black and Taylor [11] made continuous prosodic parameters (pitch and duration including that of previous and next units) available for building the decision trees. We took a different approach by using the symbolic prosodic markers (phrase breaks, pitch accents and tones) for decision tree questions, and the pitch and duration in the impurity function. There are two major problems with specifying a detailed F0 contour in the target prior to unit selection: 1) the predicted contour may contain errors, 2) even if accurate, a contour is never unique for a natural utterance. By using symbolic prosodic markers for clustering we reduce the importance of explicit prediction of a detailed F0 contour. Since the decision tree building criteria is based on the acoustic distance which incorporates pitch and duration, these two parameters are predicted implicitly by selecting units from the appropriate clusters.

When we varied the weights w_d, w_{F0} and w_C of duration, pitch and cepstrum distances respectively (see Equation 5.1) we observed that the usefulness of features

Table 5.1: Features used for clustering.

Phonetic features describing the left(l) and right(r) neighboring segments	
ph_name	name of the segment
ph_vc	vowel or consonant (+ -)
ph_vlng	vowel length (short, long, diphthong, schwa)
ph_vheight	vowel height (high, mid, low)
ph_vfront	vowel frontness (front, mid, back)
ph_vrnd	lip rounding (+ -)
ph_ctype	consonant type (stop, fricative, affricate, nasal, lateral, approximant)
ph_cplace	consonant place of articulation (labial, alveolar, palatal, labio-dental, dental, velar, glottal)
ph_cvox	consonant voicing (+ -)
Unit's position within the syllable	
pos_in_syl	segment position in syllable (integer)
relative_pos	relative position of segment (early, mid, late)
syl_initial	segment is the first in the syllable (+ -)
syl_final	segment is the last in the syllable (+ -)
onsetcoda	part of syllable the segment belongs to (onset, coda)
l/r.onsetcoda	left/right neighbor's position in syllable (onset, coda)
Higher level information	
syl_size	number of phone segments in the syllable
l/r.syl_break	break level at the left/right syllable boundary (word internal(0), word-final(1), and major(4) phrase final)
position_type	position of syllable in word (single, initial, mid, final)
stress	the lexical stress of the syllable
tobi_accent	ToBI accent on the syllable (high, low, downstepped, none)
tobi_endtone	major boundary tone (L-L%, L-H%, H-L%, H-H%, none)

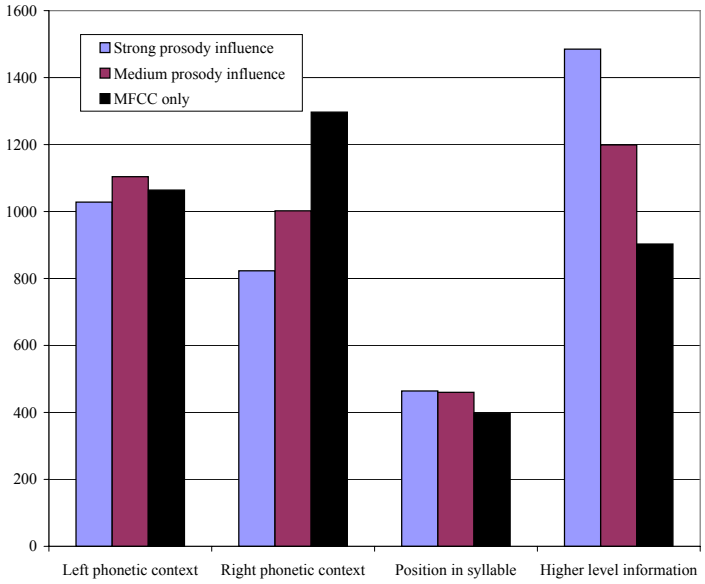

Figure 5.2a: Radio News corpus

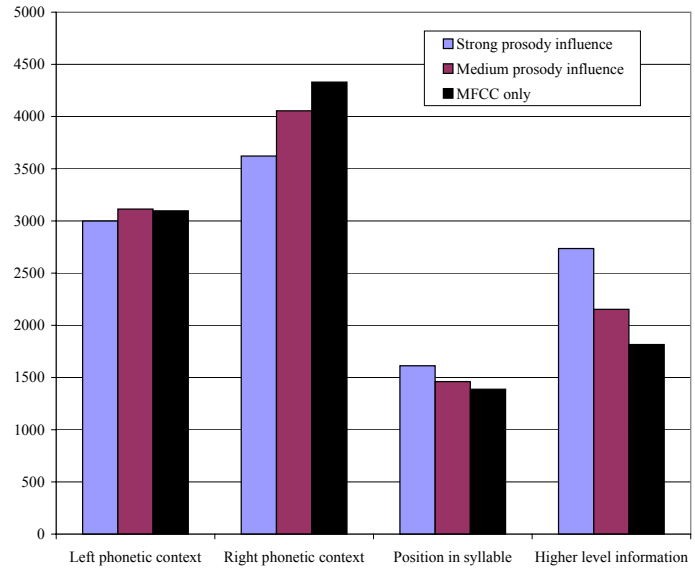

Figure 5.2b: Communicator corpus

Figure 5.2: Histograms illustrating the usefulness of different categories of features for constructing the decision trees varying with the amount of influence that prosody (F0 and duration) has on the acoustic distance between units. Each column indicates the number of times a question was asked about a given category of features in all traces from a leaf node to the root of a tree.

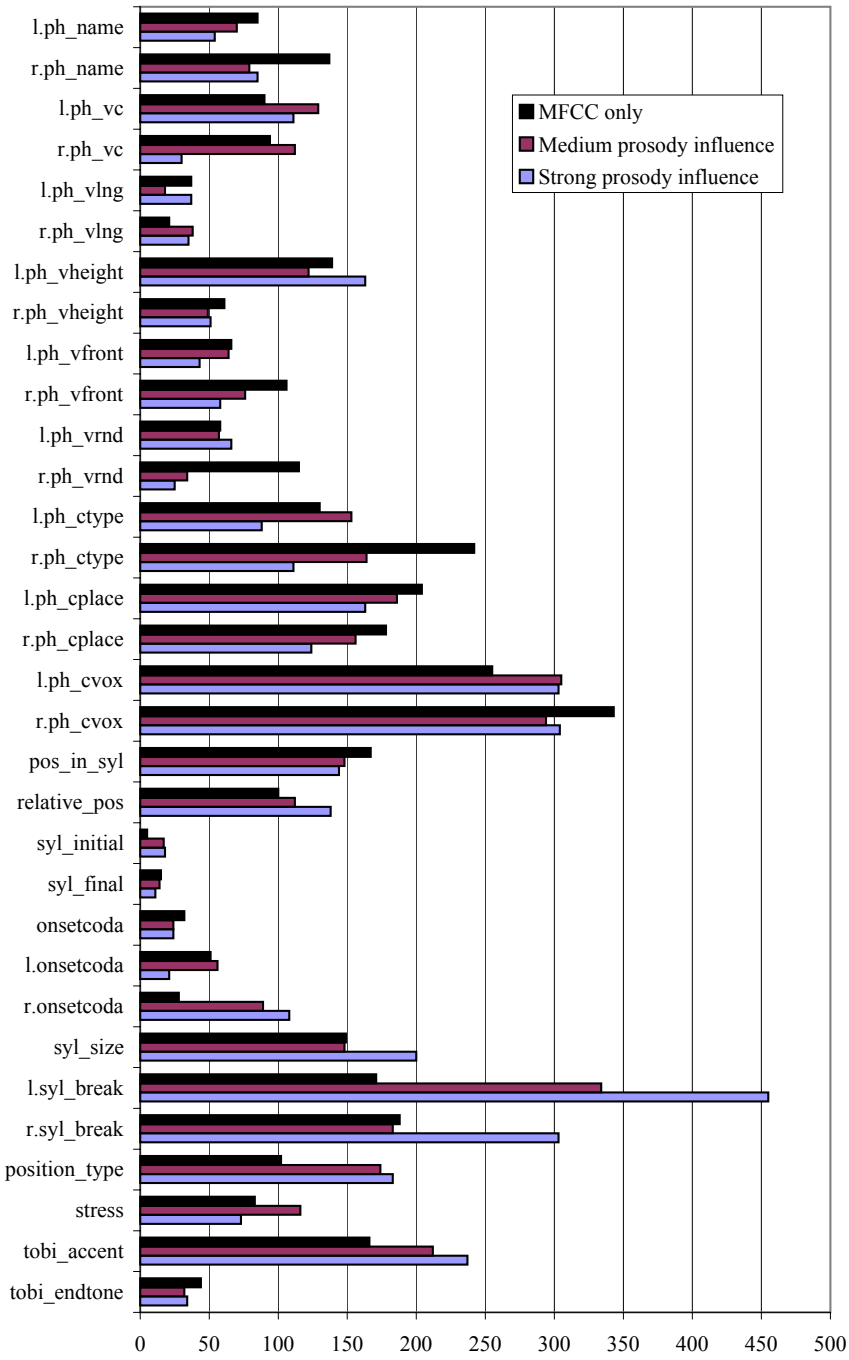

Figure 5.3: A histogram itemized for each individual feature (Radio News corpus).

for building the decision trees shifts from the phonetic context (in particular, the right context) toward higher level information when we increase the weights of duration and pitch in the acoustic distance. Figure 5.2 compares the usefulness of features observed for three different methods of computing the acoustic distances: 1) "strong prosody influence", where duration d_D and pitch d_{F0} distances comprise about two thirds of the total distance; 2) "medium prosody influence", where about one third of the total distance is due to prosody; and 3) "MFCC only", where the distance is computed only from cepstrum. These results were obtained with two data sets: hand-corrected part of the Radio News corpus and the entire training set of the Communicator corpus (see Chapter 3) shown in Figures 5.2a and 5.2b respectively. The difference in scales between these two figures reflects the difference in corpora sizes, i.e. Communicator is a larger corpus and therefore it generates more than twice as many unit clusters as Radio News. Higher level information is not as influential in building the decision trees with the Communicator corpus as it is with the Radio News. This may be attributed, in part, to the fact that only a small portion of the Communicator corpus is prosodically labeled. Also due to the limited domain of application the corpus uses a smaller vocabulary and contains phrases that are repeated in multiple utterances, which may lead to the increased importance of phonetic context observed in clustering the Communicator corpus. A more detailed comparison for each individual feature computed with Radio News corpus is given in Figure 5.3.

5.2.3 Adjusting the Clusters

At each step in building the decision tree the data is split into two mutually exclusive subsets and each of these subsets are then used independently for choosing a question for the next split. As a result, the splits made deeper in the tree are based on progressively smaller amounts of training data. Due to the binary nature of these splits (as opposed to k-means) and the use of phonetic and phonological features (rather than acoustic distances) to choose between units, the resulting clusters may

overlap in the acoustic space covered even though they do not overlap in terms of set membership. Hence many units belonging to different clusters may still be acoustically very similar. This artifact of decision tree clustering limits the potential for finding the best unit sequence when synthesis units are selected from a pool of candidates defined by the decision trees.

If we hypothesize that cluster impurity based on the acoustic distances is a good score then we can increase the possibilities for finding an optimal unit sequences when we include more acoustically similar units into the pool of candidates. Our approach is to allow units to be shared between clusters that are generated by the standard CART algorithm. For each cluster we find a *centroid*, i.e. a unit with the smallest mean distance to all other members of the cluster, and then we search the neighboring clusters and select units that fall within a specified radius from the centroid. These units are then sorted based on their distances from the centroid and up to a certain number of the best candidates are added to the cluster.

While sharing units between clusters offers more flexibility and can potentially benefit the output quality, it also increases the search complexity during unit selection due to an increase in cluster size. Pruning (commonly used in concatenative synthesis [11, 38, 27]) may be necessary to maintain real-time performance. In order to keep the cluster size unchanged we remove as many units (with the largest distance to the centroid) from a given cluster as we have added by allowing units to be shared between clusters. Consequently, units that end up not being assigned to any cluster can be permanently removed from the database. The two conditions that we enforce when expanding the clusters, namely

- a unit is added to the cluster only if its distance to the centroid is smaller than the median distance between the centroid and all other units that originally belong to the cluster, and

- the number of additional units does not exceed 50% of the cluster's original size,

ensure that the impurity of clusters will not increase when we prune the clusters to their original size, and that at least half of the cluster is composed of its original inventory. Figure 5.4 shows the average cluster impurity as a function of the number of units added to the cluster with and without pruning, computed on the Radio News corpus. Without pruning, the average cluster impurity gradually decreases as we add more units to the clusters. Pruning here essentially works as substitution of cluster outliers with out-of-cluster units. A rapid drop in impurity is observed at first when we prune the worst outliers, and then the rate of decrease in impurity gradually levels out. The reduction in average cluster impurity directly translates into smaller (on average) target cost since we compute the target cost by finding the distance between the unit and its cluster's centroid.

5.3 Models of Spectral Dynamics

In concatenative speech synthesis the output quality relies on the system's ability to avoid and, if necessary, to correct for the artifacts introduced when speech segments are spliced together. This motivates the use of models of spectral dynamics in speech, especially for short-term dynamics at the unit boundaries. Previous work has implicitly captured dynamics with derivative features, as summarized in Section 5.3.1. Building on this success, we introduce an alternative model-based approach in Section 5.3.2, and give a unified view of how these models apply to concatenation costs in Section 5.3.3.

