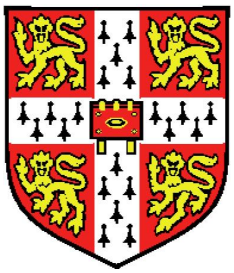
MPhil in Computer Speech, Text and Internet Technology

Adaptation of Prosodic Phrasing Models



Peter Bell

St John's College

University of Cambridge

21st July 2005

Declaration

I, Peter Bell of St John's College, being a candidate for MPhil in Computer Speech, Text and Internet Technology, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Peter Bell

21st July 2005

Word count: 14,870 (excluding code in Appendix A)

Abstract

In this thesis we investigate adaptive techniques to be used with models for predicting prosodic phrase boundaries for speech synthesis. Motivated by adaptation methods used for speech recognition, we design techniques appropriate for use with decision-tree prediction models and Hidden Markov Models. We compare techniques for adaptation between different speakers and types of spoken material. We find the most successful techniques to be a method of transforming decision tree output and a method of altering a statistical model of prosodic phrase length.

Acknowledgements

I would first like to thank my supervisors, Dr Tina Burrows and Dr Paul Taylor, for all their advice, motivation and constructive criticism over the last few months. I would also like to thank Toshiba Research Europe for allowing me to use their corpus for my experiments; and for giving Tina the time to supervise me, and for their excellent coffee. I am grateful to Professor Steve Young, who purchased the Boston University Radio News Corpus for last year's project. Without both corpora, it would have been much more difficult produce useful results.

Contents

Chapter 1

Introduction

1.1 Prosodic Phrasing

The collective term *prosody* is used to refer to aspects of speech pronunciation that cannot be described by a sequence of phones. As such, it can encode additional information about the speech such as emotion, syntax, and dialogue cues (see Fordyce [7] for more detail on this). One element of prosody is the division of an utterance into prosodic phrases, indicated in speech by pauses, changes in amplitude and pitch, and the lengthening of the final syllable within a phrase. This can significantly aid speech understanding by providing syntactic clues, and can even change the meaning of a statement completely. For example, the location of phrase boundaries, indicated by changes in pitch, would disambiguate between the following (from [22]):

```
(Police help dog)(bite victim.)
(Police help)(dog bite victim.)
```

And as an alternative (new) example:

```
(His political career wasn't ruined)(due to the actions of his mistress.)
(His political career wasn't ruined due to the actions of his mistress.)
```

(the former implies that his political career was saved by his mistress, the latter that it was ruined for some other reason).

The prediction of phrase boundary locations is an important task for speech synthesis applications, not only to ensure that the syntax of a sentence is correctly expressed, but also to increase the naturalness of the synthesised speech: as mentioned above, the amplitude, intonation and syllable duration are all dependent on the phrasing. It is clear that there is a natural correspondence between syntactic phrasing and prosodic phrasing and syntactic phrasing. For example, the phrasing of the recorded utterance

```
(Her dress was worn)(on prom night.)
```

corresponds exactly to the division of the sentence into a noun phrase and prepositional phrase. However, it is accepted that there is no exact relationship between the two (see, for example, Fach [5]), and that many other factors influence the location of prosodic phrase breaks. Nevertheless, many approaches to the prediction problem are to some extent based on syntactic information, such as parts-of-speech. For example, a rule-based approach, making predictions entirely on the basis of the sequence of function words, content words and punctuation, compares reasonably well to more sophisticated approaches. Predicting breaks coincident with punctuation marks, if these are present in text to be synthesised, is another simple but effective approach.

1.2 Variation

Despite the links with syntactic structure, there has been found to be considerable variation in prosodic phrasing between speakers using different types of speech. For example, in fast speech there will be fewer pauses: speakers tend to concatenate the phrases that are least significant for conveying meaning. Conversely, in very slow speech, phrasing may be shorter: there may be pauses between every significant word, with the corresponding changes in intonation at phrase boundaries.

In addition, phrasing is likely to vary with the domain of the subject material. For example, in the reading of more text with very long sentences, the reader could make greater use of phrasing to enable the text to be more readily understood. For non-rehearsed news-reading, the presenter may opt for a style of regular

phrase lengths with a lower dependence on the text itself, allowing the speech to sound interesting without the challenge of deep-parsing the text on the fly.

This variability presents challenges for the design of phrase break prediction algorithms. In order for the synthesised speech to sound appropriate for its intended domain, it must be possible to adapt the methods to alternate domains from the one on which they were first conceived. This reduces the usefulness of rule-based approaches, which can be adapted only by altering fixed constraints. Data-driven methods, however, in which prediction models are derived from a representative sample of training data, may readily be re-trained, provided there is sufficient data from the new domain available to do this. However, it is desirable, given the difficulty and cost of obtaining large quantities of training data for every desired speaking style and text type, to develop phrasing models that can be easily adapted to new conditions with a reduced amount of training data, but minimal reduction in predictive power: the principal goal of this project.

1.3 Adaptation of Prosodic Phrasing Models

This project aims to investigate data-driven methods of adapting well-trained phrasing models to alternate domains, using a limited quantity of training data. In the process, we implement varying methods of prediction, starting with the the "chink-chunk" rule-based method [11], and then using a decision-tree classifier [15], determining appropriate syntactic features as inputs. We then design an HMM-like method with a language model based on word syllable length, similar to that implemented by Schmidt and Atterer [19].

Various adaption methods are investigated: a change in fixed constraints for the rule-based approach; adaptation of the decision tree using by learning appropriate transformations; and parametric mapping of the HMM language model, together with a mapping of output probabilities based on perplexity and confidence.

Chapter 2

Prosodic Break Prediction

2.1 Theory of Prosodic Phrasing

In this work, we adopt the ToBI ("**To**nes and **B**reak **I**ndicies") labelling system to specify prosodic phrase boundaries. ToBI [20] is a widely-used scheme for prosodic transcription which can be used to annotate natural utterances, or to specify the prosodic features to be generated for synthesised speech. The scheme has its basis in Pierrehumbert's theory of intonation [14], and consists of three tiers of features: the *tone* tier, marking intonation; the *break index* tier, marking prosodic breaks; and a miscellaneous tier, marking other features of recorded speech such as background noise. The features in each tier are specified by a finite set of possible markers.

In the ToBI break index tier, word junctures are labelled with a number from 0 to 4 according to the strength of association between the two, with 0 being the most strongly associated and 4 being the weakest. The highest two indices correspond with the two types of intonational phrases under Pierrehumbert's theory, index 3 indicating an *intermediate* phrase boundary and index 4 indicating a full intonational phrase boundary. Of the three types of "intonation event" specified by the theory, *phrase accents* are associated with intermediate phrases and *boundary tones* with intonational phrase boundaries. A consequence of this is that level 3 and 4 breaks can be closely related to the intonation contour of an utterance.

A major advantage of using the ToBI scheme as a basis for these experiments is that the theory has a large following, and there exists a sufficient quantity of ToBI-labelled data suitable for machine-learning and evaluation. This avoids the need for recourse to signal-processing methods to extract prosodic features automatically from speech, allowing more time to be focussed on prediction and adaptation methods.

For the purposes of break prediction for TTS, it is often considered sufficient to reduce the break indices to two groups, levels 0-2 being classified as non-breaks, and levels 3 and 4 being classified as breaks. However, some works use three categories, being non-break (0-2), level 3, and level 4. We tend to use the former here: this enables results to be presented more simply, and allows easier comparisons to be made with the majority of past experiments. In addition, most level 4 breaks have been found to occur at sentence boundaries or other punctuation marks: predicting the location of level 3 breaks is where most difficulty lies. Therefore carrying out break/non-break (B/N) prediction is not a means of artificially simplifying the task.

Given the decision to treat junctures as N and B, I will tend to use 'phrase break' synonymously with 'phrase boundary', even though the former strictly refers to all possible break types, whether indicating a boundary or not.

2.2 Overview of Prediction Algorithms

2.2.1 Review

This research is concerned with the development of phrasing models that are readily adaptable to different domains and speaking styles, and the methods of performing such adaptation. As there exist many good reviews of prior work on model development (for example, [22], [13]), I will not go into detail here, but will explain the wider context of our model choice.

Early approaches tended to be rule-driven. For example, Bachenko and Fitzpatrick [1], based their system on deep syntactic analysis of the text (this built on earlier studies by Gee and Grosjean [9]). In contrast, Liberman and Church [11] made predictions based on a simple function word/content word distinction.

More recent works have used machine-learning techniques: Wang and Hirshberg [22] used Classification and Regression Trees to make predictions based on a range of textual features, including sentence type and part-of-speech (POS) information. In common with Bachenko and Fitzpatrick, they consider that the regularity of phrase lengths provides some constraint on boundary location, noting, however, that their method does not allow for the elapsed distance from the previous boundary to be used as a feature, since previous breaks are unknown.

Ostendorf and Veilleux [13] adopt a hierarchical stochastic model in which sentences are composed of major phrases, in turn composed on minor phrases, themselves composed of a series of minor breaks; with the probability of a sequence at any level in the hierarchy dependent on the sentence. Rather than being used to directly classify the breaks, decision trees are used as a conditioning on which to obtain the probability of a minor break at the lowest level in the hierarchy. Features used to train the models include word position in sentence, punctuation, word class (whether a function or content word), and information about the larger syntactic chunk containing the word.

Taylor and Black [2] take a much simpler stochastic approach, training HMMs for a two word vocabulary consisting of the juncture types N and B, with an output distribution given by the probability of observing a POS sequence, conditional upon the juncture type. An N-gram language model assigns probability to a sequence of junctures, independent of the 'observed' sentence orthography. The use of a Viterbi decoder with this language model thus allows phrase length to be taken into account. Schmid and Atterer [19] adapt this method. They note that the high-order language model required to model long intonational phrases leads to data sparsity problems in training the models, whilst the models themselves contain a high degree of redundancy. They replace the N-gram LM with a juncture probability conditioned on the distance (in syllables) from the previous phrase boundary.

Busser et al [3] used TiMBL, a memory-based learner developed for natural language processing tasks. They classify junctures with this program based on a range of features of the surrounding words: word type, POS tag, and an "expanded tag", being the word itself if a function word, and the POS tag otherwise. Memory-based learning classifies by extrapolating from the most similar items

of training data in memory, and thus has been shown to work much better than decision-trees in the classification of rarely-seen cases. Ingulfsen [10] used TiMBL for classification based on a range of deeper syntactic features (for example, from parsing using a dependency grammar), concluding that despite the additional complexity involved, classification performance was not significantly improved over that using the shallow representations of Busser et al.

2.2.2 Model Choice

Bearing in mind the previous work described, we choose models to investigate based on the following considerations:

1. The model must be easily adaptable to alternative domains and speakers. For this to be possible, the model must be data-driven. This rules out some of the earlier rule-based models, except for use with manual adaptation as a baseline system.

2. It has been shown that models based on shallow features such as POS and word type (ie. content or function) have no worse performance than those based on deeper syntactic analyses, with much lower complexity. Therefore we use models based only on these features.

3. The model should contain as many adaptable features as possible. Therefore we prefer models which consider the notion of phrase length distribution, such as those developed by Taylor and Black, and by Schmid and Atterer.

4. The model should not be unnecessarily complicated to implement. We therefore adopt a decision-tree based approach to calculation of observation probabilities, such as that used by Ostendorf and Veillieux, for which software is readily available. For simplicity, we use the same software for binary classification tasks.

For a baseline system, we use Liberman and Church's "chink-chunk" algorithm, which is extremely simple to implement given parts-of-speech information. We use HMMs with two different phrase-length based language models, similar

to those used by Schmidt and Atterer: these are preferred over the high-order N-gram models used by Taylor and Black as they make the implementation of a decoder simpler, avoid the need to deal with data sparsity issues, can be more easily parametrised for adaptation to new domains, and have been shown to achieve higher classification performance.

For decision-tree construction, we use Quinlan's C4.5 program [15]. This is open-source, so has the advantage that code for a Viterbi decoder and any other embellishments can be built on top of the existing program.

2.3 Details of Models

2.3.1 The Chink-Chunk algorithm

Tag	Function
CC	Conjunction
DT	Determiner
EX	Existential "there"
IN	Preposition
MD	Modal
POS	Possessive
RP	Particle
TO	"to"
WDT	Wh-determiner
WP	Wh-pronoun
WRB	Wh-adverb
CD	Cardinal number
PP	Personal pronoun
PP$	Possessive pronoun
UH	Interjection
WP$	Possessive wh-pronoun

Table 2.1: List of POS tags in the Penn Treebank tagset corresponding to closed-class words that are classified as functions words (F) in the CFP tagset.