5.3.1 Dynamic Features and HMMs

Hidden Markov models (HMM), widely used in speech recognition, represent the time-varying speech spectrum using a Markov sequence of states with the assumption that the vectors of spectral features are conditionally independent (in time) given the state sequence. Local spectral movements are computed at each time frame and included

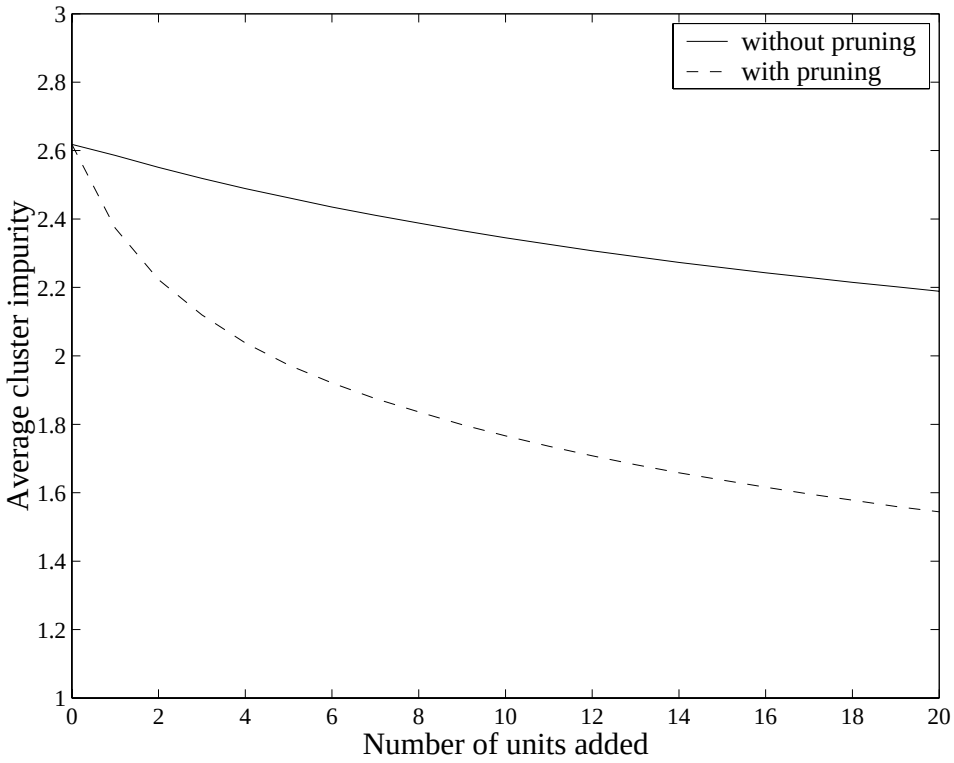

Figure 5.4: The average cluster impurity as a function of the number of units added to the cluster (Radio News corpus). Pruning ensures that the cluster size remains the same.

in the observation vector modeled by an HMM. Features that represent short-term changes in the spectrum are called dynamic features and are typically computed as a linear combination of static features across several time frames. The most commonly used dynamic features in speech recognition are delta features (also called derivatives) describing the rate of change in the spectrum, and delta-delta parameters (also called acceleration) describing the rate of change in the delta features.

The quality of HMM synthesis [101], based on generating a sequence of spectral parameters from distributions defined by HMM, relies heavily on the ability of an HMM to capture spectral dynamics. The authors in [101] developed an algorithm that allows to generate speech parameters from continuous mixture HMMs with dynamic features. It was demonstrated in [101] that synthetic speech obtained without dynamic features exhibits perceptible discontinuities, while with dynamic features the output speech is much smoother. The authors reported a relatively small difference between the speech quality with delta coefficients and that with both delta and delta-delta parameters.

5.3.2 Dynamic Model: BMM

A Buried Markov model (BMM) [8], a form of graphical model [54], augments the dependency structure relative to that of an HMM. In a BMM, each element of a feature vector may include direct dependencies on elements of feature vectors in addition to dependencies already included in an HMM (namely, the hidden state variable and possibly other elements of that same vector). These dependencies may switch based on the current hidden state value. Specifically, if X_t^i is the i^{th} element of the t^{th} feature vector, then a BMM uses the distribution $p(X_t^i|Q_t = q, Z_t^i(q))$, where Q_t is the hidden state at time t and $Z_t^i(q)$ is a q-dependent subset of feature vectors either before, at, or after time t. In this work, all BMM dependencies are linear and distributions are

unimodal Gaussian, in which case

$$p(X_t|Q_t = q, Z_t(q)) \sim N(X_t; B_q Z_t(q) + \mu_q, \Sigma_q),$$

where B_q is a sparse matrix.

Other than issues of training and dependency representation, one of the main challenges in producing a BMM system is in choosing the structure of the dependencies for each state q. Choosing all dependencies leads to an enormous free parameter increase, could lead to over-training, and is probably unnecessary. The goal for structure learning is to choose that minimal set of dependencies which are most appropriate for the task at hand [8]. For automatic speech recognition (ASR), dependencies should be chosen discriminatively, as ASR is inherently a problem of pattern classification. For speech synthesis, however, dependencies should be chosen not so much for their discriminative but rather for their predictive ability — if a BMM can be "predictively structured" so that they predict X_t well given past acoustic vectors, both synthesis quality and quality assessment could improve. A good measure of the predictive ability between two random variables is standard mutual information [23] which we investigate in this work.

In order to better understand the relationship between discriminatively vs. predictively structured BMMs, we performed two informal listening experiments using the synthesis algorithm described in [101]. The first experiment compared speech synthesized from MFCCs that had been randomly sampled from: 1) an HMM, 2) a discriminatively structured BMM (DBMM), and 3) a predictively structured BMM (PBMM). It has been shown in the past that a DBMM can lead to improved ASR results. We predicted that a DBMM would not outperform an HMM for synthesizing speech, but that a PBMM would outperform both. Our hypothesis is based on the fact that the task of ASR is to extract the word sequence for a computer (i.e. intelligibility is the only concern), which is different from synthesis which involves presenting speech to a human listener for whom naturalness is also important. A

model optimized for one task could be ill-suited for the other. The results of informal listening experiments supported this hypothesis; samples based on Radio news data are available at [116].

A second informal listening experiment compared the quality of HMM-synthesized speech when MFCCs included delta coefficients with that of PBMMs that did not include deltas. The HMM-based synthesis algorithm utilized delta-coefficients to better smooth the final speech signal [101]. The results of the experiment showed that there was little if any difference in synthesis quality, but the PBMMs used 25 percent fewer parameters. Even though the overall quality of speech output was not good enough for general user acceptance, these listening tests led us to believe that BMM models are at least as good in capturing spectral dynamics as HMMs, and hence it may be useful in concatenative speech synthesis.

5.3.3 Dynamic Models for Unit Selection

In applying these models to the process of unit selection in speech synthesis, we look specifically at concatenation points. We introduce below the idea of using the probability of the model prediction residual for a concatenation cost. In the next section, we introduce an extension to concatenation costs – splicing costs – which also builds on the dynamic model and offers the potential for reducing search complexity by constraining the set of possible splicing points.

The concatenation cost proposed in [28] is effectively the negative log likelihood of the prediction residual using the simple predictor $\hat{X}_{t+1} = X_t + \mu_q$, where q represents the context (which could be described by an HMM state index). Here, we extend the notion using the BMM, which involves a more general linear predictor of the form $\hat{X}_{t+1} = B_q Z_t(q) + \mu_q$, where $Z_t(q)$ may include X_t as well as elements from other time vectors. Our hypothesis is that if, given the phonetic context and the spectrum (specifically LSFs) on one side of the boundary, the prediction residual is relatively low (high log likelihood) for the features in a different unit, then the concatenation

cost should be low. However, for splicing costs, which will be described in the next section, we estimate the potential discontinuity when the actual adjacent candidate units are not specified. For this condition we hypothesize the opposite, i.e. that splicing costs should be *inversely* related to the predictability of the spectrum across the boundary. In particular, when a *linear* predictor works well in general, then the spectral dynamics are naturally smooth and the boundary is a poor choice for making a splice.

5.4 Unit Selection with Splicing Costs

Previous work by Yi [109] explores the use of perceptually tuned phoneme class-based concatenation costs, where the costs were manually set according to the smoothness of joins produced by concatenating units from specific phonetic categories (e.g. a vowel followed by a nasal). Their results showed that splices made at boundaries of fricatives and stops (i.e. points of abrupt spectral change) produce the least amount of distortion. This result led us to the idea that there are two different factors that determine the goodness of a concatenation: 1) the closeness of the spectral shape of the concatenated unit to that of a naturally occurring continuation (the standard distance-based view of concatenation costs), and 2) the relative abruptness of spectral change at the candidate boundary (*splicing costs*).

In this work, we define a splicing cost to be a measure of the potential discontinuity that a given unit may incur when a splice is made at its boundary. Each unit has two splicing costs: one for each of its left and right boundaries respectively. As described in Section 2.4.2, the task of unit selection is, given the target sequence $(t_1, \ldots t_N)$, to find database units $(u_1, \ldots u_N)$ that minimize the total cost C_{total} which is the sum of the target C^t and concatenation C^c costs. We modify the total cost to include the left C^{left} and right C^{right} splicing costs as follows:

$$C_{total} = C^t(u_1, t_1) + \sum_{i=2}^{N} \Big[C^{right}(u_{i-1}) + C^c(u_{i-1}, u_i) + C^{left}(u_i) + C^t(u_i, t_i) \Big]$$

Since our units are half phones we have splicing costs for both phone boundaries and mid-phones, and hence can dynamically choose most appropriate join points. In some sense, splicing costs are not needed since they could be incorporated as part of the concatenation cost. However, controlling splicing costs separately from concatenation costs can improve efficiency of the dynamic search at run-time if the search tree is pruned based on the splicing costs prior to evaluating all possible concatenations. Furthermore, consider $C^s(u_{i-1}, u_i) = C^{right}(u_{i-1}) + C^c(u_{i-1}, u_i) + C^{left}(u_i)$ to be the cost of splicing two units u_{i-1} and u_i, where C^{right} and C^{left} represent the effect of discontinuities and C^c captures the spectral match to natural speech. It there are indeed two such factors influencing our perception, factoring them out in the costs makes it easier to tune the system parameters and to perform controlled perceptual experiments.

5.4.1 Splicing Costs as the Inverse of Spectral Change

We hypothesize that unit boundaries where the spectral characteristics in the original recording are changing rapidly (fricatives and stops generally fall into this category) are good splicing points. Therefore we investigated a splicing cost that is inversely related to the spectral change at a given boundary. We explored a simple measure of rapid change – the Mahalanobis distance between two successive LSF vectors at the splice point, and consequently the splicing costs were computed as inverse of this distance as follows (for two units u_k and u_{k+1} adjacent in the original recording):

$$C^{right}(u_k) = C^{left}(u_{k+1}) = \left(\sum_{i=1}^{D} \frac{(f_i(u_k^{last}) - f_i(u_{k+1}^{first}))^2}{\sigma_i^2} \right)^{-1},$$

where u_k^{last} represents the last frame of LSFs in unit u_k, u_{k+1}^{first} is the first frame of LSFs in unit u_{k+1}, f_i are the individual LSF features.

We used the unit database extracted from the Communicator corpus (see Chapter 3) containing approximately 140,000 units to synthesize fourteen target utterances. The units were clustered by means of decision trees as described in Section 5.2, but no

adjustments such as pruning or expansion were done to the clusters. For units within a given cluster the target costs were equal to the acoustic distance between the unit and the cluster centroid, as described in Section 5.2. Frames at unit boundaries were vector quantized using a 256-vector codebook.[2] The frames were 10 msec wide and contained line spectral frequencies (LSF) of order 18, energy, and F0. Concatenation costs were then computed between each pair of VQ codebook entries by taking the Mahalanobis distance between overlapping vectors at candidate join points.

Normalization and/or relative scaling of costs is a design issue, since the costs are not equally important and the different types of costs can have different average values. In practice, concatenation costs are typically scaled higher than target costs to compensate for higher human sensitivity to smoothness of speech [11]. Through informal listening we found that scaling concatenation and splicing costs such that their means are 10 times larger than the mean target cost gives the best performance.

There were no spectral smoothing or prosodic modification applied to the signal when units were concatenated. The target sentences were taken from the same domain, i.e. travel planning, but were produced by a different text generator, so while there was some overlap in the vocabularies, many of the target words were not present in the database recordings. Each utterance was synthesized in two different versions: A) where splicing costs are computed as inverse of Mahalanobis distance between two consecutive frames at a given boundary; and B) where all splicing costs are zero. In version B we increased concatenation costs by a factor of two in order to maintain the relative importance of target costs at approximately the same level.

We conducted a perceptual experiment, where six subjects, all native speakers of American English, compared the two versions indicating whether one sounds much better/worse (score of 2/-2) or a little better/worse (score of 1/-1) than the other, or if both versions sound about the same (score of 0). There were fourteen target

[2]We did not observe a significant improvement in output quality using a codebook of size 512.

sentences. The order of the the sentences and of the two different utterances for each was randomized. The subjects were speech researchers but were naive with respect to the system implementation.

The experiment results gave an average score of 1.13 in favor of version A, which included splicing costs. Version A received non-negative scores 95% of the time, while 81% of all version A scores were strictly positive. The statistical significance of this result is $P = 4.62 \times 10^{-16}$. For two of the utterances all six listeners indicated that version A was much better (score of 2).

We used the Radio News corpus to repeat the experiment described in this section. Informal listening tests supported our conclusion that use of splicing costs for unit selection leads to improved speech output quality.

5.4.2 Computing Splicing Costs with BMMs

In the previous section we have shown how the output quality can be improved by incorporating the spectral change at unit boundaries, represented in term of the Mahalanobis distance between successive vectors of spectral features, into the unit selection cost metric. However, there is evidence of context dependency in the perceptual significance of rate of spectral change at a unit boundary [28], where a context-dependent covariance (and added mean term) leads to an improved concatenation cost. Interpreting this measure for splicing costs: when the frames on either side of a boundary are very predictable, then effects of coarticulation are strong and the boundary is a poor choice for making a splice.

We conducted experiments with the same unit databases (i.e. the Radio News and the Communicator) and the same set of spectral features (i.e. LSF) as in the previous section. Two BMM models were trained for each cluster: one with dependencies on the preceding frames (i.e. left-to-right feature dependency links), and another with the dependencies on the following frames (right-to-left). For each feature we selected three links that correspond to pairs having the highest conditional mutual information

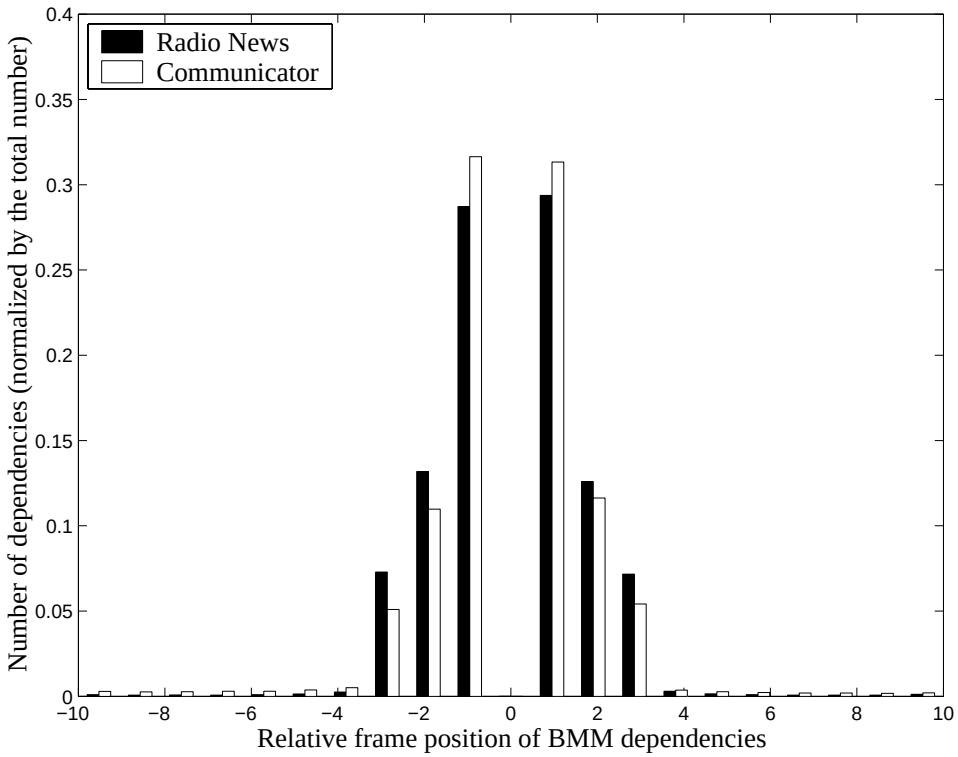

Figure 5.5: A histogram of BMM dependencies according to their relative position (in frames).

(conditioned on the unit cluster identity). We searched up to ten 10 msec frames in the past (or future, depending on the type of model) to select these dependencies. The mean distance was 1.7 frames in the past and 1.6 frames in the future for the Communicator corpus (1.6 frames both in the future and in the past for the Radio News), but features as far back or ahead as 10 frames were occasionally chosen (see Fig. 5.5).

As mentioned above, each unit has left and right splicing costs that indicate suitability of a splice at its left or right boundary respectively. The *left* splicing cost for unit U_i is computed by finding the inverse of the Mahalanobis distance between the first frame of LSFs in the unit and a frame predicted with the "left-to-right" model

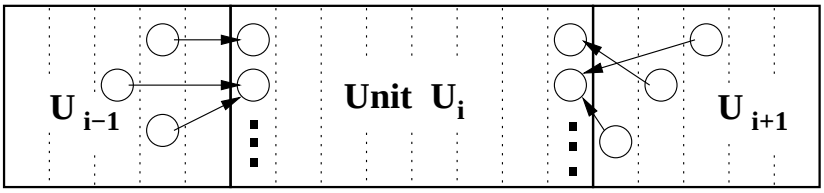

Figure 5.6: Feature dependencies used for computing splicing costs for unit U_i. Dotted lines show frame boundaries of LSF vectors. Circles are individual features.

for the cluster that U_i belongs to, using the data frames that precede unit U_i in the original recording. This is illustrated in Fig. 5.6 at the left boundary of unit U_i. Symmetrically, we used the "right-to-left" model and the data frames following U_i to predict the last frame in the unit. Then the inverse of the Mahalanobis distance between the true and the predicted last frames gives us the *right* splicing cost.

Through informal listening we observed that synthesis quality was as good and sometimes better than when using the inverse of the Mahalanobis distance between two successive frames at the boundary, but the frequency of improved utterances was low.

5.5 Redefining Join Points

In Section 5.4, we suggested that the dynamic search at run-time can be made more efficient if the search tree is pruned based on the splicing costs prior to evaluating all possible concatenations. Another approach is to disallow a splice for entire classes of boundaries. When a unit's boundary falls into one of these classes, the unit itself is still available for synthesis but only via a zero-cost concatenation at this boundary, i.e. the neighboring units must be adjacent in the original recording. Conditional mutual information that we used for selecting the BMM dependencies can provide us with an estimate of the degree of coarticulation at unit boundaries. Our hypothesis is that if the amount of information carried across a boundary is high then the boundary

is a bad place to make a splice.

Note that even when two units are sharing the same boundary (i.e. are adjacent in the original recording) they may have different sensitivity to having a splice made at this boundary. We take this into account by treating the left and the right boundaries independently. For instance, when splicing is not allowed from the left side of the boundary it may still be possible to have another unit spliced on the right. For the left boundaries we compute the conditional mutual information carried across from the previous unit. Conversely, for the right boundaries we collect information that comes from the following unit. Different costs are in fact learned for the two conditions.[3] When we considered pairs of neighboring units with the largest differences between the right splicing cost of the left neighbor and the left splicing cost of the right neighbor (i.e. splicing costs at the boundary that the units share), we found that such differences are predominantly positive (i.e. the right splicing cost is larger than the left one) and occur largely in the middle of vowels. This leads to the conclusion that in the middle of vowels prediction of spectral features is more accurate when it is based on the following (as opposed to the preceding) frames. This observation is consistent with the results from speech recognition indicating strong anticipatory coarticulation effects in vowels.

In our experiments, the unit database (same as in previous section) was altered by gradually reducing the list of potential splicing points. We removed unit boundaries with the largest amount of conditional mutual information carried across the boundary. Entire classes of boundaries were eliminated at once. We kept, however, 5% of units in each cluster with the smallest splicing costs still available for making a splice to ensure that a unit sequence can always be found. Figure 5.7 shows how the total cost of the best unit sequence (taken as an average over thirty target utterances and

[3]Here we maintain two different splicing costs for each boundary, but one could alternatively combine the two costs at each boundary via averaging or taking the maximum value. Additional perceptual experiments are necessary to determine the best approach.

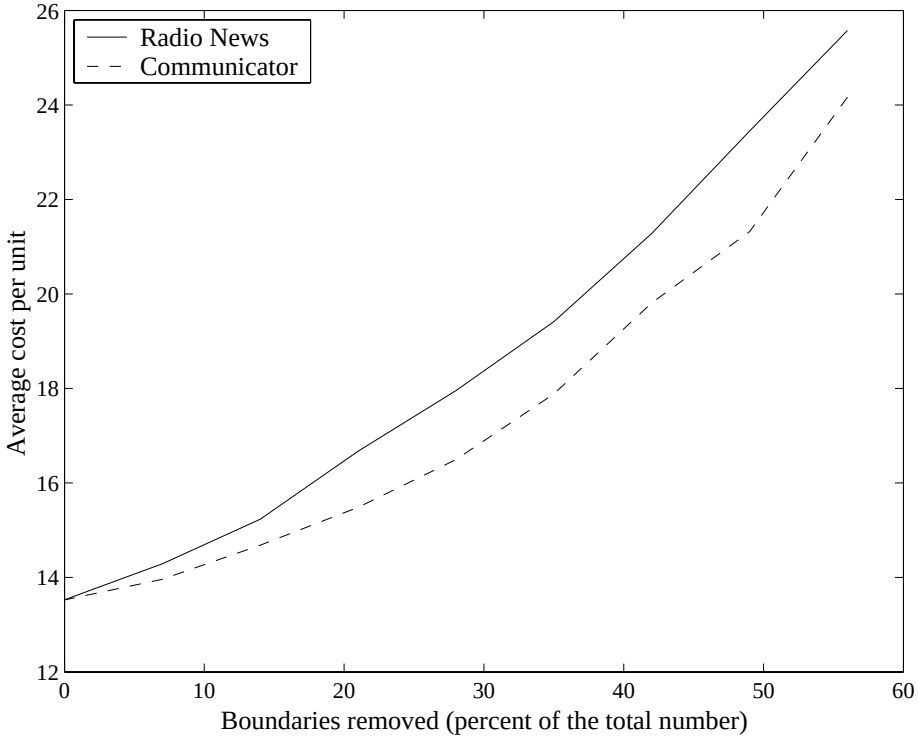

Figure 5.7: Average cost per unit in the best path, as a function of the number of boundaries made unavailable for making a splice.

normalized by the number of units) changes as we eliminate more unit boundaries. The total cost, being the combination of target, splicing and concatenation costs, is correlated with the degree of perceptual distortion in the output speech, and hence it is a good indicator of the system's performance.

From informal listening experiments we observe that the degradation in speech quality is graceful at first, but it rapidly becomes very noticeable after we remove about 40% of all potential splicing points in the unit database, boundaries which is approximately 40,000 and 130,000 boundaries for the Radio News and the Communicator corpora respectively. The Communicator corpus exhibits a slightly slower rate of increase in the average cost per unit (in particular for less aggressive pruning), compared to the rate achieved with the Radio News corpus. Such difference may be attributed to the larger size of the Communicator database as well as its limited domain of application. Limited domain corpora typically have a high degree of redundancy and therefore are less sensitive to pruning. For examples of synthesis refer to [116]. These results based on the Communicator corpus are also reported in [18].

As mentioned earlier, the goal of removing splicing points from the unit database is to reduce computational cost. We observed a near linear speedup in synthesis as we removed boundaries. In the experiment with the Communicator corpus we incorporated multiple pronunciations into the target which slowed the synthesis down to less than real time performance. However, we were able to double the speed and make it faster than real time (on a Pentium III 800MHz processor) by removing approximately 100,000 (30% of the total) splicing points. In the case when the concatenation costs are computed at run time for each pair of candidate units, one can expect a quadratic reduction in computational complexity associated with disallowing a splice to be made at specific boundaries.

As we mentioned earlier, the search tree can be pruned at run time according to target and splicing costs prior to evaluating all possible concatenations. In Figure 5.8 we compare the performance of off-line and on-line search pruning in terms of

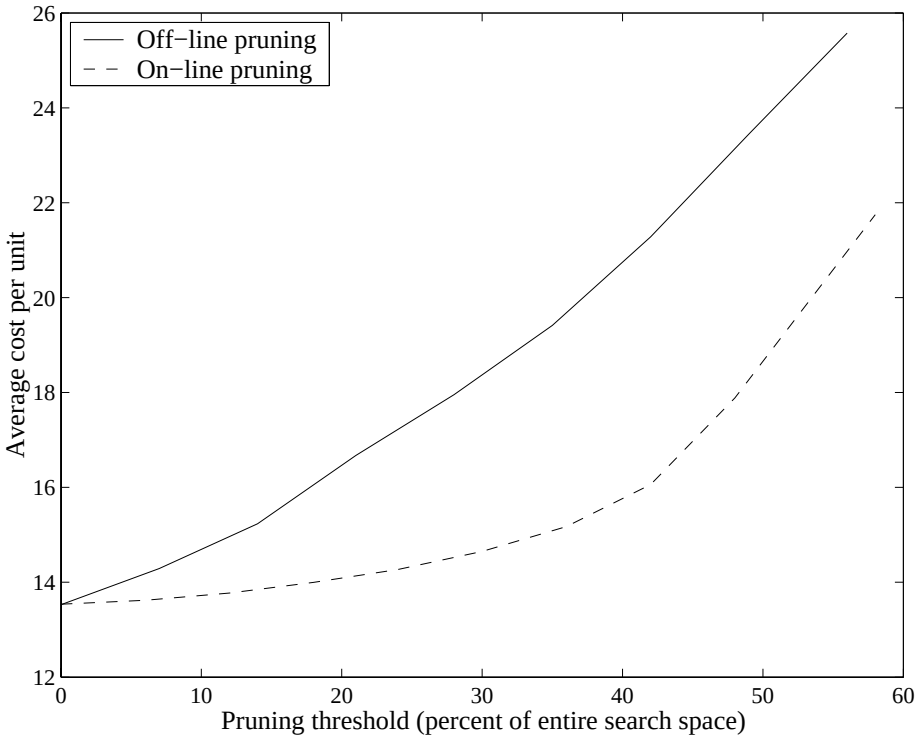

Figure 5.8: Comparison of off-line and on-line search pruning in terms of the average cost per unit in the final path (Radio News).

the average cost per unit in the final path. In the off-line case, the pruning threshold controls the number of potential splicing points. In on-line pruning, we choose a beam width that constrains the number of candidate paths to be some percentage of the total possible. We see that with on-line pruning we are able to find lower cost paths. Off-line pruning, on the other hand, uses less memory since there is no need to save splicing costs for all units, and may offer faster execution since the scores are not sorted for every candidate cluster.