The "chink-chunk" algorithm is based on the distinction between open-class "content words" (C), closed-class "function words" (F), and punctuation marks (P). Function words are shown in table 2.1 (from Busser et al). It predicts a phrase break after every punctuation mark, and at the end of a series of content words (ie. when the next word is a function word). This can be written in pseudo-code as:

```
if POS_previous = P:
    juncture <- break
else if POS_previous = C and POS_next = F:
    juncture <- break
else:
    juncture <- non-break
```

We use this deterministic algorithm to provide a baseline to which to compare our initial data-driven models.

2.3.2 C4.5 Decision Trees

The C4.5 program generates a decision tree from training data using an information theoretic approach. Given a set S of cases, in which a proportion p_i belong to class i, the entropy $H(S)$ is a measure of the information contained in the set, calculated as

$$H(S) = -\sum_i p_i log_2 p_i$$

A tree containing maximal information about the classes of training cases will minimise this figure, calculated over all branches. In our experiments, cases might typically be a trigram consisting of the parts-of-speech of two words prior to a juncture j_n and one following (which we label POS_{n-2}, POS_{n-1} and POS_{n+1}), with two classes, B and N.

When growing the decision tree, we aim to partition the set of available training cases at each branch so as to maximise the reduction in entropy. When partitioning a training set T into subsets T_k according to a question Q (for example "What tag is POS_{n-1}?"), the new entropy is given by the weighted sum

of the subset entropy:

$$H_Q(T) = \sum_k \frac{|T_k|}{|T|} H(T_k)$$

thus the reduction in entropy from partitioning with question Q is given by the information gain function

$$gain(Q) = H(T) - H_Q(T)$$

In practice, partitioning cases in order to maximise $gain(Q)$ results in bias towards questions that divide the cases into a greater number of outcomes. To overcome this, we instead choose Q so as to maximise the *gain ratio*, which expresses the information gain relative to the entropy gain of Q independent of class, given by

$$gr(Q) = \sum_k \frac{|T_k|}{|T|} log_2 \frac{|T_k|}{|T|}$$

The C4.5 tree-growing algorithm continues to expand each branch until each terminal branch ("leaf") contains cases of just one class, or until no further questions can be asked - no particular stopping criteria are used. However, this results in a tree which overfits the training data, with a consequent lower performance on unseen data. Therefore the trees are pruned. C4.5 uses an error-based pruning technique: at each node in the tree, an error rate is calculated on the assumption that all cases at the node are classified as the most common class amongst them. A pessimistic measure of error is used, being an upper binomial confidence limit for the real error based on the number of training cases classified and proportion classified incorrectly. Each subtree is replaced by a leaf (or by its most frequently-used branch) if doing so is calculated to reduce the expected error rate for that subtree.

After a tree has been grown and pruned by C4.5, it can be used to classify unseen data in more than one way. Given a set of input features, the tree can return the number of items of training data from each class at the leaf node corresponding to those features. The basic C4.5 classifier will hard-assign a class to the unseen case on the basis of the most common class at that leaf node. However, the figures can also be used to estimate the probability of the unseen case being a particular class, given its input features. The advantage of using the

decision tree for this purpose is that it conditions upon only the most relevant features (according to the minimum-entropy criterion) in order to calculate the probability, thus reducing data sparsity problems; whilst the pruning algorithm avoids probabilities being too specific to the training data. In the event that there are still too few cases at the leaf node to obtain a reliable probability estimate, it is straightforward to back-off to the immediate parent node.

It should be noted that when making decisions on the basis of POS tags, we make no attempt to automatically group tags leading to similar decisions, in order to further improve estimation accuracy. The optimal clustering of tags for this purpose has been the subject of recent research by Read and Cox [17], but we found that results no worse could be obtained using C4.5 without this refinement.

2.3.3 HMM Decoder

In developing an HMM-based break-prediction algorithm, we adapt the work of Taylor and Black, and Schmid and Atterer, combining elements from each. Presuming that a sequence of words, W, has been observed, we attempt to find the sequence of breaks and non-breaks, J, over all the word junctures, giving the highest likelihood of these words. $J = (j_1, \ldots, j_n)$ with each juncture taking the value B (break) or N (non-break). Firstly we identify $C = (C_1, \ldots, C_n)$, the entire set of features, extracted from the word sequence, which are considered relevant for break prediction (except anything connected with phrase length). In particular, we set C_i to be the "context" of juncture i, usually the values of POS_{i-2}, POS_{i-1} and POS_{i+1} from the words on either side of the juncture, as might be used as inputs to the C4.5 classifier for prediction of that break.

We therefore seek to maximise

$$\hat{J} = \arg\max_J p(J|C) \tag{2.1}$$

$$= \arg\max_J \frac{p(C|J)P(J)}{p(C)} \tag{2.2}$$

We ignore the denominator $p(C)$ since this is invariant to change in J, so isn't of relevance to the maximisation.

We d_i to be some measure of the "time" elapsed since the most recent break prior to the i^{th} juncture, j_i – ie. the length of the phrase up to that point. . We then make the following simplifying assumptions:

1. For the *POS-sequence model*, the Conditional Independence assumption,

$$p(C_i|J) = p(C_i|j_i)$$

   in other words, the features surrounding a juncture depend only on whether that juncture is a break or non-break. As with similar assumptions in HMMs for speech recognition, this is somewhat implausible due to the overlap between the context of successive junctures. In practice, however, it has been found to work well.

2. For the *phrase-length* model

$$p(j_i|J) = p(j_i|d_i)$$

   the probability of a juncture, ignoring context, depends only on the time since the last break, rather than on the exact sequence of preceding junctures.

We develop one model where d_i is measured simply in terms of the number of intervening words and one where it is measured as the number of intervening syllables, as a proxy for word duration.

Under these assumptions, (2.2) becomes

$$\hat{J} \quad = \arg\max_J \prod_i p(C_i|j_i)p(j_i|d_i) \tag{2.3}$$

This equation is not simple to solve because, since the best sequence of junctures in unknown, the values of d_i are unknown too – to determine d_i it is necessary to know the the last juncture before j_i that was a break. However, it can be solved by making the Viterbi appoximation, and using an adapted version of the Viterbi algorithm. This is explained in detail below. Henceforth we refer to the $p(C_i|j_i)$ as the *POS-sequence model*, or, following standard HMM terminology, as "output" or "observation" probabilities. We refer to $p(j_i|d_i)$ as the *phrase-length* (or simply *phrase*) model.

Viterbi formulae

The complication in solving the equation above lies in taking account of the possible sequences of junctures. Under the Viterbi approximation, we consider that only the most likely sequence of junctures contributes to the likelihood of the observed features. We define the path probability $\phi_d(i)$ to be the probability of the *most likely* partial juncture sequence ending with j_i, with time d since the previous break in this sequence. In this section we always take d to be calculated based on the most likely juncture sequence. To aid the construction of equations, we take the j_0 to be the imaginary 'juncture' at the beginning of the sentence, always a break, because the beginning of a sentence always coincides with the beginning of a prosodic phrase.

Under our first model, d takes values in $\{0, 1, 2, \ldots, D\}$ with $d = 0$ signifying that the juncture j_i itself is a break,and $d = 1$ signifying the the previous juncture is a break. D is the maximal phrase length, beyond which we treat probability $\phi_d(i)$ as being constant. The Viterbi equations are:

$$\begin{aligned}
\phi_d(0) &= \begin{cases} 1, & d = 0 \\ 0, & d > 0 \end{cases} \\
\phi_d(i) &= \begin{cases} \max\limits_{1 \leqslant k \leqslant D} \phi_{k-1}(i-1) p(B|k) p(C_i|B), & d = 0 \\ \phi_{d-1}(i-1) p(N|d) p(C_i|N), & d > 0 \end{cases}
\end{aligned}$$

for each $i = 1, \ldots, n$. (The equations are slightly different for $d = D$). In practice, we work using log probabilities throughout to avoid arithmetic underflow.

The first equation follows from the fact that the beginning of the sentence is certain to be a phrase boundary, ie. j_0 is must have $d = 0$ – and there is no observed C_i for the beginning of the sentence. For junctures after the beginning of the sentence, with $i > 0$ there are two possible cases:

$d > 0$ In other words, the juncture is not a break, and there are d words between the juncture and the previous break. This means that the previous juncture must be $d - 1$ words away from the previous break (if $d - 1 = 0$ this means that the previous juncture is a break).

Therefore the probability of juncture j_i being a d words from a break is given by the probability that the previous juncture, j_{i-1} is $d-1$ away from a break($\phi_{d-1}(i-1)$), multiplied by the probability of j_i itself not being a break given d ($p(N|d)$).

$d = 0$ This means that the juncture is itself a break. It could have ended a phrase of a range of possible lengths. Imagine that it ended a phrase that ended up being of length k. This would mean that the previous juncture, j_{i-1}, would have had to have been $k-1$ away from the previous break, and so the probability of j_i being a break would be $\phi_{k-1}(i-1).p(B|k)$, similar to the case above.

Since k could be any value, we would normally have to take account of all possible values of k when calculating the overall probability of j_i being a break. However, we make the Viterbi approximation: we only have to consider the most *likely* path – so we choose k so as to maximise $\phi_{k-1}(i-1).p(B|k)$.

To incorporate the probability of observing context C_i, we multiply the expressions derived above by one of $p(C_i|B)$ or $p(C_i|N)$ as appropriate. At each stage, we store the value of k chosen to maximise $\phi_0(i)$, which can be labelled as k_i. Traceback to find the most likely sequence of junctures is simple: if the path has a break at juncture j, the previous break will be at $j - k_j - 1$.

The case where d is measured by the number of syllables occurring between breaks is a little more complex. This time $d \in \{1, 2, \ldots, D\}$, with $\phi_d(i)$ being the probability of the best sequence $(j_1, \ldots, j_i)$ ending with a phrase at least d syllables long. The phrase is length is measured up to the end of the word following the juncture j_i, since it is fully determined by the sequence of junctures up to that point. $|w_i|$ is taken to be the length, in syllables, of the word immediately preceding the i^{th} juncture. The equations (due to Schmid and Atterer) are:

$$\phi_d(0) = \begin{cases} 1, & d = |w_1| \\ 0, & d \neq |w_1| \end{cases}$$

$$\phi_d(i) = \begin{cases} 0 & d < |w_{i+1}| \\ \max\limits_{1 \leqslant k \leqslant D} \phi_d(i-1)p(B|k)p(C_i|B) & d = |w_{i+1}| \\ \phi_{d-|w_{i+1}|}(i-1)p(N|d-|w_{i+1}|)p(C_i|N) & d > |w_{i+1}| \end{cases}$$

These equations can be explained similarly to before, this time by considering three possibilities for d:

$d < |w_{i+1}|$ This is the situation that, at the end of the following word (of length $|w_{i+1}|$), the phrase is less than $|w_{i+1}|$ long. Since junctures can't occur in the middle of words, this is impossible, so a probability of zero is attached to this case.

$d = |w_{i+1}|$ In this case, at the end of the following word, the phrase is $|w_{i+1}|$ long up to that point. This means that the juncture j_i itself must be a break. The equation is the same as in the previous set.

$d > |w_{i+1}|$ Since the phrase is greater than $|w_{i+1}|$ long, j_i must be a non-break, and the phrase length up to j_i is given by $d - |w_{i+1}|$. Again, the equation follows similarly to before.

In this case traceback is more complicated, as it is necessary to use a token-passing approach (rather than simply storing the optimal values of k) . Tokens store the position of the most recent break for the path considered, and are propagated forwards for $d > |w_{i+1}|$. For $d = |w_{i+1}|$ (signifying juncture i is a break) the token corresponding to the best value of k is recorded. The best sequence of breaks can then be found similarly to the model above, using these recorded values.

Parameter estimation

To estimate the probabilities $p(C_i|j_i)$ we use the C4.5 classifier; this is partly for ease of implementation – since the original program can easily be modified to obtain the relevant statistics – and partly to avoid data sparseness issues. Using a C4.5 tree, grown from training data according to the entropy gain criterion and

then pruned, we identify the leaf L corresponding to features C_i. The tree stores the counts of training cases of B and N at L, allowing us to estimate

$$p(B|C_i) = \frac{\#B \text{ at } L}{\text{total cases at } L} \tag{2.4}$$

and the context-independent probability $P(B)$ is estimated from total counts over the whole tree. We then use Bayes rule to obtain

$$p(C_i|B) = \frac{p(B|C_i)p(C_i)}{p(B)} \tag{2.5}$$

Note that $p(C_i)$ can be ignored since it does not vary with the sequence of junctures. $p(C_i|N)$ can be estimated similarly.