5.6 Summary

In this chapter we have presented several new methods designed to improve the efficiency of unit selection from large databases and to increase the output quality. In particular, we modify candidate clusters so that units can be shared between several clusters. In conjunction with pruning this technique can reduce the impurity of clusters, and, consequently, the expected target cost, at no extra cost in terms of run-time computation. We also introduce splicing costs as an acoustically derived measure to indicate which unit boundaries are particularly good or poor joint points. Splicing costs extend the flexibility offered by the unit selection paradigm by allowing a more efficient dynamic search optimization. Through a perceptual experiment we demonstrate an improvement in speech quality achieved by using splicing costs during unit selection. This chapter also explores the applications of Buried Markov Models (BMM) to speech synthesis. We show that BMMs are more efficient than HMMs as a synthesis model, and focus on using BMM dependencies for computing splicing costs. Finally, we show how by constraining the splicing points (off-line) or by pruning the space of candidates (on-line) the computational complexity of the dynamic search can be reduced by nearly a half with a negligible loss in synthesis quality.

Chapter 6

SYSTEM IMPLEMENTATION WITH WFSTS

This chapter provides details on how a speech synthesis system can be efficiently implemented with WFSTs. Here we have taken the concatenative approach to waveform generation using subword units of variable size selected from a large unit database (see Section 2.4 and Chapter 5). As it was stated earlier, this work focuses on improving speech synthesis quality by taking advantage of the flexibility that natural languages offer. In particular, we exploit the variability in prosody and word pronunciations. We also aim to achieve high computational efficiency of the system in order to compensate for the additional complexity due to alternative prosodic targets and pronunciations.

In this work we assume that text analysis[1] either has already been performed or is unnecessary as in the case of limited domain synthesis, where the input is produced by a text generator. Hence the input to our synthesizer has all of the information necessary to disambiguate the text and may even include prosodic specifications derived from templates.

We use strictly symbolic markers (derived from ToBI) to specify prosody in the target. These markers are predicted for each word in the target and hence prosody prediction takes place prior to converting words into phonemes (unlike shown in Figure 2.1). We also eliminate the step of continuous parameter (i.e. detailed F0 and segmental durations) specification, in part because continuous prosodic parameters are incorporated into the decision tree clustering criteria and, hence, predicted im-

[1]For an in-depth coverage of text analysis with WFSTs, refer to [89].

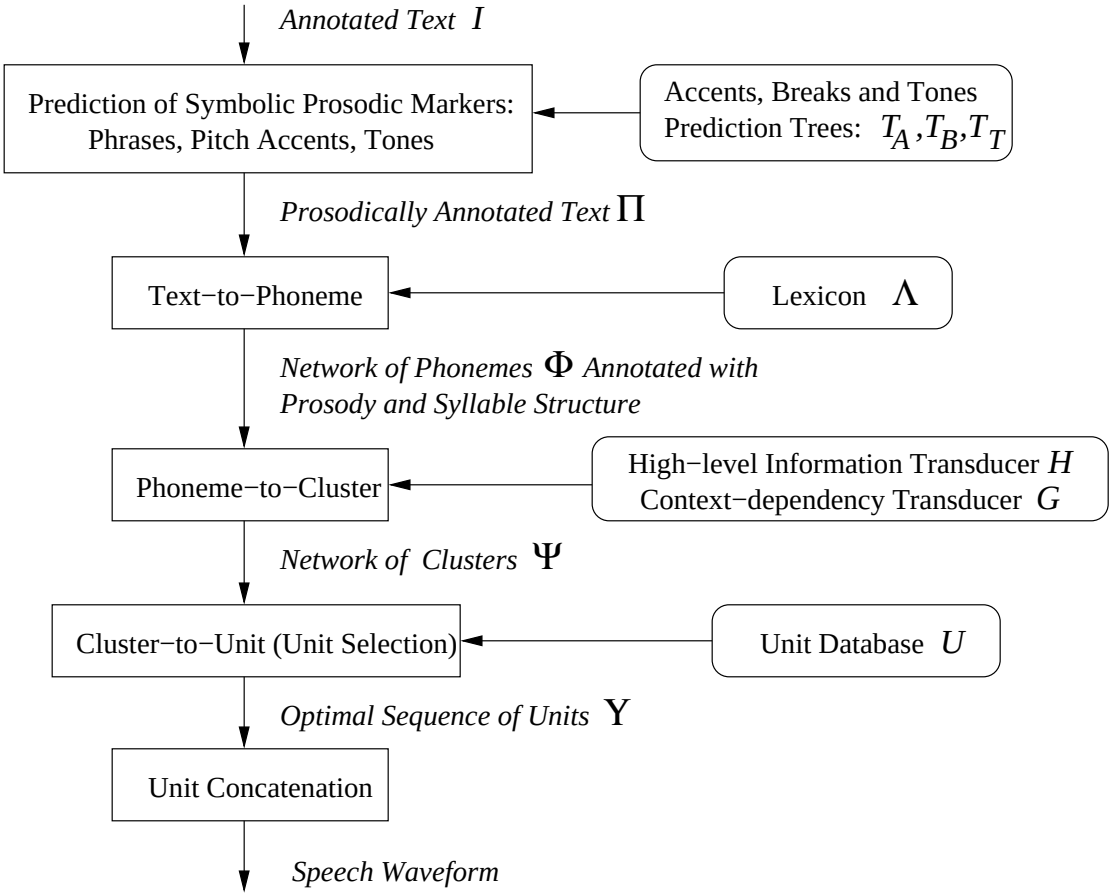

Figure 6.1: Internal structure of our subword-unit-based synthesizer. Symbols denote individual WFST components.

plicitly (see Section 5.2), and in part because the limited domain makes this strategy reasonable.

The structure of our synthesizer is summarized in Figure 6.1. A text network annotated with information necessary for prosody prediction is input to the synthesizer. The network is expanded, adding symbolic prosodic markers for each word in the input. Then words are converted into their lexical representation, which includes pronunciations and syllable structure. The resulting phonemes are transformed into context-dependent clusters based on their prosodic and lexical properties. Finally,

the context-dependent clusters of candidate units are searched for the best matching sequence of units, and these units are concatenated to produce a speech waveform. Sections 6.1 and 6.2 will describe our approach to implementing all of these modules (with the exception of unit concatenation) using WFSTs. Aspects of the synthesizer's performance will be addressed in Section 6.3. Throughout the chapter, we assume some familiarity with WFST terminology and procedures; for a review, see Section 2.8.

6.1 Flexible Target Specification

6.1.1 Prosodic Variability

The key to capturing prosodic variability is to predict a distribution of prosodic labels instead of selecting the most probable one. This can be accomplished with a variety of stochastic models. Decision trees, one of the most frequently used methods for prosody prediction, has been typically implemented as a deterministic predictor, containing only the most probable value at each leaf node. In general, leaf nodes can contain complete distributions of the random variable, describing the variability observed in the training data.

The input to our synthesizer I is normalized text annotated with information pertinent to prosody prediction. Figure 6.2 provides an example of an input utterance "to Boston" represented as an FSM. Each word is annotated with prosody-related labels α_i (part-of-speech tags, syntactic and semantic labels, punctuation, etc.). Each sequence of K labels $\alpha_1 \ldots \alpha_K$ is then converted into a network of prosodic markers P_i by composing the input I with the WFST T derived from the prosody prediction decision tree. In general, each sequence of labels α_i can be represented by one multi-dimensional label. A sequence of labels, however, allows the decision tree transducers T to be constructed with greater efficiency, in particular when several decision trees are used to predict various prosodic markers – the approach we took in this work. A

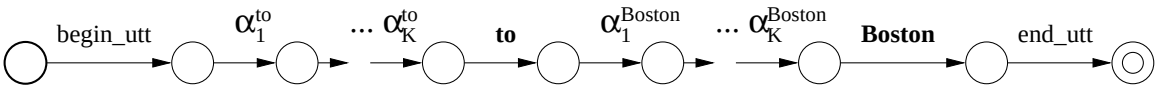

Figure 6.2: Input (I) phrase "to Boston" annotated with labels α_i.

decision tree can be represented as a WFST via a straightforward mapping of each input label sequence $\alpha_1 \dots \alpha_K$ for each trace in the tree, or by means of more sophisticated methods such as the one described in [92]. For simplicity let's assume that a complete prosodic target is specified by one discrete variable P of cardinality $|P| = N$, but in general it may be represented as a sequence of prosodic markers.

We incorporate three types of prosodic markers: pitch accents, breaks, and boundary tones, into the target specification Π. We also train individual decision trees for each of these markers. Consequently, the transducers T_A, T_B, and T_T derived from the accent, break, and boundary tone decision trees respectively, must be applied to the input I in stages. The order and type of the transducer application is subject to the dependencies between these prosodic markers. For example, if the location of breaks depends on the presence of pitch accents, and if boundary tones depend on both pitch accents and breaks, then prosody prediction would entail cascading the transducers T_A, T_B, and T_T by means of composition, removing the input labels[2] α with transducer R_α, taking the right projection π_2 and removing ε-transitions ρ_ε:

$$\Pi = \rho_\varepsilon(\pi_2[I \circ T_A \circ T_B \circ T_T \circ R_\alpha]).$$

Alternatively, intersection operation is appropriate to combine transducers when prosodic markers are independent. For example, if breaks and tones are independent of accents then prosody network Π is obtained as follows:

$$\Pi = \rho_\varepsilon(\pi_2[(I \circ T_A \circ R_\alpha) \bigcap (I \circ T_B \circ T_T \circ R_\alpha)]).$$

[2]This functionality can be incorporated in T_T, but here it is made explicit to show that the removal of α's is the last step in this operation since each tree in the sequence requires α's in its input.

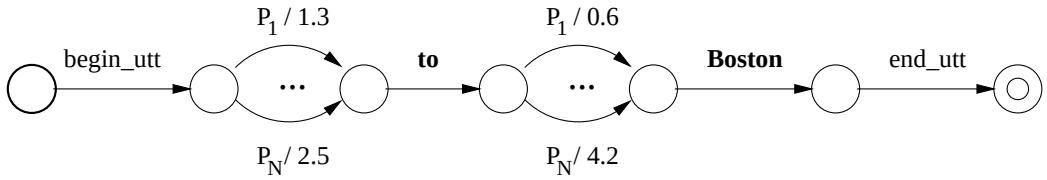

Figure 6.3: An example of prosody-annotated network of words (Π).

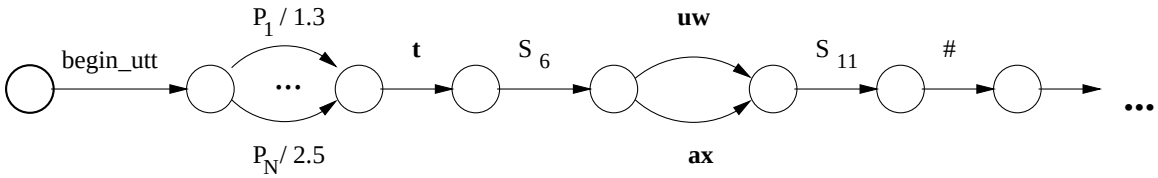

Figure 6.4: A fragment of the network Φ containing a phonetic representation of the word "to".

The form of prosody-annotated network Π is illustrated in Figure 6.3. Note that each word is preceded by a network of prosodic targets weighted according to the negative log probabilities obtained from the decision trees.

6.1.2 Using Alternative Pronunciations

A typical approach to word-to-phoneme conversion is to map the word into its canonical pronunciation with possible contextual adjustments. Constraining the search to a single phone sequence limits the synthesizer's ability for achieving the highest naturalness of output. Speech databases, especially in limited domain applications, do not always provide a "perfect" balance of units. Some sounds may be recorded in a wider variety of context than others due to the nature of the application or the dialectal specifics of the target speaker. A non-canonical (but generally accepted) alternative pronunciation may result in a better match from the database in terms of smoothness of concatenation points as well as the proximity to the prosodic target.