In the event that there are insufficient training cases at L to obtain a reliable estimate (three or fewer was found to work best), it is possible to back off to L's parent node in the tree, obtaining juncture counts from this node instead. The rationale for this is that the feature discriminating L from its parent is the least relevant for classification. In most cases, however, back-off is not required, as the tree-pruning process carried out by C4.5 tends to remove leaf nodes with a very small number of cases. A second, cruder back-off procedure is used in the case that there are sufficient training cases at L, but all are of one juncture type. In this case the unseen juncture type is given a count of one.

To estimate $p(j_i|d_i)$ we obtain counts of breaks, $C_B(d)$, and non-breaks, $C_N(d)$, occurring at a distance d from the previous break, by processing the training data sequentially. Then, simply,

$$p(B|d) = \frac{C_B(d)}{C_B(d) + C_N(d)} \tag{2.6}$$

In the case of the latter model, to obtain syllable counts for each word, we used an algorithm by Greg Fast [6]. This counts one syllable for each sequence of vowels, then makes adjustments for certain groups of letters in the word, for example, adding a syllable for "ia" and "io", but then subtracting one for "tia" and "ion". The algorithm is also used to construct syllable counts for test data for input to the decoder.

Data sparsity problems are very small compared with an N-gram model (such as that proposed by Taylor and Black) where the number of parameters to estimate increases exponentially with D, rather than linearly as in this case. However, problems do occur at high d as the pool of training cases is much smaller: only phrases that are at least as long as d contribute a case to the calculation of $p(B|d)$.

Chapter 3

Adaptation

3.1 General Theory

We aim to adapt well-trained phrasing models to new domains, using a relatively small, fixed amount of labelled adaptation data. (This is known as *static supervised adaptation*). To date, little work has been carried out on this task. However, much research has focused on similar adaptation to different speakers of models developed for speech recognition (ASR), and it is to this that we turn to find techniques that may be used for phrasing models.
Adaptive techniques for ASR can generally be categorised as one of the following:

1. **Classification:** Well-trained models are constructed, corresponding to a range of speaking styles. The adaptation data for a particular speaker is used to select the most appropriate existing model set.

2. **Normalisation:** Elements of the feature vector space are mapped to make speakers appear similar, so that one model set can be used for all.

3. **Model adaptation:** Model parameters are re-estimated, using adaptation data.

The use of these techniques is described in more detail in [8]. Briefly: for speaker classification, models are trained on clusters of speakers, grouped according to the projection of individual model parameters onto a low-dimensional eigenspace giving the highest inter-speaker variability; speaker normalisation can be carried

out using a range of front-end signal processing techniques; and model adaptation methods include Maximum A-Posteriori estimation and linear mapping of parameters, fitted using least-squares regression.

Adaptive methods for the prediction of phrase breaks can, broadly speaking, be similarly classified. However, there are several differences between the two tasks which necessitate considerable alteration to the methods as applied to ASR. They can be summarised as follows:

- In ASR models, the feature vector space is high-dimensional and continuous. In this case, it is low dimensional, with elements taking a small set of discrete, unordered values.
- In HMMs used for ASR, the language model takes a large number of discrete, unordered values. In our phrase break prediction HMM, the "phrase model" takes a small number of ordered values.
- Although we have data recorded from a range of speakers, there is little evidence that phrasing varies significantly from speaker to speaker if the overall speaking style is the same (whereas for ASR, differences in physiology, prosody and phonetic realisation are all important). It is therefore harder to create a range of model sets as described in (1) above.
- In ASR, there is generally a high degree of consistency between differently lexically-tagged data sets (and even phonemically-tagged ones). Prosodic tagging, especially tagging of phrase boundaries, is much more subjective. There may be systematic inter-annotator differences between data sets, in addition to any natural phrasing variation. Theories of prosodic phrasing are not yet, unlike phonemic theory, fully agreed upon.
- The C4.5 prediction method we use is not a statistically-based model, and thus it is not possible to use techniques, in particular those classified as (3) above, which, in ASR, are based on altering statistical parameters.

In describing our methods of adaptation of the phrase-length models described in the previous chapter, we will discuss how their development from standard ASR techniques is influenced by the above considerations.

3.2 Methods Used

3.2.1 Transforming C4.5 trees

Background

As discussed above, C4.5 is not a statistical classifier. Methods of parameter mapping used for HMM-based ASR are not applicable. If we wish to make some transformation of C4.5 predictions to make them more appropriate to a new domain (without retraining a new tree from scratch), alternative methods are required.

We work from the observation that there is a degree of homogeneity in the syntactically-based determination of break location across all domains. This is illustrated by the fact that the "chink-chunk" algorithm, based on a content-word/function-word/punctuation distinction, achieves similar performance across all domains. Also, we have found that, using well-trained part-of-speech models from one domain, and applying suitable phrase length constraints (see 3.2.2, below), it is possible to achieve a performance on data from an alternative domain not much lower than well-trained models from the same domain. This is despite the fact that no syntactic information from the new domain is used in training.

This suggests that given a limited amount of training data from a new domain, insufficient to create a reliable C4.5 tree based on a large POS-tagset, it should be possible to base the tree on a large training set from an alternative domain. Rather than using the limited data from the new domain to create an entire tree, it can be used to identify systematic differences in the relationships between POS-context and juncture type between the new domain and the alternative training domain.

This can be illustrated by an example using two of our data sets, "F2B" and "X-Z", which correspond to different domains (see chapter 4 for details of the data sets used). A C4.5 tree trained on a large amount of data from the F2B set

will predict the juncture following an adjective as a non-break. Adaptation data from the X-Z set shows no evidence against this prediction, *except* in the where the juncture is also followed by a singular or mass noun, where 81% of such cases are breaks. (There are few other examples of such strong evidence against the predictions of the F2B tree). Therefore it is likely that better predictions on X-Z data could be obtained by modifying the F2B tree to predict a break, rather than a non-break, in this case.

Method

Following the theory above, given a large amount of training data from one domain which we wish to adapt to an alternative domain using a limited amount of adaptation data, we aim to identify contexts where the adaptation data provides significant evidence against the predictions of the foreign training data. To do this we carry out the following procedure:

1. Grow a C4.5 tree from the foreign training data
2. Use this tree to make predictions (B or N) for the adaptation data
3. Label each case in the adaptation data with one of four tags, tB, fB, tN and fN; where tB indicates a "true break", a break correctly predicted, fB incorrects a "false break", a predicted break which is actually a non-break; and similarly for N.
4. Train a classifier for the four cases above, using the same features as before, according to some heuristic.

To use this to predict phrase boundaries for test data from the new domain, we initially use the foreign tree to make a decision, and then use the output of the second classifier to reverse this if necessary. For example, if the tree predicts B and the second classifier predicts fB (ie. there is evidence that the foreign tree tends to be wrong in this case) then we reverse the initial decision, to predict N instead.

The reason for using four classes, rather than minimum of two – "change" and "don't change" – is that it is expected that these will generally cluster in feature

space, making the second classification task easier. It is, of course, possible to constrain the second classifier so that if the original tree predicts a B for a particular juncture context, then the second classifier will predict only tB or fB, as the other classes are meaningless in this case (and similarly for N).

Classification

We experiment with a range of techniques for the adaptation classifier; results are shown in chapter 5. We take into consideration the following:

- **Choice of classifier**: we use the C4.5 decision-tree classifier, as for the main classification task; but with varying amounts of pruning.
- **Constraints**: whether to restrict the classifier to predicting tB or fB, given the original tree predicting a B; or whether to ignore meaningless predictions.
- **Heuristics**: whether to choose the most probable class, according to training on the adaptation data, or to attempt to optimise the value of a more complicated risk function.

In the latter case, we can construct a "reward" function for the classification. Given that the original tree classifies a test case as B, and assuming the second classifier than classifies it as either tB or fB, then if the test case really is N, and the classifier predicts fB, then the prediction will be reversed, and this will lead to a performance improvement. If the classifier predicts tB, there will be no change in performance. Conversely, if the test case really is B and the classifier predicts fB, there will be a reduction in performance. Arbitrarily choosing r to be the relative "negative reward" associated with a bad decision, the reward function, R for a decision can be represented

	Classified As	
Class	**tB**	**fB**
tB	0	$-r$
fB	0	1

and a similar table can be constructed for cases for which the original classifier predicts N.

Given context C_i, we can classify at the adaptation level to maximise the expected reward function, $\mathbb{E}[R]$, calculated here for the case of an initial prediction of B:

$$\begin{aligned} \mathbb{E}[R(\text{tB}|C_i)] &= 0 \\ \mathbb{E}[R(\text{fB}|C_i)] &= p(\text{tB}|C_i) - r.p(\text{fB}|C_i) \end{aligned}$$

The probabilities shown can be estimated from the cases at the decision tree node corresponding to C_i, in the same way as for our HMM method.

3.2.2 HMM Parameter Adaptation: Phrase-Length Model

The phrase length model concerns the estimation of $p(j_i|d_i)$, where j_i represents the type of the i^{th} juncture and d_i represents the distance from the start of the current phrase to the i^{th} juncture (assuming the most likely sequence of phrase breaks up to the current juncture). We concentrate here on the model in which phrase length is measured in syllables: in this case, our early results indicated potential for performance increases by adjusting these probability figures to ones more appropriate to the new domain (see Chapter 5 for more detail on this). This technique can be carried out independently of any change to the POS-sequence models.

Unlike N-gram language models in ASR, here d takes ordered, discrete values, and we can reasonably expect the probability of a break to vary continuously with phrase length, with the probability of a break, given that the current phrase is already of at least length d, to increase steadily with d – in other words, the longer the phrase already has become, the more likely it is to end at that point.

Figure 3.1 shows $p(B|d)$ for varying d, estimated from one of our large data sets, and this appears to confirm our expectation. Given this fairly smooth curve, we can attempt to adapt the model using as little adaptation data as possible by describing the curve using a small number of parameters, and then fitting these parameters to the adaptation data. This is similar to techniques for adapting observation-probability models in ASR, where the state space is continuous.

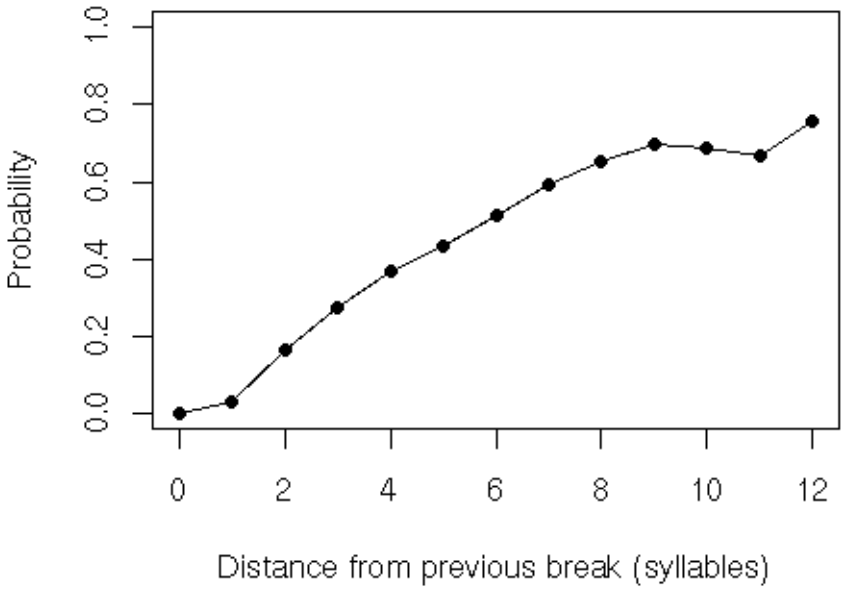


Figure 3.1: Example graph showing the probability of the current phrase ending, given the number of syllables since the start of the phrase

Underpinning our approach is the assumption that each domain has its own intrinsic "average phrase length". This might vary with, for example, speaking rate – Yeon-jun Kim and Yung-hwan Oh [23] have observed that the higher the speaking rate, the more words there are in a prosodic phrase – or with the material spoken – a lengthy radio news story may have long, regular phrases, whilst simple directions from an in-car navigation system may have many more pauses to make the utterance as easy as possible to understand. To relate this notion to the instantaneous break probabilities used by the models, however, we need to consider the underlying phrase length distribution.

We define Y to be the discrete random variable representing the distribution of phrase lengths over a particular domain. The phrase length distribution function is

$$F(y) = p(Y \leqslant y)$$

and we set

$$\lambda_y = p(Y = y | Y \geqslant y)$$

This is the probability of a phrase break occurring at a juncture, given that the phrase is already y units long. Since every phrase must have a positive length, $p(Y \geqslant 1) = 1$ and $\lambda_0 = 0$.

The formula (2.6) given in the previous chapter for estimating $p(B|d)$ actually estimates the values given by

$$\begin{aligned} p(B|d = y) &= p(Y = y | Y \geqslant y, \text{ juncture at } y) \\ &= \frac{\lambda_y}{p(\text{juncture at } y | Y \geqslant y)} \end{aligned}$$

This calculation takes account of the fact that it is not possible for the phrase to end any arbitrary y – only those which coincide with word endings (junctures). However, the denominator above can be simply approximated by the word-to-syllable ratio of the data, which was found to be close to 0.6 for all the datasets. Therefore we ignore the distinction between $p(B|d = y)$ and λ_y.

It follows from all this that

$$\begin{aligned} p(Y > y) &= p(Y > 1|Y \geqslant 1)p(Y > 2|Y \geqslant 2) \cdots p(Y > y|Y \geqslant y-1) \\ &= \prod_{k=0}^{y}(1-\lambda_k) \\ \Rightarrow F(y) &= 1 - \prod_{k=0}^{y}(1-\lambda_k) \end{aligned}$$

So we can derive an estimate for the distribution function $F(y)$ via estimates of λ_y, from the original estimates of $p(B|d)$, and vice versa. The formula can be inverted using

$$\lambda_y = \frac{1 - F(y)}{1 - F(y-1)} \tag{3.1}$$

although some smoothing is required as $F(y)$ increases towards 1, when this fraction is very sensitive to small rounding errors.

Figure 3.2 shows graphs of the probability density functions for the phrase length r.v. Y, for three data from three different domains. No common type of probability distribution provides of sufficiently good fit to these functions (of those investigated, the closest was the Poisson distribution). However, to make an effective adaptation using a small amount of data from the new domain, we wish to reduce the distribution to a very small number of degrees of freedom. We therefore make the assumption that the underlying "shape" of the distribution is common to all domains, and that a linear transform to link any two can be found. This means that, given well-trained estimates for the phrase length distribution of training data, Y_{trn}, this can be transformed to one appropriate to a new domain using

$$Y_{new} = aY_{trn} + b$$

In fact, we know that $b = 0$ since all phrase length distributions are constrained so that $F(0) = 0$. Thus $\hat{a}$, the estimate of the true value for a can be simply determined using a method of moments, taking expectations of both sides:

$$\begin{aligned} \mathbb{E}(Y_{new}) &= a\mathbb{E}(Y_{trn}) \\ \Rightarrow \hat{a} &= \frac{\hat{\mu}_{trn}}{\hat{\mu}_{adp}} \end{aligned}$$

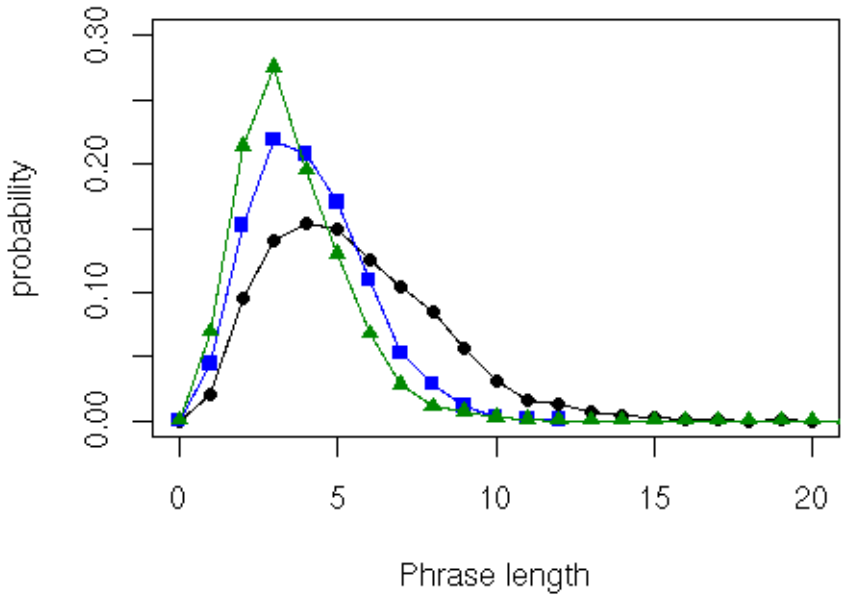


Figure 3.2: Example graph showing estimated phrase length probability density functions for three separate data sets

where $\hat{\mu}_{trn}$ and $\hat{\mu}_{adp}$ are estimated means derived from the observed phrase length distributions of the training data and adaptation data respectively.

Having estimated $\hat{a}$, we carry out the following:

1. Make the discrete distribution function reliably estimated from training data, $\hat{F}_{trn}(y)$, continuous, by means of cubic spline interpolation.
2. Calculate $\hat{F}_{new}(y) = \hat{F}_{trn}(y/\hat{a})$
3. Obtain, from this function, values of $\hat{\lambda}_y$ using (3.1) (smoothed for values of y above around 10), which can be used as estimates for $p(B|d = y)$ for the new domain.

A major advantage of this method is that it avoids problems with data sparsity at higher d, where there are fewer training cases. If there is a limited amount of training data for a domain, $p(B|d)$ is nevertheless likely to be estimated accurately for low d, as there will still be a large number of training cases (one for every phrase that is at least length d). However, accuracy will rapidly diminish for higher d. The advantage of using the limited training data to adapt a well-trained phrase-length model, rather than to construct one independently, is that it all training cases are used equally in the estimation of the average phrase length; the well-trained model, adjusted for this average, can be used to obtain a reliable estimate throughout the required range of d.

The performance of the adapted models can be compared to that of the originals by calculating the perplexity on the test set. For a test set of junctures J, the entropy of a phrase-length model, consisting of probabilities $p(B|d)$, is given by

$$\begin{aligned} H(J) &= \mathbb{E}(-log_2\, p(J)) \\ &= \mathbb{E}(-\sum_i log_2\, p(j_i|d_i)) \end{aligned}$$

from which the perplexity is found as

$$\text{perplexity}(J) = 2^{H(J)}$$

A rule of thumb used in ASR is that the language model perplexity is proportional to the square of the word error rate. We investigate the correlation between perplexity and break classification performance in these experiments.

3.2.3 HMM Parameter Adaptation: POS-sequence model

The POS-sequence model concerns the estimation of $p(C_i|j_i)$, where C_i represents the parts-of-speech of words surrounding the i^{th} juncture and j_i represents the type of the juncture. Given estimates well trained on one domain, our adaptive procedure attempts to re-estimate the probabilities for a new domain, given a limited amount of data from that domain. We work via an estimates of $p(j_i|C_i)$ since from this $p(C_i|j_i)$ can be derived, using equation 2.5:

$$p(C_i|j_i) = \frac{p(j_i|C_i)}{p(j_i)}$$

The denominator $p(j_i)$ should be consistent with the phrase-length model used.

Our POS-sequence model uses a C4.5 tree to group together the C_i and the probabilities are estimated at each leaf node. The distribution is discrete and unordered, so it is not possible to parametrise it, as we have done with the phrase-length probabilities above and as is done with observation probabilities used by HMMs in ASR.

It might be possible to make adaptations of the decision tree structure itself in order to re-estimate the distribution. However, this is likely to be extremely difficult due to the fact that decision trees are not constructed globally, rather by a series of local branching decisions based on the data available at each node. We do not consider this further. Instead we focus on producing, for each context C_i a probability estimate appropriate to the new domain, $p_{new}(j_i|C_i)$, based on the well-trained estimate from the original domain, $p_{trn}(j_i|C_i)$ and the estimate from adaptation data, $p_{adp}(j_i|C_i)$. We investigate two alternative methods.

Confidence-based approach

Here we take $p_{new}(j_i|C_i)$ to be weighted average of the probability estimates from the training and adaptation set.

$$p_{new}(C_i|j_i) = \frac{1}{w_{trn} + w_{adp}}(w_{trn}.p_{trn}(j_i|C_i) + w_{adp}.p_{adp}(j_i|C_i))$$

We choose the weights w_{trn} and w_{adp} according to the measures of confidence in each estimate. Two issues are presented:

1. How to calculate a measure of confidence in each estimate

2. How to calculate the weights, based on this confidence measure.

To resolve the first of these we assume that the number of breaks at at a particular leaf of a tree is a binomial random variable, $X = Bin(n, \theta)$, where n is the number of cases at the leaf and $\theta = p(B|C_i)$. This provides the justification for the estimation formula shown in (2.4):

$$\begin{aligned} \hat{p}(B|C_i) = \hat{\theta} &= \frac{X}{n} \\ &= \frac{\#B \text{ at } L}{total \text{ cases at } L} \end{aligned}$$

(where L is the leaf corresponding to case C_i).

One possible measure of confidence in the estimate $\hat{\theta}$ would be given by the variance of the probability estimate

$$\begin{aligned} var(\hat{\theta}) &= var(\frac{X}{n}) \\ &= \frac{1}{n^2} var(X) \\ &= \frac{\theta(1-\theta)}{n} \end{aligned}$$

but this is not much use, as to use this formula, we need to approximate θ by its estimated value, $\hat{\theta}$. This causes major problems when $\hat{\theta}$ is close to 0 or 1 – for example, if $\hat{\theta} = 0$ (which would be the case if no breaks were observed), the estimate for the variance would be 0, which clearly can't be correct!

An alternative measure of confidence is to use the width of an exact binomial confidence interval for any fixed confidence level (say 75%). This works more reliably for probability estimates close to 0 and 1. To calculate this, we used an algorithm by Sauro and Lewis [18] from a formula originally derived by Clopper and Pearson [4]. The formula is complex and we do not go into details here.

We use the reciprocal of the confidence interval width as our measure, so that a higher measure figure indicates a greater confidence. For each C_i, we calculate confidence measures from the relevant leaf of each of the training-data probability-tree and the adaptation-data probability-tree.

We investigate several schemes for choosing the weights w_{trn} and w_{adp} as a function of the confidence measures. Whilst we aim to assign greater weight to a probability estimate with greater confidence, we also consider that, due to coming from the same domain, the adaptation data is likely to be more representative of the test data than is the foreign training data. The latter will is, however, likely to produce more reliable estimates. Some possible schemes are:

- Set each weight to be equal to the respective confidence measure. This could lead to under-use of the adaptation data.
- Use the estimate from the adaptation data, unless its confidence measure is below a certain level, below which, adjust the weights linearly with the confidence measure.
- Use the estimate in which there is most confidence.

Perplexity-based approach

We modify an approach used by Jianfeng Gao et al for adaptation of N-gram language model. Again, we take $p_{new}(j_i|C_i)$ to be weighted average of the probability estimates from the training and adaptation set, but using a single parameter, α:

$$p_{new}(C_i|j_i) = (1-\alpha).p_{trn}(j_i|C_i) + \alpha.p_{adp}(j_i|C_i))$$

α remains constant – unlike the confidence-based approach, it does not vary with the probability being estimated.

We aim to choose the value of α to minimise the perplexity of the new POS-model on the test set. This should maximise the F-score. From Gao et al's work, we expect that as α is increased from 0 to 1, initially the test set perplexity will fall as the POS-model becomes more appropriate to the test set, but will eventually rise as the model becomes over-fitted to the adaptation data. (Perplexity calculated on the adaptation data would be expected to fall continually as α is increased).

A method is needed of estimating the test-set perplexity of the model – it would of course be "cheating" to use the test set itself for this purpose! Again,

following Gao et al, we estimate this figure using a technique known as "jack-knifing". We divide the adaptation data into 5. Removing one partition in turn, we estimate a POS-model using the full training data and the remaining 80% adaptation data, and then obtain the perplexity on the removed data. An average of the five perplexity figures is calculated after every partition has been removed. Since figures have all been calculated on "unseen" data not used to train the models, they should approximate the perplexity on the actual test set. We repeat this procedure for values of α ranging between 0 and 1.