The phonetic representation Φ of the target (shown in Figure 6.4) can be obtained

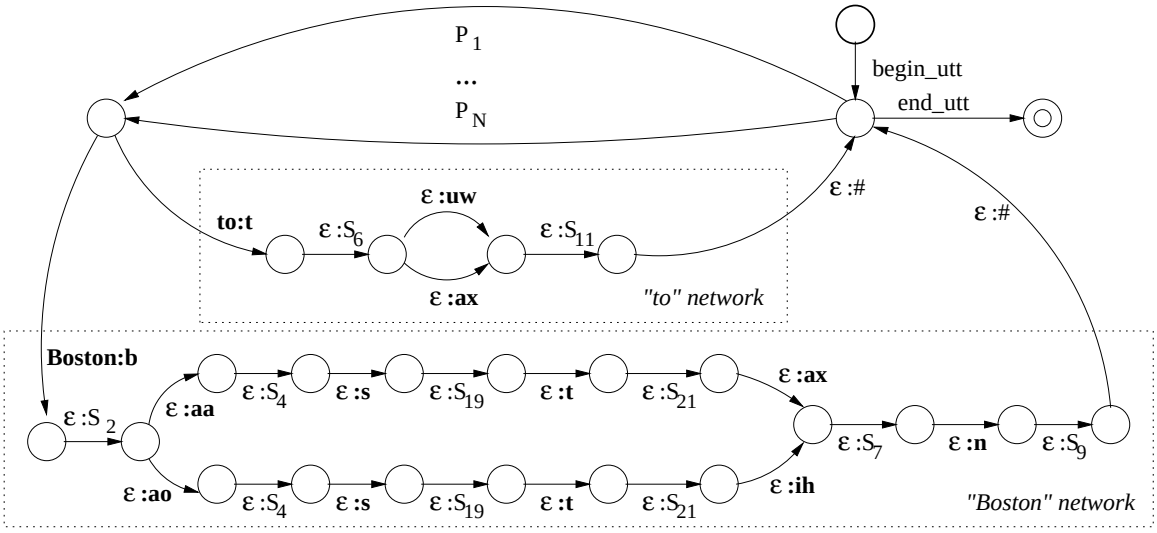

Figure 6.5: A fragment of the lexicon Λ, containing words "to" and "Boston". The output labels S_i are hypothetical indices to syllable structure tags. Empty string ε is used as input in order to insert additional output labels.

by composing the prosody-annotated network Π (such as the one shown in Figure 6.3) with the lexicon Λ and taking the right projection:

$$\Phi = \pi_2(\Pi \circ \Lambda).$$

The task of the lexicon is to generate phonetic labels annotated with syllable information S_i, and to preserve the prosodic markers P_i, which can later be used for mapping the phonemes into context-dependent phones. Again, for illustration purposes, we will assume that all relevant syllable information can be represented by a single tag S_i (analogous to prosodic markers P_i), but in general, multiple tags may be useful (as for the input labels α_i in Figure 6.2). The lexicon also inserts a special symbol # at the end of each word.

We design the lexicon to include alternative pronunciations, for example, *[t uw]* and *[t ax]* for the word "to", and *[b aa s t ax n]* and *[b ao s t ih n]* for the word "Boston". A fragment of the lexicon containing words "to" and "Boston" is shown in Figure 6.5.

6.2 Efficient Unit Selection

6.2.1 Modeling Context Dependency with WFSTs

The network of phonemes Φ annotated with prosodic and syllabic information must be transformed into a network of context-dependent phones which identify clusters of candidate units grouped according to the decision tree clustering procedure described in Section 5.2. A single WFST capable of handling all possible prosodic, syllabic and phonetic contexts may, however, be prohibitively large. Hence, we split this process into two stages. First, high level information (prosody P and syllable S) is processed by transducer H and then appropriate phone labels are chosen according to the left and right neighboring phonemes by transducer G, as follows:

$$\Psi = \rho_{\varepsilon}(\pi_2[\Phi \circ H \circ G]).$$

Shown in Figure 6.6 is the transducer H which translates prosody P_i and syllable S_i information into tags C_i that specify combined syllabic and prosodic context in detail sufficient for distinguishing between unit clusters of a given phoneme according to the decision tree that was used to cluster the units. This step constrains the number of possible values that C can take to be not more than the number of unit clusters. Also note that word boundary symbols # are removed, and some are mapped into pauses, depending on the prosodic context P.

The second step involves mapping the high level context tags C_i into phone cluster labels for each immediate left and right phonetic context using context-dependency transducer G. This can be accomplished by a transducer such as the one shown in Figure 6.7, where each state corresponds to a phoneme pair, and an arc connecting two states represents a triphone. So for instance an arc connecting states “t-uw” and “uw-b” corresponds to a triphone “t-uw-b”.

The transducers H and G together essentially implement the process of walking through the decision tree to a leaf node (cluster index). These transducers can be de-

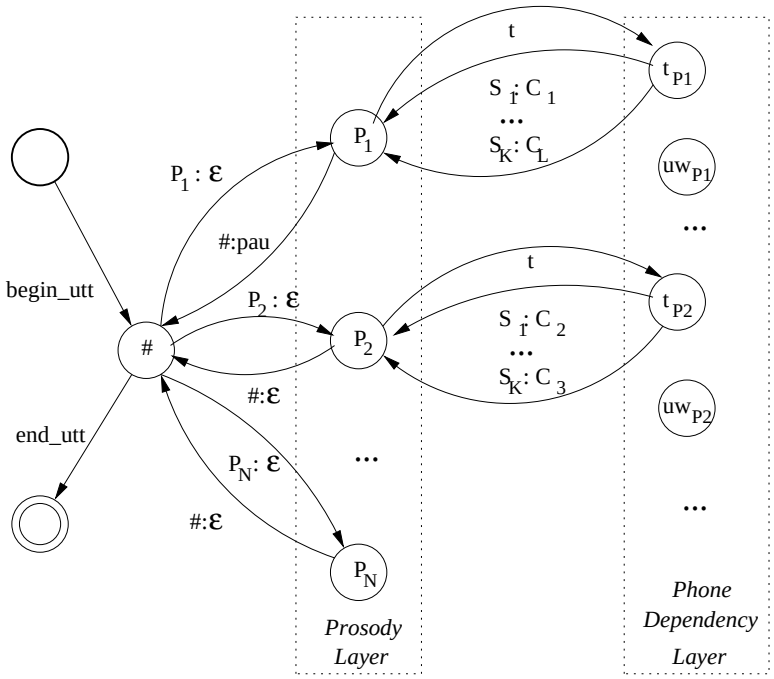

Figure 6.6: High-level context information transducer H.

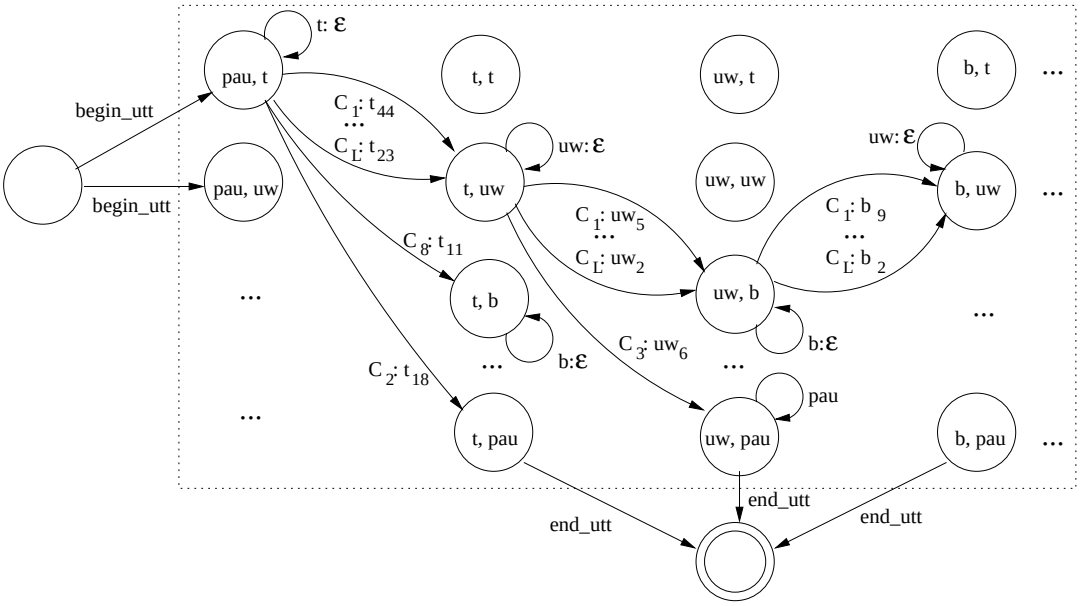

Figure 6.7: Context-dependency transducer G.

rived from the decision trees automatically. First, all leaves are categorized according to the prosodic and syllabic context, and each unique context, which may include several tree leaves, receives a label C_i, and is reflected in the high-level context transducer H. The context-dependency transducer G is built by examining phonetic context for each leaf and combining it with the high-level context.

The structure of the output network Ψ can be described by the regular expression

$$begin\_utt \cdot \psi^* \cdot end\_utt,$$

where ψ is the alphabet containing all of the candidate cluster indices and the symbol *pau* which corresponds to a short silence.

6.2.2 Unit Selection by WFST Composition

With all sequences of candidate clusters represented in the finite state network Ψ, the unit selection process, i.e. dynamic search over the space of candidate units (see Sections 2.4 and 5.4), can be accomplished via finite-state operations provided that the unit database is represented in the form of a WFST. This Section will describe our approach to unit selection with WFSTs.

At a glance, unit selection is performed with the following operation:

$$\Upsilon = \rho_\varepsilon(\pi_2(BestPath(\Psi \circ X \circ U))).$$

The input network of clusters Ψ is composed with an extension WFST X, which adds auxiliary arcs τ representing potential splicing points after each cluster target ψ, resulting in a network, described by the regular expression

$$begin\_utt \cdot pau \cdot \tau \cdot (\psi \cdot \tau)^* \cdot pau \cdot end\_utt.$$

This network is then composed with the unit database WFST U, which maps cluster targets ψ into unit indices *uid*. Then we find the best path through the network of

candidate units, take the right projection π_2 and remove ε-transitions ρ_ε, obtaining the desired sequence of units.

As it was mentioned earlier, we have taken the vector quantization approach to reduce the complexity associated with computing the concatenation costs. Frames at unit boundaries were vector quantized using a 256-vector codebook. The frames were 5 msec wide and contained line spectral frequencies (LSF) of order 18, energy, and F0. Concatenation costs were then computed between each pair of VQ codebook entries. Vector quantization allows concatenation costs to be treated independently of the unit identity and to be represented in the form of a fixed-size fully interconnected WFST, with the number of states equal to the number of codebook entries. Each link transduces a special symbol τ into an empty string ε. An example of a small VQ concatenation network is illustrated in the bottom of Figure 6.8 representing a codebook of size 4. Note that each node has a self loop with a non-zero cost $C(i, i)$ which we define to be equal to the mean distance between all pairs of frames that are mapped into the i-th entry in the codebook. In our implementation two units have a zero concatenation cost only if they occur adjacent in the original recording.

Each unit has a corresponding state in the WFST. In Figure 6.8 these states are grouped into clusters, representing the leafs of the decision trees that we used to cluster the candidate units. Thus $Cluster_1$ contains all candidate units for the target specification t_{44}. Each state in these clusters has a self loop that carries out a transduction of a given target ψ to a unit ID uid uniquely specifying the unit in the database. The cost of this transduction is given by the target cost $T(uid)$, which we define to be the distance between the unit and its cluster's mean.

Each unit state has two links, incoming and outgoing, that connect it to the VQ concatenation network, where corresponding states represent frames at the unit's left and right boundaries respectively. The costs assigned to these links $S\_left(uid)$ and $S\_right(uid)$ are the left and right splicing costs for the unit (see Section 5.4).

If two units are adjacent (i.e. have a common boundary) in the original recording,

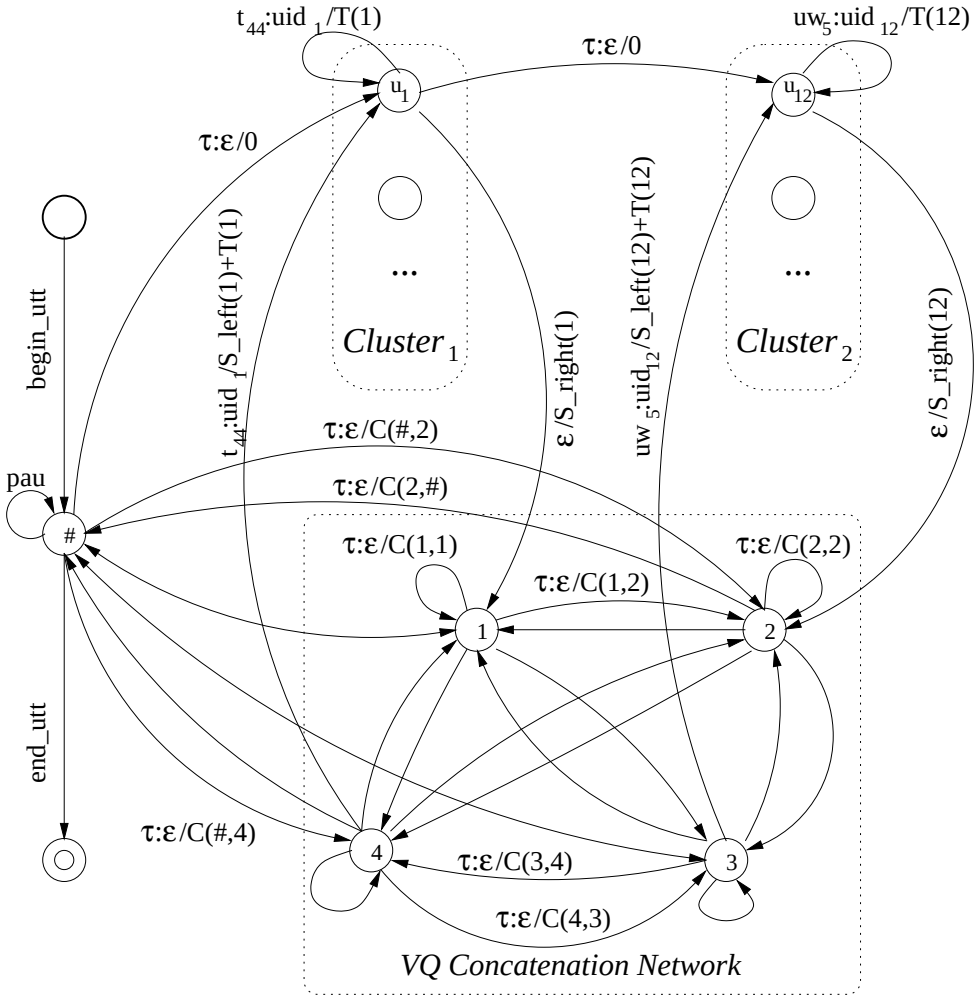

Figure 6.8: WFST implementation of the unit selection module U. Shown are two unit clusters and a VQ concatenation network of size 4. Costs are labeled as follows: $T(u)$ is the target cost for unit u; $C(i, j)$ is the concatenation cost between vector quantized frames i and j; $C(\#, i)$ and $C(i, \#)$ costs of transitioning from silence to VQ frame i and vice versa; $S\_left(u)$ and $S\_right(u)$ are left and right splicing costs for unit u.

then the concatenation and splicing costs between them are zero, which is implemented by having a direct link from the left unit to its right neighbor with a zero concatenation cost, as shown in Figure 6.8 between units uid_1 and uid_{12}.