3.2.4 Domain Classification

We investigate a crude version of the Classification method of adaptation used for ASR, whereby the adaptation data is used to select the best model set from a range of different speaking styles. Usually in ASR each model set is trained on several speakers, grouped together according to similarity in individual model parameters. This approach is not practical here due to the very limited number of training domains we have available.

Nevertheless, it possible to use to technique to a limited extent. We take the adaptation data from one domain, D, and calculate, for each foreign domain, the performance on this data of models well-trained on the foreign domain. We then "cluster" this model with the foreign domain. To predict breaks on D, we simply use the models trained on the foreign domain with no adjustment. We compare their performance with the performance of models derived from the limited adaptation data. This gives some idea of the usefulness of the method.

For this to work well, a good range of models must be available, so that at least one of similar to the new domain. Unfortunately this is not the case here. Instead, we simulate these conditions by choosing, for our test, a domain which is known to be much more similar to some foreign domains than others, for the domains for which we have data.

Chapter 4

Experimental Setup

4.1 Corpora

4.1.1 Boston University Radio News Corpus

The Boston University Radio News Corpus (BURNC) is a corpus created for speech synthesis research, and is described in detail by Ostendorf et al [12]. The corpus consists of US-English speech from 7 newsreaders from the WBUR radio station. All the speech is fully transcribed, and words are automatically tagged with parts-of-speech from the Penn Treebank tagset. POS-tags were hand corrected for a subset of each speaker's data (around 20% in the case of F2B). From this, tagging error was found to be small, at around 2%.

Much of the speech is prosodically-annotated by human labellers using the ToBI labelling scheme. Inter-annotator agreement for phrase-break markers was reported to be 95%, ignoring paragraph boundaries (which are certain to coincide with intonational phrase boundaries), and merging neighbouring break classes when annotators used an uncertainty marker. Ostendorf et al note that there are examples in the corpus where the phrase boundaries marked do not coincide with syntactic boundaries, showing that annotators did not purely rely on syntax for their boundary labelling.

The radio announcers recorded in the corpus are divided into categories A and B: A for announcers who normally read news live, and B for those who normally pre-record and edit their stories. Stories read by type B announcers

tend to be longer and more in-depth. From this corpus, we use data solely from the female category-B speaker, F2B. The main reason for this is because it is the only data set with full (and reliable) prosodic annotation, including phrase-break information – for the other speakers only a subset of recorded speech has yet been prosodically-labelled, insufficient for training, testing and comparing models for individual speakers. To provide sufficient contrast with models trained from the Toshiba Corpus for adaptation experiments, it is helpful to have each dataset as homogeneous as possible, which is why we do not group together data from other speakers with F2B to make a large single BURNC dataset.

There are other advantages to using the F2B set. Because the speaker is category B, the utterances are more lengthy, providing a greater contrast with the Toshiba corpus. Also, many other researchers who have used the Boston corpus for prediction tasks, such as Fach [5] and Fordyce [7] have used the F2B set alone. Choosing the same set allows easier comparison with their results.

The F2B set consists of 166 utterances. There are 13,075 junctures in total, of which 9,137 are non-breaks (ToBI labels 0-2) and 3,938 are breaks (ToBI labels 3 and 4), so 30.1% of junctures are breaks.

4.1.2 The Toshiba US-English Female TTS Voice Corpus

The Toshiba US-English Female TTS Voice Corpus has kindly been made available for use in this project by the Speech Technology Group at Toshiba Research Europe Limited. The corpus consists of recordings from one female speaker, grouped into sets according to the type of material ("domain"). Examples are:

Domain	Sets
Declaratives	A-J
Exclamations	K
Questions	Q
ICE Corpus sentences	L-O
Sentences for in-car navigation	X-Z

For each utterance, we were provided with the following:

- the transcription, including punctuation
- POS-tags (using an internal Toshiba tagset)

- prosodic break tags, corresponding to ToBI levels 1 (no break), 3 and 4

and other data such as stress markers and higher-level syntactic information, less relevant to this work.

Figures relating to some of the different utterance types are shown in the following table.

	A-J	L-O	X-Z
Number of utterances	990	392	300
Total number of junctures	8189	7905	2081
Number of breaks	3197	2848	833
Number of non-breaks	4992	5057	1248
Proportion of breaks	39.0%	36.0%	40.0%

Since most of the models we use take POS-tags as input features, it is necessary to use a consistent tagset for all experiments. There is no one-to-one mapping between the Toshiba internal tagset and the Penn tagset, so re-tagging of either the BURNC or the Toshiba corpus is required. Since we have no means of tagging with the Toshiba tagset, we re-tag all the Toshiba data with Penn tags using the freely available MXPOST tagger [16], which comes pre-trained on the Penn tagset. We do not attempt to investigate which of the sets is the more useful as input features for break prediction.

4.1.3 Division of Data

Training and Test Data

In common with many other similar experiments in this field, we divide the full set of data corresponding to a particular domain or speaker into 90% training data and 10% test data. We do this by blind selection, designating every 10th utterance as test data, leaving the remainder as training data. This aims to avoid having any systematic differences between the two sets, whilst maintaining the order of junctures within an utterance (this is not required for C4.5 prediction, or for the chink-chunk algorithm, but is for the phrase-model in the HMM-based predictor).

We make this division for all data sets. This allows us to compare the performance of adapted models with the performance of models well-trained on native data.

Adaptation Data

To test an adaptation method, we train models for one domain on the 90% data set, as normal – these are considered to be "well-trained", since they are usually at the limit of the performance increases that can be achieved simply by adding more training data.

To adapt this to the new domain, we generally use the 10% set from the new domain as adaptation data – this can be varied with the size of the data set – with the remaining data from the new domain being used to test the performance of the adapted models. This simulates a situation in which there is little data available from the new domain.

The four domains used are the F2B read news from the Boston Corpus; and the A-J, L-O and X-Z sets of material from the Toshiba Corpus. These all provide sufficient data for accurate training and testing.

4.2 Performance measures

As our principal evaluation metric we adopt the F score. This combines the measures of *precision* and *recall* commonly used in natural language processing. A parameter β is used to weight the relative importance of the two measures. Here, precision, P, is the proportion of break predictions that prove to be correct predictions out of all the junctures predicted as breaks. Recall, R, is the proportion breaks in the data that are correctly predicted. This means, for example, that an algorithm that classified every juncture as a break would have recall of 100%, but would be likely to have a very low precision. The F score with parameter β is given by the formula

$$F_\beta = \frac{(\beta^2 + 1).P.R}{\beta^2 P + R}$$

In common with other work, we set $\beta = 1$, so that the F score is the harmonic mean of the precision and recall. This has been used as the principal performance

measure by, for example, Busser et al, Schmidt and Atterer, and Ingulfsen. The latter also used the BURNC for break prediction, making a direct comparison of results particularly helpful.

Alternative metrics have been used, however. One scheme, inherited from ASR, is to compare the predicted sequence of junctures with the actual sequence, computing the number of insertions, deletions and substitutions required to transform one sequence to the other. Various measures can be calculated from these figures. Taylor and Black took this approach, as did Read and Cox.

Some would argue that, since the main purpose of break prediction is to produce more natural, intelligible synthetic speech, the only measure that is really of importance is a perceptual score. However, this is by its nature highly subjective, and time-consuming to carry out; also there is no standard measure that could be used to make comparisons between systems. It has been suggested that from a perceptual standpoint, precision is a more important than recall since if a TTS system fails to predict a break, it is possible for the listener to infer the syntactic structure of the sentence from other syntactic clues; whereas a falsely-inserted break will cause confusion. No experiments have yet proved this theory.

Maintaining a high F-score, whatever β is chosen, requires a suitable balance to be struck between precision and recall. It would be relatively easy to optimise F_β for any particular value of beta, were there compelling evidence to indicate the best value – precision can generally be increased by being more cautious in predicting a juncture to be a break, whilst recall can be increased by the opposite. The difficult task is increasing one whilst minimising the reduction in the other. In the absence of a clear opinion about the relative merits of the two, maximising the F-score with $\beta = 1$ provides a worthy challenge.

4.3 Tools Used

The main tool used was Quinlan's C4.5 decision tree classifier, which is open-source. In addition to using the program in its distributed form, we made substantial modifications to the code (in C) to

- return log probabilities from the decision tree;

- use different classification heuristics, such as maximising a reward function; and
- load multiple decision trees simultaneously.

We made no modifications to the tree-growing part of the program.

We implemented our Viterbi decoder from scratch in C++, linking to functions from C4.5. About 1000 lines of C++ code were written. Most data processing tasks were carried out by writing Python scripts, as was the compilation of statistics for the the phrase-models for the Viterbi decoder. About 1500 lines of script were written in total. Code is included in a separate appendix.

In addition to the above, we used a syllable-counting routine written by Greg Fast in Perl and Adwait Ratnaparkhi's MXPOST statistical POS tagger [16]. For the manipulation and graphing of the phrase-length models discussed in (3.2.2), the GNU project's R language was used.

Chapter 5

Experiments

5.1 Performance of Natively-Trained Models

5.1.1 Preliminary Experiments

We ran baseline experiments using the chink-chunk algorithm and C4.5 classifier. The most basic experiments used as inputs features from the 3-tag "CFP" set distinguishing between content words, function words and punctuation This was discussed in 2.3.1, with table 2.1 giving the set of function words. The chink-chunk algorithm uses, for each juncture, the type of word immediately prior to the juncture, POS_{n-1} and the type immediately following the juncture, POS_{n+1}, using the CFP set. For all other algorithms, the tag POS_{n-2} from two words before the juncture was also included, giving a POS trigram.

The C4.5 classifier was tested using different input features: POS trigrams from the CFP tagset; POS trigrams from the full Penn Treebank tagset; and with a pair of trigrams, one from each set. Trees were trained and tested on the same domain with the 90%/10% data split. It should be noted that the aim here was not to produce the best possible results, but to provide an acceptable baseline next to which the later adapted models could be judged.

Results for the Boston F2B speaker are shown in table 5.1, and for the Toshiba A-J set in table 5.2 (note all scores are in %). From these it was clear that the better performance could be achieved using POS trigrams from the full Penn tagset. These features were used for subsequent experiments. Adding the CFP

Experiment	Precision	Recall	F-Score
Chink-chunk	67.0	73.4	70.1
C4.5 - CFP tagset	68.7	71.9	70.3
C4.5 - Penn tagset	83.3	76.1	79.5
C4.5 - CFP & Penn	83.1	76.2	79.5

Table 5.1: Preliminary results on the F2B dataset. All scores are in %.

Experiment	Precision	Recall	F-Score
Chink-chunk	75.0	59.1	66.1
C4.5 - CFP tagset	76.9	58.1	66.2
C4.5 - Penn tagset	88.4	76.6	82.1

Table 5.2: Preliminary results on the Toshiba A-J dataset.

trigram to the feature set, giving six features overall, made almost no change to these results. Similar results were found for the L-O and X-Z domains.

5.1.2 Experiments Using the Viterbi Decoder

	Decoder (words)			Decoder (syllables)			C4.5
Set	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**F-Score**
F2B	82.2	82.0	**82.1**	80.9	82.0	**81.4**	**79.5**
A-J	87.3	79.7	**83.3**	85.5	81.2	**83.3**	**82.1**
L-O	82.8	76.9	**79.8**	80.9	79.3	**80.1**	**76.5**
X-Z	77.9	75.9	**76.9**	79.5	83.5	**81.5**	**82.5**

Table 5.3: Results for the Viterbi decoders on all domains, compared with C4.5 results

Table 5.3 shows the performance achieved by the Viterbi decoders. The observation probabilities for the models were derived from a C4.5 tree with a Penn tagset POS trigram as the input features, as for the best C4.5 models above. Two

different decoders were built. There first has a phrase-length model based simply on the number of words between breaks; the second has measures phrase length in terms of syllables.

There is no consistent trend in the relative performance of the different methods. However, the C4.5 classifier is out-performed by both Viteri decoders in 3 out of 4 cases. Simple averages of the F-scores over the four domains are 80.5%, 81.6% and 80.2% for the word-based decoder, the syllable-based decoder, and C4.5 respectively. These figures are very close: however, it should be borne in mind that a large majority of breaks are easy to predict, being indicated by punctuation and content/function word distinctions. This is illustrated by the relatively high performance of the basic chink-chunk algorithm. A consequence of this is that small gains in F-score are more significant than they might appear.