The unit database WFST has a state # representing silence. It is connected to the VQ concatenation network with arcs carrying costs $C(\#, i)$ and $C(i, \#)$ that measure smoothness of transitioning from silence to a given VQ frame i and vice versa, respectively. These costs are computed as concatenation costs between the corresponding VQ frame and the mean of all frames labeled as silence in the recorded utterances. The state # is also connected to some "unit" states with zero-cost arcs in cases where the unit is preceded (or followed) by silence in the original recording. The network has a start and an exit states also connected to the state #.

Normalization and/or relative scaling of costs in the WFST is a design issue, since the costs are not equally important and the different types of costs can have different average values. In practice, concatenation costs are typically scaled higher than target costs to compensate for higher human sensitivity to smoothness of speech [11]. Through informal listening we found that scaling concatenation and splicing costs such that their means are 10 times larger than the mean target cost gives the best performance.

6.3 Performance Considerations

The computational performance of our system strongly depends on the design of individual WFST components. The steps of prosody prediction and text-to-phoneme conversion expand the network to include alternative sequences associated with the variability in prosody and pronunciations. This additional flexibility translates directly into increased computational complexity at each following step in the process. So, perhaps, limiting the prosodic alternatives to several most likely options may be necessary for acceptable performance.

Table 6.1: Notation used to define sizes of certain WFSTs.

K	Number of syllable structure labels (S_i)	1,000
L	Number of high-level context labels (C_i)	20
M	Number of phonemes (ϕ)	50
N	Number of prosody labels (P_i)	20
B	Average cluster size	25
D	Unit database size	25,000
V	VQ codebook size	256

The step of converting phonemes to context-dependent clusters, on the other hand, corresponds to a "many-to-one" mapping and reduces the size of the network. However, it involves design of complex transducers to model context dependency. As described in Section 6.2.1, we perform phoneme-to-cluster conversion in two steps. Transducer H shown in Figure 6.6 translates high-level information such as prosody P and syllable S structure into phoneme-dependent context tags that only retain details sufficient to distinguish between clusters. This step is designed to reduce the space of possible high-level context labels C_i, which makes the full context-dependency transducer G (see Figure 6.7) smaller.

For example, let us consider the features listed in Table 5.1. There are 4 categories for pitch accent, 5 categories for boundary tone and 2 categories for phrase breaks. Given that boundary tones only occur at major phrase boundaries, tones and breaks can be combined into one feature with 5 possible values. This results in $N = |P| = 4 \times 5 = 20$ (listed in Table 6.1) distinct specifications that completely describe prosody of a given word. As for the information describing syllable structure S, this number is much higher because there are many more features related to syllable information in Table 5.1. However, there is a high degree of redundancy among these features. By

means of hand-set rules[3], we were able to represent the entire space of these features by 10^3 distinct values $K = |S| = 10^3$ (compared to more than 10^5 combinations if the features were independent). We used a phoneset of 50 phonemes $M = |\Phi| = 50$. Hence, the size of transducer H is approximately $N + N \times M \approx 10^3$ states and $N \times (K + 1) \times M + 2N \approx 10^6$ arcs.

As mentioned earlier, the compressed context information C output by transducer G has only as much detail as needed to distinguish between unit clusters. Therefore, the number of values that the variable C can take is bounded by the number of clusters for a given phoneme. On average we have 20 clusters per phoneme, so in the worst case scenario, $L = |C| = 20$. The size of network G is approximately $M^2 = 2.5 \times 10^3$ states and $M^3 \times L = 2.5 \times 10^6$ arcs. If we did not have high-level information pre-processed by transducer H, the size of G would have been as much as 1000 times larger.

Note that transducer G is *non-deterministic*, i.e. there exist one or more states, such that there are more than one arc with the same input label leaving the state. To be more precise, each of the states representing pairs of phonemes has a set of L arcs with input labels $C_1, \ldots, C_L = \mathcal{C}$, each connected to M other states. Deterministic transducers (such that at each state all outgoing arcs carry unique input labels) are more efficient to perform operations on because there is no ambiguity at any step during graph traversal. As mentioned earlier, the phoneme-to-cluster conversion is unambiguous as there is one and only one cluster for a given context. Such an operation can be performed by a deterministic transducer. Automatic determinization of WFSTs can be very complex and may not always be possible [57, 58]. However, with a slight change in design of transducer G, we can make it deterministic (denoted G_d).

In the network obtained by processing high-level context information $\Phi \circ H$, we

[3]These rules can be learned automatically given sufficient training data.

map context-independent phoneme labels $\phi \in \mathcal{F}$ into phoneme pairs $\phi_1\_\phi_2$ as they appear in the target sequence, and then compose it with the deterministic context-dependency transducer G_d as follows:

$$\Psi = \rho_\varepsilon(\pi_2[\Phi \circ H \circ F_1 \circ F_2 \circ F_3 \circ G_d]),$$

so effectively non-deterministic G is replaced with $F_1 \circ F_2 \circ F_3 \circ G_d$ where each component is deterministic. This operation is performed in three intermediate steps in order to simplify the WFST components F_1, F_2 and F_3. Transducer F_1 inserts "silence-to-phoneme" transitions before each phoneme that follows a silence. Its functionality is defined by the following regular relation:

$$F_1 \models pau \cdot \phi \rightarrow pau \cdot pau\_\phi \cdot \phi, \ \forall \phi \in \mathcal{F},$$

where $\models$ denotes the transducer's definition. The phoneme and pause labels are then duplicated by F_2 as follows:

$$F_2 \models C \cdot \phi \rightarrow \phi \cdot C \cdot \phi, \ \forall C \in \mathcal{C}, \ \forall \phi \in \mathcal{F} \cup \{pau\}$$

in order to simplify the step of mapping two consecutive phonemes into a corresponding pair, accomplished by transducer F_3:

$$F_3 \models \phi_1 \cdot \phi_2 \rightarrow \phi_1\_\phi_2, \ \forall \phi_1 \in \mathcal{F}, \ \forall \phi_2 \in \mathcal{F} \cup \{pau\}.$$

Consider the design for a deterministic context-dependency transducer G_d illustrated in Figure 6.9. Each state corresponds to a phoneme with its left and right phonetic context. It is unnecessary to have a separate state for each individual triphone; contexts can be grouped in the same way that units are grouped into clusters by the decision trees. Hence the states only need to represent the variety of phonetic context observed in the tree leafs. Such network results in at most $M \times (L+1) \approx 10^3$ states – 2.5 times fewer states than in the non-deterministic network shown in Figure 6.7. Each state has up to L self-loops converting high-level context tags C_i into

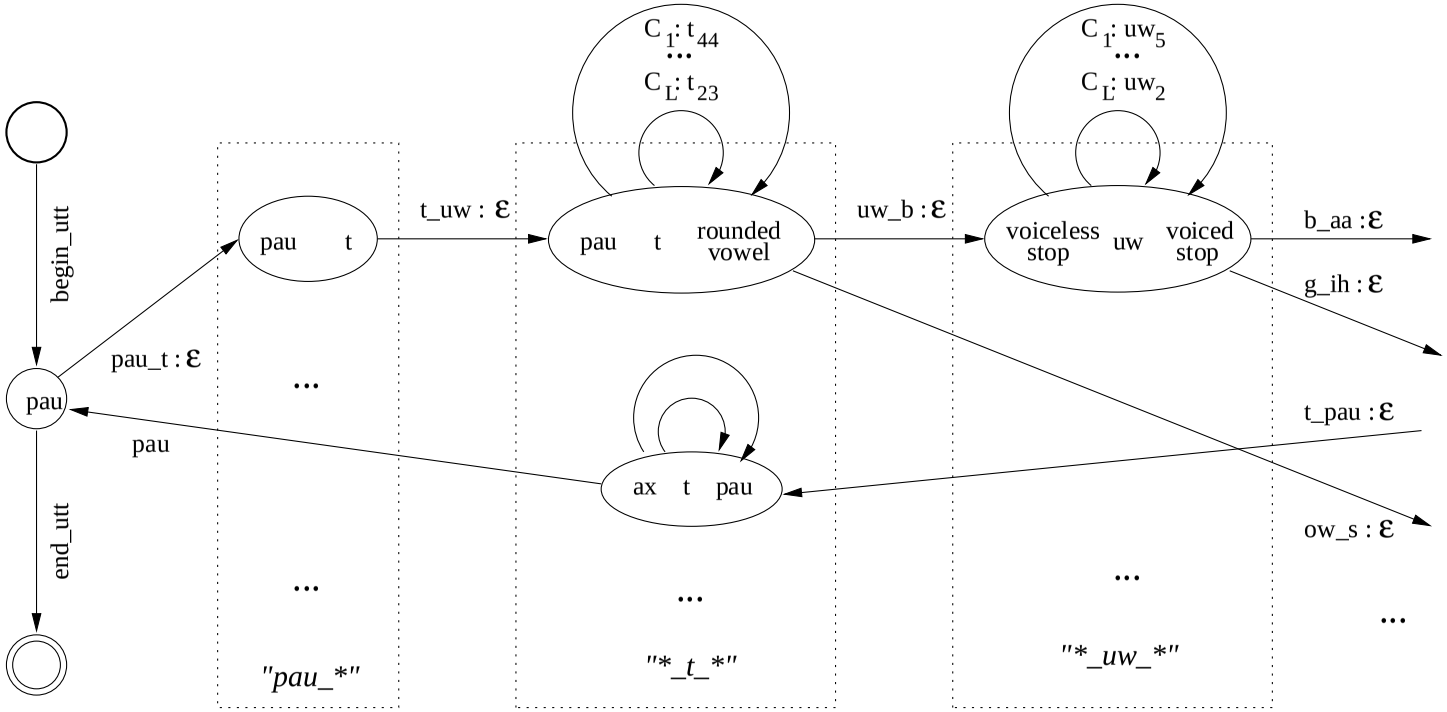

Figure 6.9: Deterministic context-dependency transducer N_d for combining high level context C_i and phonetic context to obtain decision tree clusters ω_i, where ω is the target phone.

cluster indices. There are also up to M^2 arcs leaving each state connecting it to other states according to the phonetic context. In total, the network contains up to $M \times L \times (L + M^2) \approx 2.5 \times 10^6$ arcs – the same number of arcs as in the non-deterministic network. The main advantage of using the deterministic WFST is that it is M (50) times faster.

The final transducer in the chain, the unit database WFST U, maps cluster labels (i.e. context-dependent phones) into database unit indices. By its nature, phone-to-unit conversion corresponds to a one-to-many relation which makes transducer U non-determinizable. The VQ concatenation network is responsible for a significant part of non-determinism in transducer U. Nevertheless, the VQ concatenation network makes it feasible to represent the unit selection WFST U for large unit databases by constraining the size of the WFST to grow linearly with the number of units in the database. Let us denote $D = 25,000$ to be the number of units in the database, $B = 25$ is the average cluster size, and $V = 256$ is the number of states in the VQ concatenation network. The network U shown in Figure 6.8 has $D + V \approx 2.5 \times 10^4$ states and $2 \times V^2 + 4 \times D \approx 2.3 \times 10^5$ arcs. Because of the non-determinism in the VQ concatenation network, if we were to implement WFST U by connecting units directly to each other and computing concatenation costs for each individual pair of units, then such a WFST would be at least $V/B \approx 10$ times faster, but it would contain more than $D^2 = 6.25 \times 10^8$ arcs, making it of little practical value.