5.2 Performance on Foreign Data

Experiments were carried out using models trained on 90% data from one domain, but tested on data from alternative domains. The aim of this was to determine the potential for testing adaptation techniques between the different domains. It was hoped that models trained on a foreign domain would perform poorly compared to natively trained models, thus requiring adaptation methods to improve performance.

Performance figures for models trained on the Boston F2B data set and tested on the Toshiba sets are shown in table 5.4. Figures for models trained on the Toshiba A-J set and tested on the two other Toshiba sets are shown in table 5.5. In each case, figures from models trained on the same domain are shown for comparison.

Since the F2B data is from a completely different corpus and different speaker, it was expected that the greatest differences in performance would be found using models trained on this data, compared to models trained on the Toshiba data. This was found to be correct: performance using the foreign F2B-trained models, measured by F-score, was lower on average by by 6.5%, 5.2% and 5.6% using C4.5, the decoder (words) and the decoder (syllables) respectively. Whereas the A-J models tested on the other Toshiba domains gave average reductions of 3.7%,

Set	Experiment	Precision	Recall	F-Score	Native F-Score
A-J	C4.5	89.6	64.7	**75.1**	**82.1**
	Decoder (words)	85.7	68.0	**75.8**	**83.3**
	Decoder (syllables)	87.1	69.5	**77.3**	**83.3**
L-O	C4.5	85.7	60.0	**70.6**	**76.5**
	Decoder (words)	80.1	66.0	**72.3**	**79.8**
	Decoder (syllables)	82.3	68.3	**74.7**	**80.1**
X-Z	C4.5	93.4	63.9	**75.9**	**82.5**
	Decoder (words)	90.2	66.2	**76.4**	**76.9**
	Decoder (syllables)	90.6	65.5	**76.1**	**81.5**

Table 5.4: Results on Toshiba data using models trained on F2B. Natively trained F-Score (the score using models well-trained an the same domain) given for comparison

Set	Experiment	Precision	Recall	F-Score	Native F-Score
L-O	C4.5	81.9	67.4	**73.9**	**76.5**
	Decoder (words)	77.2	73.5	**75.3**	**79.8**
	Decoder (syllables)	78.1	77.7	**77.9**	**80.1**
X-Z	C4.5	88.9	69.0	**77.7**	**82.5**
	Decoder (words)	86.2	73.7	**79.5**	**76.9**
	Decoder (syllables)	87.2	73.5	**79.8**	**81.5**

Table 5.5: Results on for models trained on the Toshiba A-J data, tested on other Toshiba domains. Natively trained F-Score given for comparison

1.0% and 2.9% respectively. It is clear from these results that there *is* potential for testing adaptive techniques between these domains (especially between the Boston corpus and Toshiba corpus) in order to reduce the gap between native and foreign performance.

On the foreign domains, the HMM models with Viterbi decoder consistently out-performed the C4.5 models. The decoder using phrase length measured in syllables out-performed, by an average of 1.3%, the decoder using phrase length measured in words. The higher performance of the HMMs is likely to be due to the fact that they provide phrase length constraints, which C4.5 does not. These constraints become more important if the POS-based prediction is performing poorly. This is illustrated by the high-precision figures in the tables, particularly for the C4.5 models, which is symptomatic of under-prediction of breaks. The phrase-length models used by the decoders increase the probability of a break, the longer the distance since the previous one. This helps to reduce this under-prediction.

5.3 Adapted Models

5.3.1 C4.5 trees with transformations

We present results obtained by adapting a tree trained on a large amount of foreign data (the 90% set), by making selected transformations on the basis of a small amount of native adaptation data (the 10% set). The theory was discussed in 3.2.1. The aim was that predictions using the transformed tree should be higher than both the original foreign tree and a tree trained using just 10% of the data. The results for three experiments are shown:

1. C4.5 'second classifier' used to classify each case with any one of tB, fB, tN and fN. Juncture class only changed from B to N if fB predicted, and from N to B if fN predicted.

2. As above, but classification based on a reward function. Best results were achieved when the reward for a correct transform was equal to the penalty for an incorrect transform. These are the results shown.

3. Cases classified with the restriction that if a break is predicted by the initial classifier, then the second classifier can predict only tB or fB; and if a non-break is predicted, the second classifier can predict only tN or fN.

We also investigated reward-based classification with the restriction in (3), but results were almost totally unchanged from those using (2).

Experiment	**Precision**	**Recall**	**F-Score**
Native tree	88.4	76.7	82.1
Native tree - trained 10%	86.7	68.3	76.4
F2B tree	89.6	64.7	75.1
Experiment 1	83.9	71.9	77.4
Experiment 2	85.8	70.0	77.1
Experiment 3	81.9	75.3	78.5

Table 5.6: Results on A-J set

Experiment	**Precision**	**Recall**	**F-Score**
Native tree	86.1	68.9	76.5
Native tree - trained 10%	80.0	65.9	72.3
F2B tree	85.7	60.0	70.6
Experiment 1	81.0	68.3	74.1
Experiment 2	81.1	67.2	73.5
Experiment 3	77.4	71.8	74.5

Table 5.7: Results on L-O set

Results are shown in tables 5.6, 5.7 and 5.8 with initial trees trained on the F2B set – considered to the most dissimilar to the other three. A sample of the 'second classifier' output for the L-O test data is reproduced here showing the correct classes against the predicted classes. We aim to maximise the number of fB and fN correctly classified (12 and 222 respectively in this example) when the transforms will improve performance, whilst minimising the number of tB classified as fB (7) and the number of tN classified as fN (168) when the transforms will worsen performance.

Experiment	Precision	Recall	F-Score
Native tree	81.5	83.5	82.5
Native tree - trained 10%	82.3	70.8	76.1
F2B tree	93.4	63.9	75.9
Experiment 1	79.9	77.3	78.6
Experiment 2	82.4	77.1	79.6
Experiment 3	81.8	79.2	80.5

Table 5.8: Results on X-Z set

```
 (a)  (b)  (c)  (d)    <-classified as
---- ---- ---- ----
1460    7         92    (a): class tB
 212   12         37    (b): class fB
  40        222  776    (c): class tN
  51    1   168 4126    (d): class fN
```

The results show that, whilst none of the adaptation techniques is able to improve performance up to the level of the well-trained native C4.5 trees, all manage to improve upon the individual performance of both the foreign C4.5 tree, trained on the F2B speaker, and a native tree, trained using just the 10% adaptation data. Out of the three experiments, experiment 3, where predictions are restricted according to the outcome from the F2B tree, was consistently the best.

5.3.2 Phrase length model adaptation

Results are shown for our experiments of adapting the phrase-length models (the values of $p(B|d)$) used by the Viterbi decoder. In each case observation probabilities (the POS-sequence model) were derived from the F2B data using C4.5. A series of experiments were performed: firstly we trained phrase-length models on the actual 90% data that was used for testing. This provided a figure for the likely maximum performance that could be achieved by this method. Models

were then trained on the remaining 10% data from the same domain, and compared with models based on the F2B phrase but adapted using the 10% data. Performance was found to be very similar to the first experiment, presumably because a relatively small amount of data is required for the probabilities to be accurately estimated. Consequently, in order to test the adaptation technique itself a third set of experiments were carried out using a much smaller set of adaptation data, 1% of the A-J set (66 junctures) and 5% of the smaller X-Z set (102 junctures). Syllable-based phrase models were used throughout: preliminary results showed that these showed much higher potential for improvements by using natively-trained models, and consistently achieved higher performance overall.

Tool	**Phrase Model**	**Precision**	**Recall**	**F-Score**
Decoder (words)	Self	80.9	73.2	76.9
Decoder (syllables)	Self	81.1	76.9	79.0
	Native 10%	79.7	78.4	79.0
	Adapted 10%	81.7	76.4	78.9
	Native 1%	76.4	78.8	77.6
	Adapted 1%	81.2	76.5	78.8

Table 5.9: Results on the A-J set for HMMs with output distributions trained on F2B data, and various phrase-length models.

Tool	**Phrase Model**	**Precision**	**Recall**	**F-Score**
Decoder (words)	Self	84.6	72.0	77.8
Decoder (syllables)	Self	83.4	74.5	78.7
	Native 10%	84.3	72.8	78.1
	Adapted 10%	86.0	72.3	78.5
	Native 5%	79.8	72.9	76.2
	Adapted 5%	85.0	74.3	79.3

Table 5.10: Results on the X-Z set for HMMs with output distributions trained on F2B data, and various phrase-length models.

Results are for the A-J domain and X-Z domain are shown in tables 5.9 and 5.10 respectively. It can be seen that even when using only a small amount of adaptation data the adaptation technique can improve the performance to a level close to that achieved by training on the phrase-models on the test set itself. The F-score figures of 78.8 and 79.3 on the respective domains compare favourably to the 75.1 and 75.9 achieved by the F2B-trained C4.5 trees.

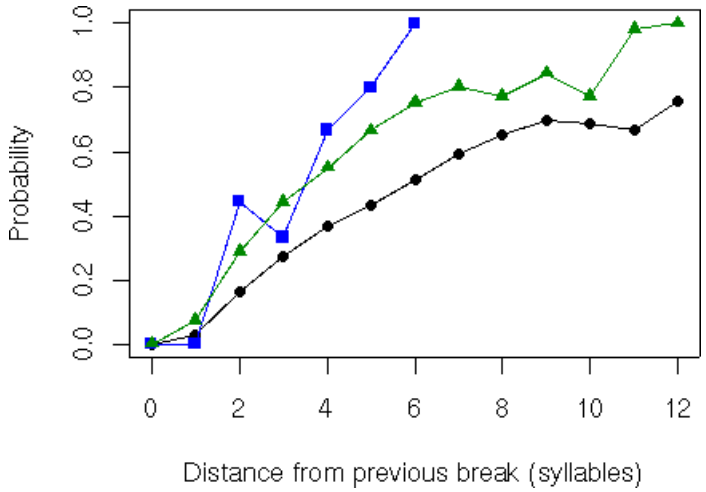


Figure 5.1: Graph showing the probability of the current phrase ending, $p(B|d)$, estimated from F2B data (black dots), 1% A-J data (blue squares) and F2B data, adapted using 1% A-J data (green triangles)

Phrase Model	**A-J**	**X-Z**
F2B	1.810	1.920
Self	1.707	1.708
Native 10%	-	1.726
Adapted 10%	1.709	1.724
Native 1%/5%	-	-
Adapted 1%/5%	1.709	1.715

Table 5.11: Perplexity figures for various phrase models, measured on the A-J and X-Z test sets.

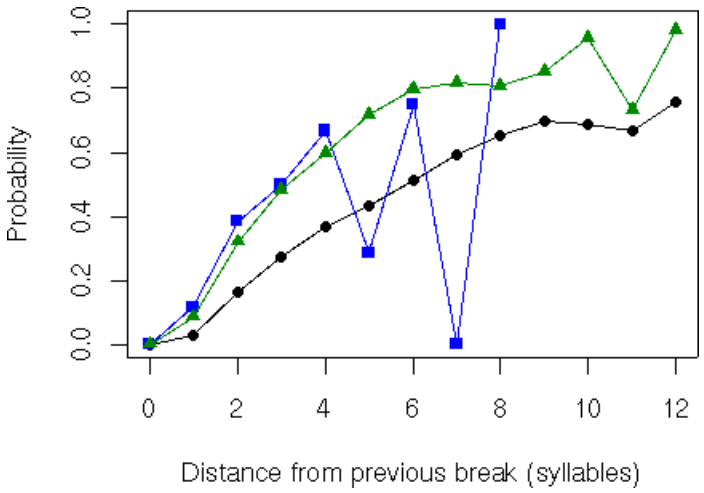


Figure 5.2: Graph showing the probability of the current phrase ending, $p(B|d)$, estimated from F2B data (black dots), 5% X-Z data (blue squares) and F2B data, adapted using 5% X-Z data (green triangles)

When using the 10% adaptation data, the adaptation technique shows little improvement over using the data directly to estimate break probabilities, as discussed above. However, when small datasets are used, the technique does show an improvement, of 1.2 with 1% A-J adaptation data and 3.1 with 5% X-Z data. Graphs of the probabilities are shown in figures 5.1 and 5.2. Note that the probability graphs trained on a small amount of data are much less smooth, illustrating the inaccuracies in estimation.

Table 5.11 shows perplexity figures for the various phrase models used, and provides a good example of the inverse relationship between perplexity and F-score. Note that the maximum perplexity for a phrase-length model is 2 (B and N both equally likely for every d) and the minimum is 1 (one of N or B certain to occur at every juncture).

Further experiments were carried out on model adaptation from the A-J set to the X-Z set. Table 5.12 shows results for models with observation probabilities estimated from the A-J data and tested on the X-Z set, with various phrase-length models. The figure of 82.1 using a self-derived phrase model is close to the maximum score of 82.5 achieved on the set using models entirely trained on X-Z

Phrase Model	Precision	Recall	F-Score
A-J	87.2	73.5	79.8
Self	86.4	78.2	82.1
Native 5%	82.5	75.5	78.8
Adapted 5%	89.3	71.1	79.2

Table 5.12: Results on the X-Z set for HMMs with output distributions trained on the A-J data, and various phrase-length models. All results use a phrase model based on syllable count.

data. This suggests that most of the difference between the two domains is in the average phrase lengths, rather than the POS-sequence models. There was less improvement using a phrase-length model derived from the A-J set, adapted using the 5% X-Z adaptation data. However, the use of a phrase model adapted using the same data from the well-trained F2B model (as used in an earlier experiment), yielded an F-score of 80.7. This may reflect the fact that much more data was available to train the F2B phrase model to begin with.

5.3.3 POS-Sequence Model Adaptation

Results are shown for our experiments of adapting the POS-sequence models (the values of $p(C_i|j_i)$). Initially we used phrase-length models from the F2B set, to use a baseline result for the experiments based solely on foreign training data. However, after initial experiments had been carried out, we investigated the performance improvement to be gained by combining both POS-sequence and phrase-length adaptation in together. As in earlier adaptation experiments, we used the Viterbi decoder with syllable-based phrase-length models.

The first experiments were performed on the A-J set with models adapted from F2B with 10% native adaptation data. As with the phrase-length model experiments we later reduced the size of the adaptation data set, to 5% of the total, to better observe the efficacy of the adaptation technique itself.

Tables 5.13 and 5.14 show results for the A-J domain for the confidence-based techniques and perplexity-based techniques respectively. Table 5.13 shows the

Model	Precision	Recall	F-Score
F2B data only	87.1	69.5	77.3
10% data	86.3	73.1	79.2
Linear confidence weighting	87.8	71.1	78.6
Maximum confidence weighting	87.7	70.5	78.1

Table 5.13: Results on the A-J set using POS-sequence models adapted from the F2B set using the confidence-based method

Model	Precision	Recall	F-Score
10% adaptation data			
$\alpha = 0.0$	87.1	69.5	77.3
$\alpha = 0.6$	88.0	72.1	79.3
$\alpha = 1.0$	86.3	73.1	79.2
5% adaptation data			
$\alpha = 0.0$	87.1	69.5	77.3
$\alpha = 0.45$	88.2	71.2	78.8
$\alpha = 1.0$	85.9	71.5	78.0
5% data with native phrase model			
$\alpha = 0.0$	84.1	74.1	78.8
$\alpha = 0.45$	85.4	75.6	80.2
$\alpha = 1.0$	82.5	75.4	78.8

Table 5.14: Results on the A-J set using POS-sequence models adapted from the F2B set using the perplexity-based method

most successful of a series of confidence-based experiments. These results are not encouraging, in every case being lower than models trained entirely on the 10% adaptation data. Turning to the perplexity-based models, perplexity graphs were produced for varying α, the factor determining the relative weight to be allocated to the adaptation data and foreign training data ($\alpha = 1$ gives full weight to the adaptation data). Graphs for the 10% set are shown in figure 5.3. It can be seen that the estimate of test-set perplexity is close to that actual figure. Based on this $\alpha = 0.6$ was chosen as the optimal weighting factor. However, using this value resulted in only a small improvement in F-score. This is probably because the variation in perplexity with α over its full range is actually very small, and there appears to be no major over-fitting problem for high α. To better investigate the adaptation method, a smaller, 5% data set was used, where greater problems with over-fitting would be expected. Note also from the graph that the self-perplexity graph for the adaptation data begins to increase for increasing α, when it would be expected to decrease monotonically. It is not clear why this should be the case.

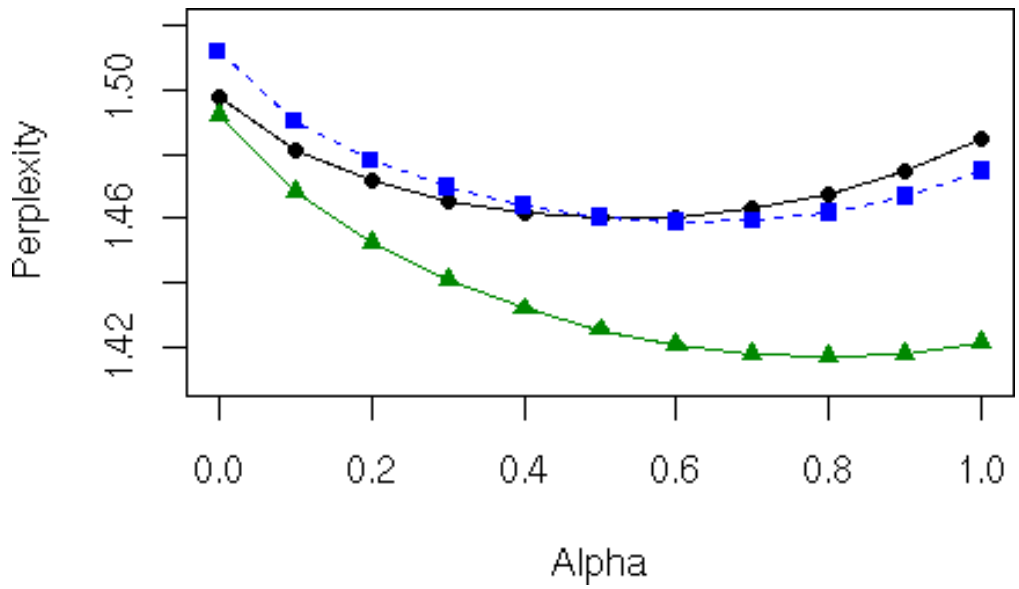


Figure 5.3: Graph showing variation in perplexity with α with F2B training and 10% A-J adaptation data: estimated test-set perplexity (black dots), actual test-set perplexity (dashed blue squares), adaptation-set perplexity (green triangles)

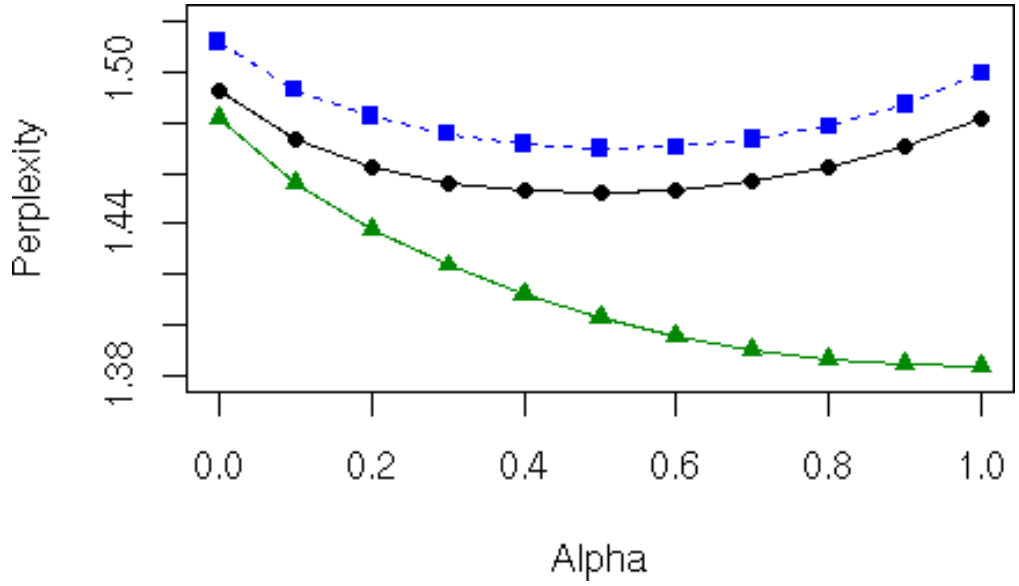


Figure 5.4: Graph showing variation in perplexity with α with F2B training and 5% A-J adaptation data: estimated test-set perplexity (black dots), actual test-set perplexity (dashed blue squares), adaptation-set perplexity (green triangles)

The perplexity graph for the 5% adaptation data showed a clearer trend towards over-fitting for high α, and a monotonically decreasing curve for ythe adaptation data self-perplexity. Corresponding, the optimal value of α chosen was lower: 0.45. This resulted in a more significant increase in performance over both the model trained entirely with the adaptation data, and the model trained entirely with the foreign data, as shown in table 5.14. The same trend was displayed in a further experiment using a phrase-length model derived from the same data, in 5.3.2. The F-score of 80.2 was higher than the scores obtained making the POS-model and phrase-model adaptations individually, and was overall the highest adapted score achieved on the A-J domain using adaptation data. Using 10% adaptation data, the score was increased further to 80.7. These figures compare to a baseline of 75.1 using C4.5 trained entirely on F2B data, and a top native score of 83.3 obtained using the Viterbi decoder with models trained on the full 90% A-J data set.

The experiments were repeated for the X-Z set using 10% adaptation data and

Model	Precision	Recall	F-Score
F2B data only	90.6	65.5	76.1
10% data	84.6	69.9	76.5
Linear confidence weighting	91.7	65.9	76.7
Maximum confidence weighting	89.6	68.3	77.5
Maximum confidence - adapted phrase model	85.1	72.0	78.0

Table 5.15: Results on the X-Z set using POS-sequence models adapted from the F2B set using the confidence-based method

Model	Precision	Recall	F-Score
10% adaptation data			
$\alpha = 0.0$	90.6	65.5	76.1
$\alpha = 0.4$	91.3	66.7	77.2
$\alpha = 0.6$	90.5	66.0	76.4
$\alpha = 1.0$	84.6	69.9	76.5
10% data with native phrase model			
$\alpha = 0.0$	87.5	70.6	78.1
$\alpha = 0.4$	88.7	72.8	80.0
$\alpha = 0.6$	88.4	73.5	80.2
$\alpha = 1.0$	80.1	75.2	77.6

Table 5.16: Results on the X-Z set using POS-sequence models adapted from the F2B set using the perplexity-based method

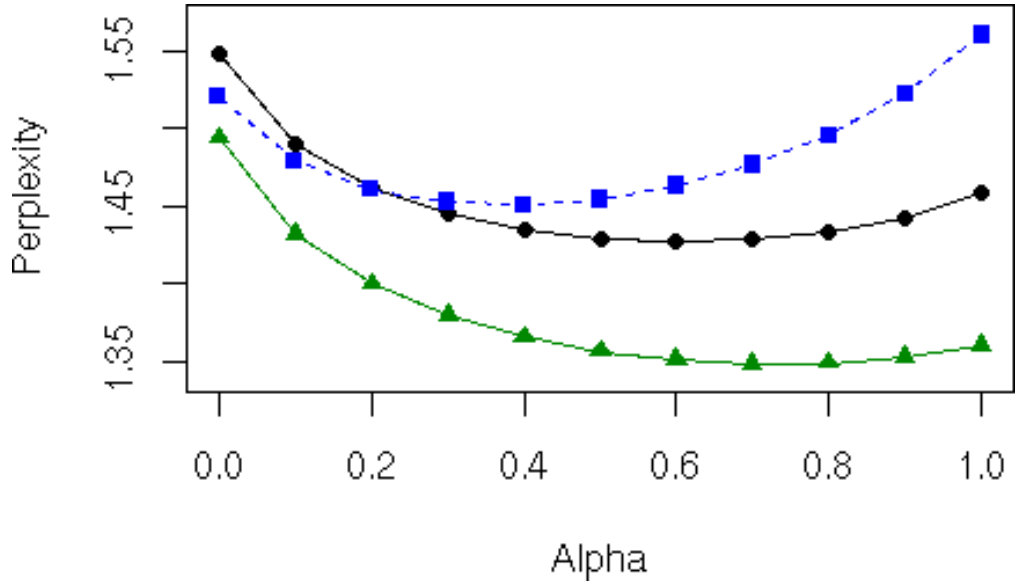


Figure 5.5: Graph showing variation in perplexity with α with F2B training and 10% X-Z adaptation data: estimated test-set perplexity (black dots), actual test-set perplexity (dashed blue squares), adaptation-set perplexity (green triangles)

are shown in tables 5.15 and 5.16. In this case, the estimate of test-set perplexity proved to poor, as can be seen from figure 5.5. The graph of the estimate indicated a minimum of perplexity at $\alpha = 0.6$, whereas it actually seemed to be closer to 0.4. This was demonstrated in the F-scores, which were at a maximum at $\alpha = 0.4$. A possible cause of this problem the much smaller size of the adaptation data set compared to that for the A-J domain, leading to inaccurate perplexity estimates using the partition-removal method.