The network U shown in Figure 6.8 differs slightly from what we have proposed earlier in [17]. With the "silence" state # bidirectionally connected to the VQ concatenation network we now can synthesize arbitrarily long utterances. We also eliminated the special symbols *c_in* and *c_out* assigned to the left and right splicing arcs respectively. Instead, the left splicing arcs perform the cluster-to-unit transduction with the cost $S\_left(i) + T(i)$, i.e. combined left splicing and target costs. This modification significantly reduces non-determinism of WFST U and makes its application much faster.

Chapter 7

CONCLUSION

In this chapter we conclude by first summarizing the main contributions of this work, then stating some of the questions that were raised but remained unanswered throughout the thesis, and outlining some future directions for research in dialog systems and concatenative speech synthesis.

7.1 Summary of Results and Contributions

We have developed a methodology for designing a speech synthesis system that takes into account the variability of natural speech, i.e. the fact that two (or more) acoustically different utterances can convey essentially the same meaning. We showed that natural variability in the choice of words and prosodic realizations can be put to the system's advantage, when all of these factors are optimized jointly as part of the unit selection process. Instead of specifying a single desired utterance, we effectively make a "soft" decision about the word sequence and prosodic target and let the synthesizer select the utterance that will sound most natural. In informal listening experiments, we also found that that allowing pronunciation variability is useful.

We have proposed a framework for an efficient integration of language generation and speech synthesis in spoken dialog systems. Through perceptual experiments we have demonstrated that by expanding the space of candidate responses with alternative word sequences and multiple prosodic targets we can achieve higher quality speech output. These results were obtained with a dialog system that uses template-based language generation, but our approach is general and can be applied to other

mechanisms for language generation and is particularly well suited to probabilistic methods. The latter claim has a direct implication for the field of natural language generation as it provides new motivation for stochastic generators.

In order to compensate for the increased search complexity associated with the input flexibility and joint optimization, we have developed several new methods aimed at improving the efficiency of unit selection from large speech databases. In particular, we proposed a variation of the cluster-based unit database design that allows units to be shared among multiple clusters. We also developed a methodology for automatically estimating the suitability of a splice to be made at a given boundary prior to performing the unit selection search, which allows the search space to be pruned more efficiently both at run time and off-line. The approach uses a model of speech dynamics to formulate concatenation and splicing costs, building on the hypothesis that concatenation costs should be inversely related to predictability. In addition to improved efficiency, these methods offer improved output quality, as we have established with perceptual experiments.

The implementation of our speech synthesis system, based on the weighted finite-state transducer formalism, offers a compact representation of flexible targets and allows the dynamic search to be performed efficiently using a general purpose toolbox. This work is the first to implement a complete synthesis system by representing each module with WFSTs. Our system supports a large database of subword units and can be used for limited domain synthesis as well as unrestricted text-to-speech (when combined with a text analysis module). We provide a design for the prosody prediction module which integrates prosody templates with a more general word-based predictor. We also provide a new design for the unit selection module, offering the ability to have unit-specific concatenation costs – an improvement over a WFST architecture previously proposed in [111].

This work demonstrates a new approach to integrating language generation and speech synthesis components in dialog system, which can significantly improve the

output quality by taking advantage of the variability in natural speech. While our experiments focus largely on limited domain applications, the proposed system architecture supports selection of subword units of variable size from a large unit database, and hence it is generalizable to unrestricted text-to-speech.

7.2 Open Questions

While this work resulted in several contributions related to speech generation system architectures and design, there are several questions that have been raised or left unanswered.

In our experiments, we did not apply any prosodic modification or spectral smoothing to the output speech signal. While this approach is reasonable in the case of word-level concatenation in limited domain applications, artifacts at the concatenation points become more noticeable when subword units are concatenated in less constrained domains or unrestricted TTS. While the improved flexibility of our system and the use of splicing costs help finding units that closely match the target, some limited prosodic modification can improve the output for the cases when the target match is poor. A key question is whether the benefits of these techniques remain unchanged when the speech signal is modified. One might conjecture that the improvements to naturalness would be much smaller, since the distortion associated with poor joins would be lower. Conversely, one might anticipate that the computational savings would be greater since the search space could be pruned more aggressively. Perceptual experiments are required to answer this question.

Our approach to computing splicing costs is based on acoustic features and therefore is completely automatic. In our experiments, we showed that using splicing costs offered improved performance relative to not using them, but we did not provide explicit evidence to support our conjecture that concatenation and splicing costs should be inversely proportional to log probability. Also, it is possible that other features

may contribute to the robustness of splicing costs, including both automatically extracted parametric representations (MFCC, perceptually modified cepstra [28], etc.), and/or knowledge-based linguistic properties that can be applied to context clustering or weighting specific costs. Again, perceptual experiments are needed to establish the best approach.

In experiments with speech generated for a constrained domain, we observed that users have individual preferences to the alternative word sequences offered in the template. The system can be made adaptive to the user if the templates are weighted according to user preferences. It would be interesting to study these preferences on a large set of users.

An overlap between the fields of speech recognition and speech synthesis offers a great potential for combining technologies in these fields. Working with speech synthesis we have experienced the benefits of borrowing techniques from the speech recognition field. Whether the lessons learned designing a synthesis system can feedback to ASR remains an open question.

7.3 Use of Hand-Labeled Data

A carefully prepared speech corpus is crucial for the performance of a speech synthesis system. It is, however, important to be able to do most of this preparation (i.e. segmentation, labeling, clustering, etc.) without human supervision in order to reduce development costs. While a significant amount of corpus preparation in this work is automatic (e.g. F0 and epoch computation, part-of-speech tagging, syntactic analysis, decision tree clustering) we still use hand-labeled ToBI prosody markers and some hand-corrected unit boundaries in our system. Our long-term goal is an automatically trainable system requiring little or no hand labeling. Nevertheless, use of hand-labeled data in this work reduces the scope of the effort and provides us with a baseline indicating the usefulness of symbolic prosody labels.

Unsupervised training methods can be applied to reduce prosodic labeling costs. For example, one can use a small amount of hand-labeled data to train a predictor and use it to predict labels (or probabilities of labels) on a larger corpus which could then be used as if the full data set was labeled. This approach can be extended by using co-training techniques [13], which improve performance by including multiple predictors. The probabilities of the predicted prosody labels can propagate into the target costs assigned to the units in the WFST database.

Another long term goal is the use of fully automatic phone segmentations rather than the hand segmentations used here. Currently, automatic methods of phone segmentation are not completely error free, and misalignments of unit boundaries is a major source of distortion in concatenative synthesis. It is an open question as to whether splicing costs are more or less helpful when segmentations are fully automatic. However, it is possible that the dynamic speech model that we used for splicing costs can provide a way to automatically refine HMM segmentations.

7.4 Opportunities for Further Advances

While offering advances in various types of speech synthesis applications – limited domain response generation and unrestricted text-to-speech – this work leaves room as well as offers an opportunity for further improvements.

In this work we considered only simple hand-derived wording alternatives in a template generator. It would be of interest to explore the use of joint search algorithms with stochastic language generators, which can provide a large space of word sequences, as well as non-uniform costs associated with each. Alternatively, one could investigate the possibility for learning template expansions automatically from dialogs between humans.

The response generation component of a dialog system can be improved if the information about the state in the dialog is taken into account when assigning costs

to various targets in the template. For example, if the system's response follows an error correction by a user, it may be appropriate to emphasize the information that has been corrected, in which case the template is adjusted for the word that corresponds to the corrected information to make placing a high pitch accent on this word more desirable. This feature would require designing a prosody model that captures information flow in the dialog. The Communicator corpus used in our experiments was collected by recording isolated utterances and therefore could not be used to design a dialog-sensitive prosody model.

Further research is needed on optimizing relative weighting of different types of costs. Cost normalization needs to be studied within individual WFST modules: i.e. splicing, target and concatenation costs for the unit selection module; template prosody and general prosody prediction costs for the prosody module. It may also be useful to assign individual costs to alternative word sequences and pronunciations. In addition, since the relative scaling of the costs across different modules influences the outcome of the WFST composition, it may be useful to more carefully assess these in our system.

We have demonstrated how context and the degree of coarticulation can be taken into account when computing splicing costs. Our approach uses HMM or BMM prediction accuracy as an indicator of how suitable a given boundary is for making a splice. The prediction accuracy is measured at the boundary frames, but one could also assess the prediction over a range of frames within the unit, which may give a more accurate estimate of the effects of coarticulation. In addition, one could use the residual likelihood to determine the specific join points of two units, but this would increase the computation of the concatenation cost substantially. The dynamic speech model (BMM) that we used for computing splicing costs, can also be used for computing concatenation costs or applied to perform spectral smoothing at unit boundaries by adjusting the acoustic frames to maximize the BMM likelihood.

BIBLIOGRAPHY

[1] A. Acero. Source-filter models for time-scale and pitch-scale modification of speech. In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, pages 881–884, 1998.

[2] A. Acero, 2000. Personal communication.

[3] J. Allen, S. Hunnicutt, and D. Klatt. *From text to speech: the MITalk system.* MIT Press, Cambridge, MA, 1987.

[4] J. Bachenko and E. Fitzpatrick. A computational grammar of discourse-neutral prosodic phrasing in English. *Computational Linguistics*, 163:155–170, 1990.

[5] S. Bangalore, O. Rambow, and M. Walker. Natural language generation in dialog systems. In *Proceedings of Human Language Technology Conference*, pages 67–73, 2001.

[6] M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, and A. Syrdal. The AT&T Next-Gen TTS system. In *The Proceedings of the Joint Meeting of ASA, EAA, and DAGA*, pages 18–24, Berlin, Germany, 1999.

[7] M. Beutnagel, M. Mohri, and M. Riley. Rapid unit selection from a large speech corpus for concatenative speech synthesis. In *Proceedings of Eurospeech*, volume 2, pages 607–610, 1999.

[8] J. Bilmes. Buried Markov Models for speech recognition. In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, volume 2, pages 713–716, 1999.

[9] A. Black. *CHATR, version 0.8, a generic speech synthesizer*. ATR-Interpreting Telecommunications Laboratories, Japan, March 1996.

[10] A. Black and K. Lenzo. Limited domain synthesis. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 2, pages 411–414, 2000.

[11] A. Black and P. Taylor. Automatically clustering similar units for unit selection in speech synthesis. In *Proceedings of Eurospeech*, volume 2, pages 601–604, 1997.

[12] C. Blackburn and S. Young. A self-learning predictive model of articulator movements during speech production. *Journal of the Acoustical Society of America*, 107(3):1659–1670, 2000.

[13] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

[14] A. P. Breen and P. Jackson. A phonologically motivated method of selecting non-uniform units. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 6, pages 2735–2738, 1998.

[15] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth & Brooks, Pacific Grove, CA., 1984.

[16] I. Bulyko and M. Ostendorf. Predicting gradient F0 variation: pitch range and accent prominence. In *Proceedings of Eurospeech*, volume 1, pages 1819–1822, 1999.

[17] I. Bulyko and M. Ostendorf. Unit selection for speech synthesis using splicing costs with weighted finite state transducers. In *Proceedings of Eurospeech*, volume 2, pages 987–990, 2001.

[18] I. Bulyko, M. Ostendorf, and J. Bilmes. Robust splicing costs and efficient search with BMM models for concatenative speech synthesis. In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, 2002. to appear.

[19] J. Cahn. A computational memory and processing model for prosody. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 3, pages 575–579, 1998.

[20] N. Campbell. Multi-lingual concatenative speech synthesis. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 7, pages 2835–2838, 1998.

[21] M. Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of ACL*, pages 184–191, 1996.

[22] G. Coorman, J. Fackrell, P. Rutten, and B. Van Coile. Segment selection in the L&H Realspeak laboratory TTS system. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 2, pages 395–398, 2000.

[23] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[24] A. Cronk and M. Macon. Optimized stopping criteria for tree-based unit selection in concatenative synthesis. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, pages 680–683, 1998.

[25] W. Ding, K. Fujisawa, and N. Campbell. Improving speech synthesis of CHATR using a perceptual discontinuity function and constraints of prosodic modification. In *the* 3^{rd} *ESCA/COCOSDA International Workshop on Speech Synthesis*, pages 191–194, 1998.

[26] R. Donovan. *Trainable speech synthesis*. PhD thesis, Cambridge University, 1996.

[27] R. E. Donovan. Segment preselection in decision-tree based speech synthesis systems. In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, volume 2, pages 937–940, 2000.

[28] R. E. Donovan. A new distance measure for costing spectral discontinuities in concatenative speech synthesizers. In *Proceedings of ESCA*, 2001.

[29] R. E. Donovan, M. Franz, J. S Sorensen, and S. Roukos. Phrase splicing and variable substitution using the IBM trainable speech synthesis system. In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, volume 1, pages 373–376, 1999.

[30] R. E. Donovan and P. C. Woodland. Automatic speech synthesizer parameter estimation using HMMs. In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, pages 640–643, 1995.

[31] R. E. Donovan and P. C. Woodland. A hidden Markov-model-based trainable speech synthesizer. *Computer Speech and Language*, 13(3):223–241, 1999.

[32] M. Eichner, M. Wolff, S. Ohnewald, and R. Hoffmann. Speech synthesis using stochastic Markov graphs. In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, volume 2, pages 829–832, 2001.

[33] C. Fordyce and M. Ostendorf. Prosody prediction for speech synthesis using transformational rule-based learning. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 3, pages 843–846, 1998.

[34] T. Fukada, K. Tokuda, T. Kobayashi, and S. Imai. An adaptive algorithm for mel-cepstral analysis of speech. In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, volume 1, pages 137–140, 1992.

[35] M. Galey, E. Fosler-Lussier, and A. Potamianos. Hybrid natural language generation for spoken dialog systems. In *Proceedings of Eurospeech*, volume 3, pages 1735–1738, 2001.

[36] J. Hirschberg. Pitch accent in context: predicting intonational prominence from text. *Artificial Intelligence*, 63:305–340, 1993.

[37] J. Hitzeman, A. W. Black, P. Taylor, C. Mellish, and J. Oberlander. On the use of automatically generated discourse-level information in a concept-to-speech synthesis system. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 6, pages 2763–2766, 1998.

[38] H. Hon, A. Acero, X. Huang, J. Liu, and M. Plumpe. Automatic generation of synthesis units for trainable text-to-speech systems. In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, volume 1, pages 293–296, 1998.

[39] X. Huang, A. Acero, J. Adcock, H. Hon, J. Goldsmith, J. Liu, and M. Plumpe. Whistler: a trainable text-to-speech system. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 4, pages 169–172, 1996.

[40] X. Huang, A. Acero, H. Hon, Y. Ju, J. Liu, S. Meredith, and M. Plumpe. Recent improvements on Microsoft's trainable text-to-speech system: Whistler.

In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, pages 959–962, 1997.

[41] X. D. Huang, A. Acero, and H. Hon. *Spoken Language Processing*. Prentice Hall, 2001.

[42] A. Hunt and A. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proceedings of the Intl. Conf. on Acoustic, Speech, and Signal Processing*, volume 1, pages 373–376, 1996.

[43] F. Itakura. Line spectrum representation of linear predictive coefficients of speech signals. *Journal of the Acoustical Society of America*, 57(4):535, 1975.

[44] R. M. Kaplan and M. Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, 1994.

[45] L. Karttunen. Constructing lexical transducers. In *The Proceedings of the Fifteenth International Conference on Computational Linguistics*, volume 1, pages 406–411, 1994.

[46] L. Karttunen et al. Regular expressions for language engineering. *CUP Journals: Natural Language Engineering*, 4:305–328, 1996.

[47] L. Karttunen, R. M. Kaplan, and A. Zaenen. Two-level morphology with composition. In *The Proceedings of the International Conference on Computational Linguistics*, volume 1, pages 141–148, 1992.

[48] K. Kirchhoff. A comparison of classification techniques for the automatic detection of error corrections in human-computer dialogues. In *Proceedings of NAACL Workshop on Adaptation in Dialogue Systems*, pages 33–40, 2001.

[49] E. Klabbers and R. Veldhuis. On the reduction of concatenation artifacts in diphone synthesis. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 6, pages 2759–2762, 1998.

[50] E. Klabbers and R. Veldhuis. Reducing audible spectral discontinuities. *IEEE Transactions on Speech and Audio Processing*, 9(1):39–51, 2001.

[51] D. Klatt. Review of text-to-speech conversion for english. *Journal of the Acoustical Society of America*, 82(3):737–793, 1987.

[52] W. B. Kleijn and J. Haagen. Transformation and decomposition of the speech signal for coding. *IEE Signal Processing Letters*, 1(9):136–138, 1994.

[53] I. Langekilde and K. Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of COLING-ACL*, pages 704–710, 1998.

[54] S. L. Lauritzen. *Graphical Models*. Clarendon Press, 1996.

[55] A. Ljolje, J. Hirschberg, and J. P.H. van Santen. Automatic speech segmentation for concatenative inventory selection. In J. P.H. van Santen, editor, *Progress In Speech Synthesis*, chapter 24, pages 304–311. Springer-Verlag New York, 1997.

[56] M. W. Macon. *Speech synthesis based on sinusoidal modeling*. PhD thesis, Georgia Institute of Technology, Oct. 1996.

[57] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:2:269–312, 1997.

[58] M. Mohri. Minimization algorithms for sequential transducers. *Theoretical Computer Science*, 234:177–201, March 2000.

[59] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16:69–88, 2002.

[60] M. Mohri, F. C. N. Pereira, and M. Riley. Weighted automata in text and speech processing. In *ECAI-96 Workshop, Budapest, Hungary*, pages 46–50, 1996.

[61] A. Monaghan. Rhythm and stress-shift in speech synthesis. *Computer Speech and Language*, 4:71–78, 1990.

[62] E. Moulines and F. Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication*, 9(5):453–467, 1990.

[63] A. H. Oh and A. Rudnicky. Stochastic language generation for spoken dialogue systems. In *Proceedings of ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 27–32, 2000.

[64] M. Ostendorf, P. Price, and S. Shuttuck-Hufnagel. The Boston University Radio News Corpus. Technical Report ECS-95-001, Boston University, 1995.

[65] M. Ostendorf and N. Veilleux. A hierarchical stochastic model for automatic prediction of prosodic boundary location. *Computational Linguistics*, 201:27–54, 1994.

[66] S. Pan and K. McKeown. Integrating language generation with speech synthesis in a concept to speech system. In *Proceedings of ACL/EACL Concept to Speech workshop*, pages 23–28, 1997.

[67] S. Pan and K. McKeown. Prosody modeling in concept-to-speech generation: methodological issues. *Philosophical Transactions of the Royal Society*, 358(1769):1419–1431, 2000.

[68] B. Pellom, W. Ward, and S. Pradhan. The CU Communicator: an architecture for dialogue systems. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 2, pages 723–726, 2000.

[69] J. Pitrelli, M. Beckman, and J. Hirschberg. Evaluation of prosodic transcription labeling reliability in the ToBI framework. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 1, pages 123–126, 1994.

[70] M. Plumpe, A. Acero, H. Hon, and X. Huang. HMM-based smoothing for concatenative speech synthesis. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, pages 2751–2754, 1998.

[71] M. Plumpe and S. Meredith. Which is more important in a concatenative text-to-speech system - pitch, duration, or spectral discontinuity? In *the* 3^{rd} *ESCA/COCOSDA International Workshop on Speech Synthesis*, pages 231–236, 1998.

[72] A. Potamianos, E. Ammicht, and H-K. J. Kuo. Dialogue management in the Bell Labs Communicator system. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 2, pages 130–133, 2000.

[73] S. Prevost and M. Steedman. Specifying intonation from context for speech synthesis. *Speech Communication*, 15:139–153, 1994.

[74] A. Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, pages 133–141, 1996.

[75] A. Ratnaparkhi. Trainable methods for surface natural language generation. In *Proceedings of the 1st Meeting of the NAACL*, pages 194–201, 2000.

[76] E. Roche and Y. Schabes. Deterministic part-of-speech tagging. *Computational Linguistics*, 21:2:227–53, 1995.

[77] E. Roche and Y. Shabes, editors. *Finite-state language processing.* MIT Press, 1997.

[78] K. Ross. *Modeling of intonation for speech synthesis.* PhD thesis, Boston University, 1995.

[79] K. Ross and M. Ostendorf. Predicting abstract prosodic labels for speech synthesis. *Computer Speech and Language*, 10:155–185, 1996.

[80] A. Rudnicky, C. Bennett, A. Black, A. Chotimongkol, K. Lenzo, A. Oh, and R. Singh. Task and domain specific modelling in the Carnegie Mellon Communicator system. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 2, pages 603–606, 2000.

[81] A. Rudnicky, E. Thayer, P. Constantinides, C. Tchou, R. Shern, K. Lenzo, W. Xu, and A. Oh. Creating natural dialogs in the Carnegie Mellon Communicator system. In *Proceedings of Eurospeech*, volume 4, pages 1531–1534, 1999.

[82] Y. Sagisaka, N. Kaiki, et al. ATR ν-talk speech synthesis system. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 1, pages 483–486, 1992.

[83] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmidt, and V Zue. Galaxy-II: A

reference architecture for conversational system development. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 3, pages 931–934, 1998.

[84] S. Seneff and J. Polifroni. Dialogue management in the mercury flight reservation system. In *Proceedings of the Satellite Dialogue Workshop, ANLP-NAACL*, 2000.

[85] S. Seneff and J. Polifroni. Formal and natural language generation in the mercury conversational system. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 2, pages 767–770, 2000.

[86] K. Silverman. *The Structure and Processing of Fundamental Frequency Contours*. PhD thesis, Cambridge University, 1987.

[87] K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, and others. ToBI: A standard for labeling English prosody. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 2, pages 867–870, 1992.

[88] R. Sproat. English noun-phrase accent prediction for text-to-speech. *Computer Speech and Language*, 8:79–94, 1994.

[89] R. Sproat, editor. *Multilingual text-to-speech synthesis*. Kluwer, 1998.

[90] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. Normalization of non-standard words. *Computer Speech and Language*, 15(3):287–333, July 2001.

[91] R. Sproat and J. Olive. An approach to text-to-speech synthesis. In W. B. Kleijn and K. K. Paliwal, editors, *Speech Coding and Synthesis*, pages 611–633. Elsevier, 1995.

[92] R. Sproat and M. Riley. Compilation of weighted finite-state transducers from decision trees. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 215–222, 1996.

[93] D. Stallard. Talk'n'Travel: A conversational system for air travel planning. In *Proceedings of the Association for Computational Linguistics 6th Applied Natural Language Processing Conference*, pages 68–75, 2000.

[94] M. Steedman. Representing discourse information for spoken dialog generation. In *The Proceedings of the International Symposium on Spoken Dialog*, pages 89–92, 1996.

[95] Y. Stylianou. Applying the harmonic plus noise model in concatenative speech synthesis. *IEEE Transactions on Speech and Audio Processing*, 9(1):21–29, Jan. 2001.

[96] D. Talkin. Voicing epoch determination with dynamic programming. *Journal of the Acoustical Society of America*, 85(Supplement 1), 1989.

[97] D. Talkin. A robust algorithm for pitch tracking. In *Speech Coding and Synthesis*, pages 495–518. Elsevier, 1995.

[98] P. Taylor. Concept-to-speech synthesis by phonological structure matching. *Philosophical Transactions of the Royal Society, Series A*, 358(1769):1403–1416, 2000.

[99] P. Taylor and A. Black. Festival speech synthesis system. Technical report, Human Communication Research Center, University of Edinburgh, 1997.

[100] P. Taylor and S. Isard. Automatic diphone segmentation. In *Proceedings of Eurospeech*, pages 709–711, 1991.

[101] K. Tokuda et al. An algorithm for speech parameter generation from continuous mixture HMMs with dynamic features. In *Proceedings of Eurospeech*, pages 757–760, 1995.

[102] J. P. H. van Santen and B. Möbius. Modeling pitch accent curves. In *Proceedings of ESCA Tutorial and Research Workshop on Intonation: Theory, Models and Applications*, pages 321–324, 1997.

[103] M. Walker, J. Aberdeen, J. Boland, E. Bratt, J. Garofolo, L. Hirschman, S. Lee A. Le, S. Narayanan, B. Pellom K. Papineni, J. Polifroni, A. Potamianos, P. Prabhu, A. Rudnicky, G. Sanders, S. Seneff, D. Stallard, and S. Whittaker. Darpa communicator dialog travel planning systems: the June 2000 data collection. In *Proceedings of Eurospeech*, 2001.

[104] M. Wang and J. Hirschberg. Automatic classification of intonational phrase boundaries. *Computer Speech and Language*, 6:175–196, 1992.

[105] C. W. Wightman, A. K. Syrdal, G. Stemmer, A. Conkie, and M. Beutnagel. Perceptually based automatic prosody labeling and prosodically enriched unit selection improve concatenative speech synthesis. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 2, pages 71–74, 2000.

[106] J. Wouters. *Analysis and synthesis of degree of articulation.* PhD thesis, Oregon Graduate Institute, June 2001.

[107] J. Wouters and M. Macon. Control of spectral dynamics in concatenative speech synthesis. *IEEE Transactions on Speech and Audio Processing*, 9(1):30–38, 2001.

[108] J. Wouters and M. W. Macon. A perceptual evaluation of distance measures

for concatenative speech synthesis. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 6, pages 2747–2750, 1998.

[109] J. Yi. Natural-sounding speech synthesis using variable-length units. Master's thesis, MIT, 1998.

[110] J. Yi and J. Glass. Natural-sounding speech synthesis using variable-length units. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, pages 1167–1170, 1998.

[111] J. Yi, J. Glass, and L. Hetherington. A flexible, scalable finite-state transducer architecture for corpus-based concatenative speech synthesis. In *Proceedings of the Intl. Conf. on Spoken Language Processing*, volume 3, pages 322–325, 2000.

[112] S. Young and F. Fallside. Speech synthesis from concept: a method for speech output from information systems. *Journal of the Acoustical Society of America*, 66(3):685–695, 1979.

[113] http://www.research.att.com/∼walker/eval/eval.html.

[114] http://fofoca.mitre.org/.

[115] http://www.ai.sri.com/∼communic/.

[116] http://ssli.ee.washington.edu/projects/synth/icassp02.html.

VITA

Ivan Bulyko grew up in Moscow, Russia. In 1992, he moved to Boston, MA where he received a full tuition scholarship from Suffolk University. He graduated Summa Cum Laude in 1997, earning a B. S. degree in Computer Science and Electrical Engineering. Ivan started his graduate studies at Boston University, where in 1999 he received an M. S. degree in Computer Engineering, before transferring to the University of Washington to pursue his Ph.D. Ivan's research interests include speech synthesis, recognition and natural language processing. He is the first author on one published journal article and one article in preparation, as well as five conference publications.