The confidence-based method, where for each C_i, the estimate with the highest confidence is chosen, produced the highest F-score in the POS-model experiments on this set, 77.5. However, on experiments using this method with an adapted phrase-length model (using the same X-Z adaptation data). the gain was small: just 0.5. The perplexity-based method appeared to be more effective when used with the adapted phrase-length model, giving, for $\alpha = 0.6$, a score of 80.2.

Overall, the results presented in this section are not entirely conclusive. Whilst the perplexity-based method does generally appear to yield improvements, there

remain anomalous results, in particular concerned with the test-set perplexity estimation procedure. Also it is not clear why some self-perplexity curves should be upward-sloping.

One likely problem is connected with the back-off procedure employed for estimation of $p(j_i|C_i)$. This is necessary because of the situation where a break/context pairing is found in the test set that has not appeared in the training or data. In this case, the perplexity is undefined. A back-off procedure is required because a non-zero probability must be applied to the case to avoid a meaningless perplexity figure being produced. It is possible that our back-off procedure, whilst avoiding meaningless figures, does distort the perplexity calculation in some way – further research might be useful to determine a method of applying a probability to unseen cases to minimise this distortion. Also, the use of a back-off procedure may give a higher F-score when used with a small amount of adaptation data that would otherwise lead to an over-fitted model. This could lead to baseline scores higher than they it would otherwise be. Whilst a good thing it itself, this makes it difficult to gauge the effectiveness of the perplexity-based technique in isolation.

5.3.4 Domain Classification

Whilst of limited value when the number of available domains is small, as is the case here, the potential of this technique can be illustrated by conducting experiments on the L-O data set. This domain has empirically been found to be similar to the A-J domain, as is demonstrated below.

Firstly, the performance of foreign models on the L-O adaptation data was measured. Results are shown in table 5.17. The HMM Decoder used was the Viterbi decoder with syllable-based phase-length models. From this it is seen that of the foreign models, those from the A-J domain achieve the highest F-score using both C4.5 and the Viterbi decoder. Results using models trained on 90% L-O data are shown for comparison.

According to the classification technique, the A-J set would be chosen as the most similar, and models from this set would be used to make break predictions for any further data from the L-O domain. The performance of these models on

Method	Data	Precision	Recall	F-Score
C4.5	F2B	87.9	61.0	72.0
	A-J	83.8	68.1	75.2
	X-Z	68.7	73.7	70.9
	L-O	86.1	68.9	76.5
Decoder	F2B	83.3	69.3	75.7
	A-J	78.7	79.3	79.0
	X-Z	69.4	79.7	74.2
	L-O	80.9	79.3	80.1

Table 5.17: Results on the L-O adaptation data using well-trained models from various domains

the remaining L-O data (the 90% set) is shown in table 5.18. It can be seen that, in common with their performance on the adaptation data, the well-trained A-J models achieve the highest F-scores once more, for both prediction algorithms. The performance of models trained directly on the L-O adaptation data (a limited amount of data) is shown for comparison. The A-J models out-perform the poorly-trained L-O models by 1.6 using C4.5 and 2.7 using the decoder. This demonstrates the success of the classification technique: given a good range of well-trained models from different domains, it can be beneficial to use limited adaptation data to select a domain and use the models from that, even if no more advanced adaptation techniques are carried out on them.

Tool	Data	Precision	Recall	F-Score
C4.5	F2B	85.7	60.0	70.6
	A-J	81.9	67.4	73.9
	X-Z	68.2	71.7	69.9
	L-O	80.0	65.9	72.3
Decoder	F2B	82.4	68.4	74.7
	A-J	78.2	77.7	78.0
	X-Z	68.7	77.4	72.8
	L-O	76.6	74.2	75.3

Table 5.18: Results on L-O test data using models well-trained on various domains. Results with models trained on L-O adaptation data shown for comparison.

Chapter 6

Conclusion

6.1 Summary and discussion

We have presented two algorithms for data-driven prosodic phrase-break prediction, and several methods for adapting them for a different type of speech or spoken material, using a limited amount of adaptation data from the new domain.

The prediction algorithms we implemented were to some extent novel. Whilst their design was largely based on earlier research, no single previous work has brought together the components in the same way. The performance of the algorithms, particularly the HMM Viterbi decoder, compares favourably with previous results. For example, the highest F-score achieved by Ingulfsen on the Boston corpus was 77.9, compared to 82.1 here. The highest score achieved by Busser et al on data from the MARSEC corpus, was 78.3. Our highest overall result was 83.3 on the Toshiba A-J set.

Most adaptation techniques we investigated were found to be effective, to some extent, when measured against performance of both models trained on foreign data and models trained directly on the adaptation data. In general, the adaptation was found to result in a greater improvement when a smaller amount of adaptation data was used. Here is a brief summary of the key results.

- **C4.5 transformations**: In each case, the adaptation was found to improve on F-scores from models trained on the F2B, set by 2.1, 2.2 and 4.4 for the

A-J, L-O and X-Z domains respectively, and on scores from models trained on the adaptation data by 3.4, 3.9 and 4.6 respectively.

- **Phrase model adaptation**: In both the domains where models were adapted from F2B, adaptation resulted in improvements over the models trained on the foreign data and models trained on adaptation data, given a sufficiently small adaptation data set. In the former case, improvements in F-score achieved were 1.5 and 3.1 for A-J and X-Z respectively; in the latter case, improvements were 1.2 and 3.1 respectively.

- **POS-sequence model adaptation**: Improvements were less consistent. Testing on the A-J set showed increases in F-score of 2.0 over the foreign models, but just 0.1 over the natively-trained model. The only method found to yield improvements over both on the X-Z set was not effective on the A-J set. However, the perplexity-based approach *was* found to be effective when combined with an adapted phrase model.

- **Domain classification**: A limited investigation showed that performance could be improved by using adaptation data to select an appropriate domain from which to use more well-trained models.

Overall the C4.5 transformation method was found to give the highest improvements on average. However, it should be noted that the baseline scores on foreign or limited tended to be lower for these models, whereas the Viterbi decoder was found to be more robust in this situation due to the addition of phrase-length constraints. Also the adaptation of phrase-length and POS-sequence components of the HMM-based predictions had the advantage that they could combined together to give higher improvements.

6.2 Further work

The data sets used here were not as heterogeneous as we would have liked: reductions in performance due to training on data from a different set were relatively small – no more than a fall in F-score of about 9% on the best models. A greater

drop would have probably resulted in a wider spread of results using the different methods, giving a greater contrast between them. The only variations between the different domains as recorded were in the speaker and type of material – more interesting results might have been generated had we had we had access to material recorded with a range of different speaking rates or different emotions. It would have been particularly interesting to investigate the potential use of adapted phrase-length models for controlling tempo in synthesised speech, which could offer an improvement over existing methods such as those used by Trouvain [21].

We did not attempt to investigate a modification of the "speaker normalisation" technique used in ASR. This might have involved the identification of systematic differences in break labelling POS-tagging between the different corpora. One option for normalising break labelling would be to avoid the subjectivity inherent in the ToBI hand-tagging by attempting to identify breaks using signal processing techniques. This would, however, require substantial further research. For POS-tagging, we could perhaps have taken account of the uncertain nature of each tag prediction, incorporating the probability of a given tag being correct directly into our statistical models. We instead made the simplifying assumption that all tags were correct, on the basis of the reporting tagging accuracy.

Finally, we have already discussed the appropriateness of using F-score as our main evaluation metric, given uncertainty as to the most perceptually relevant measure. Most of our experiments gave a higher precision than recall. However, our results show how recall can be increased by adapting phrase length models to favour shorter phrases – for example, the shift from the F2B phrase model to an A-J phrase model shown in figure 5.1. It would not be difficult to adjust the models to favour whichever statistic was considered most perceptually important. Unfortunately we were not able to use a TTS system to conduct listening tests on the phrasing predictions of our models.

Bibliography

[1] J. Bachenko and E. Fitzpatrick. A computational grammar of discourse-neutral prosodic phrasing in English. *Computational Linguistics*, 16(3):155–170, 1990. 2.2.1

[2] Alan Black and Paul Taylor. Assigning phrase breaks from part-of-speech sequences. In *Proceedings of Eurospeech '97*, pages 995–998, 1997. 2.2.1

[3] Bertjan Busser, Walter Daelemans, and Antal van den Bosch. Predicting phrase breaks with memory-based learning. In *Proceedings 4th ISCA tutorial and research workshop on speech synthesis*, pages 29–34, 2001. 2.2.1

[4] C.J. Clopper and E. Pearson. The use of confidence intervals for fiducial limits illustrated in the case of the binomial. *Biometrika*, 26:404–413, 1934. 3.2.3

[5] Marcus Fach. A comparison between syntactic and prosodic phrasing. In *Proceedings of Eurospeech '99*, volume 1, pages 527–530, 1999. 1.1, 4.1.1

[6] Greg Fast. Algorithm for counting syllables, 1999. 2.3.3

[7] C. Fordyce and M. Ostendorf. Prosody predicition for speech synthesis using transformational rule-based learning. In *Proceedings of International Conference on Spoken Language Processing*, 1998. 1.1, 4.1.1

[8] M.J.F. Gales and P.C. Woodland. Speech processing II. CSTIT lecture notes, 2005. 3.1

[9] J.P. Gee and F. Grosjean. Performance structures: a psycholinguistic and linguistic appraisal. *Cognitive Psychology*, 15:411–458, 1983. 2.2.1

[10] T. Ingulfsen. Influence of syntax on prosodic boundary prediction. Master's thesis, Cambridge University, 2004. 2.2.1

[11] M. Liberman and K. Church. *Advances in Speech Signal Processing*, chapter Text Analysis and Word Pronunciation in Text-to-Speech Synthesis. Marcel Dekker, 1991. 1.3, 2.2.1

[12] M. Ostendorf, P. Price, and S. Shattuck-Hufnagel. The Boston University Radio News Corpus. Boston University, 1995. 4.1.1

[13] M. Ostendorf and N. Veilleux. A hierarchical stochastic model for automatic prediction of prosodic boundary location. *Computational Linguistics*, 20(1), 1994. 2.2.1

[14] Janet B. Pierrehumbert. *The Phonology and Phonetics of English Intonation*. PhD thesis, MIT, 1980. 2.1

[15] J. Ross Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, 1993. 1.3, 2.2.2

[16] A. Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proceedings of the Emperical Methods in Natural Language Processing*, 1996. 4.1.2, 4.3

[17] Ian Read and Stephen Cox. Using part-of-speech for predicting phrase breaks. In *Procedings of 8th International Conference on Spoken Language Processing*, volume 1, 2004. 2.3.2

[18] J. Sauro and J.R. Lewis. Estimating completion rates from small samples using binomial confidence intervals: Comparisons and recommendations. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2005. 3.2.3

[19] Helmut Schmidt and Michaela Atterer. New statistical methods for phrase break prediction. In *Proceedings of the 20th International Conference on Computational Linguistics*, 2004. 1.3, 2.2.1

[20] K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirschberg. ToBI: A standard for labelling English prosody. In *Procedings of the International Conference on Spoken Langauge Systems*, volume 2, pages 867–870, 1992. 2.1

[21] Jürgen Trouvain. Tempo control in speech synthesis by prosodic phrasing. In *Procedings "Konferenz zur Verarbeitung natrlicher Sprache"*, pages 227–230, 2002. 6.2

[22] Michelle Q. Wang and Julia Hirschberg. Automatic classification of intonational phrase boundaries. *Computer Speech and Language*, 6:175–196, 1992. 1.1, 2.2.1

[23] Yung Hwan Oh Yeon Jun Kim. Prediction of prosodic phrase boundaries considering variable speaking rate. In *Proceedings of the International Conference on Spoken Language Processing*, pages 1505–1508, 1996. 3.2.2

Appendix A

Code Used

Here follows printouts of the code written for this project. This includes code written for the Viterbi decoder together with any code from C4.5 that was substatially modified for use in the decoder.

Code is not included in the PDF version of this report.