Methods for Optimal Text Selection

Jan P. H. van Santen[1] *Adam L. Buchsbaum*[2]

[1]Lucent Technologies – Bell Labs, 600 Mountain Ave., Murray Hill, NJ 07974, U.S.A., jphvs@research.bell-labs.com

[2]AT&T Labs, 180 Park Ave., P.O. Box 971, Florham Park, NJ 07932-0971, U.S.A., alb@research.att.com

ABSTRACT

Construction of both text-to-speech synthesis (TTS) and automatic speech recognition (ASR) systems involves usage of speech data bases. These data bases usually consist of read text, which means that one has significant control over the content of the data base. Here we address how one can take advantage of this control, by discussing a number of variants of "greedy" text selection methods and showing their application in a variety of examples.

1. INTRODUCTION

Both automatic speech recognition (ASR) systems and text to speech (TTS) systems have components that are trained on text—typically read text. Surprisingly often, training text is selected without giving much thought to optimality of the selected text. For limited domain situations, it may very well suffice to select randomly a subset from the domain for training purposes. In many ASR applications, and certainly in most TTS applications, however, the domain is open. And, as discussed at length in [7], in open domain situations the issue of *coverage* is vital, because by any measure differences between sub-domains are too profound to generalize with any confidence from one sub-domain to another. For example, [7] mentions that only 50 per cent of triphones occurring in two large text corpora occurred in both corpora.

In this paper, we first define the concept of coverage via the concept of "unit" and then describe several techniques for optimizing coverage.

2. COVERAGE AND "UNITS."

Text can be analyzed at many different levels, corresponding to different types of unit. In ASR, an example of a unit is a phonemic-context sensitive phone, such as an /i/ in a /p/_/t/ context, or a sub-word unit roughly corresponding to some phone sequence.

In TTS, duration components typically receive as input feature vectors that describe the identity, identities of neighbors, and prosodic context of a given phonetic segment. Assignment of pitch accent types to words often involves vectors describing the lexical identity of the word in question, its location in the phrase, and parts-of-speech tags of surrounding words. The intonation component may use vectors describing features associated with a syllable, including its segmental makeup, lexical stress, and pitch accent type of its associated word.

In each of these cases, the input domains are discrete but extremely large. For example, using parts-of-speech tags in a window of five words in combination with five location positions creates a space of potentially one hundred million units.

Whether one should attempt to cover all units in the domain, or only the most frequently occurring units, depends on the number of units, the cost of training, and the cost of missing units. The latter point is somewhat subtle. As remarked in [7], there are often cases where the frequency distribution is extremely uneven, but where the number of rare units is sufficiently large that their combined probability mass makes it quite likely that even a small text sample will contain at least one rare unit. Hence, any system has to be *prepared* for rare units.

But being prepared does not necessitate that all units be *included* in the training sample. In TTS, for example, many components involve rules that, to the degree that they are accurate, allow generalizing from seen cases to unseen cases. And in ASR, the recognition outcome is the result of several probabilistic constraints of which the acoustic analysis forms only one component—other components being bi-gram models, vocabulary, and syntactic constraints. Because of these *error correction mechanisms*, one may not need good acoustic model estimates for each phone in any context. One might only need good estimates for frequently occurring units.

There is a risk, however, in relying too much on frequencies observed in a particular corpus, because of the instability of frequency distributions across text corpora. We already mentioned that triphone occurrences differ sharply between text corpora. These differences persist when we look at frequencies of triphones that are shared between two corpora, with typical correlations of only 0.30 or less between their frequencies. The reason for this instability is elementary. Even if the underlying frequency distributions are identical, the expected frequencies of most units are small due to the unevenness of the common distribution, resulting in many observed frequencies of zero, one, or two. Which particular units occur once or twice in one corpus, but not in the other, has a large chance component.

And of course, underlying frequency distributions can never be assumed to be identical. For example, even very large corpora (such as the Associated Press Newswire in 1987) have units (e.g., n-phones contained in the name "Reagan") that for good historical reasons have less chance of occurring in similar corpora (such as the Associated Press Newswire in 1997).

In summary, text corpora, no matter how large, should be viewed as samples of a larger, abstractly defined domain (e.g., the written English language). Tying system construction too closely to a particular corpus runs the risk of neglecting units that may prove unexpectedly frequent in new corpora. Unless one's system has rules that allow generalization (as in TTS) or error correction mechanisms (as in ASR), coverage of all units is the goal.

3. BASIC GREEDY ALGORITHM.

We now discuss algorithms for optimizing coverage. The best-known algorithm is the *greedy algorithm* as applied to the set-covering problem [3]. Consider a set of sentences, and a parallel set that contains for each sentence a list of diphones occurring in the corresponding sentence. How do we select a small set of sentences so that their corresponding diphones contain each diphone in the larger list at least once? This problem has a well-known approximate solution in the form of the greedy algorithm. This algorithm successively selects sentences. The first sentence is the sentence with the largest diphone type count; all diphones occurring in that sentence are removed from the larger list. Once N sentences have been selected, the next sentence selected is the sentence with the largest type count of the remaining diphones.

We applied this basic algorithm for a perceptual experiment, in which we wanted listeners to process the smallest number of sentences that yet contained each acoustic inventory element at least once [6]. Starting with a list of over 67,440 sentences, the greedy algorithm found a subset of 650 sentences with complete coverage of all (2533) elements in the larger list.

As mentioned, it may be meaningful in training acoustic models in ASR to optimize frequency-weighted coverage of, e.g., triphones. The change in the greedy algorithm is simple: instead of maximizing the type counts, one maximizes the frequency-weighted type counts. There is an additional change that can be made. Often having only one token of a given triphone is not sufficient for adequate modeling—one needs at least, say, five tokens. This can be accomplished by associating with each triphone a counter, initialized to 5, and subtracting 1 each time a sentence is selected containing that diphone. We found that, for example, a training set of 15,000 names can cover (at 94 per cent frequency weighted coverage) the triphones in 169,328 personal names at least 5 times; random samples of 15,000 names cover only 72 per cent. The advantage of greedy selection over random sampling increases as the number of units decreases and the size of the training sample (here: 15,000) relative to the domain (here: 169,328) decreases.

Paradoxically, we found that applying weights that are the inverse of frequencies can improve performance in situations where complete coverage is feasible. This scheme focuses the algorithm on sentences with rare units; in the process, the more frequent units are picked up *en passant* so that the algorithm does not have to search for additional sentences containing these units. In applications of this procedure, we obtained reductions in training text size of up to 10%.

4. FEATURE SPACES AND SUB-VECTORIZATION.

Up to now, we have treated units as atomic. However, in many cases units are vectors. For example, in duration prediction, input units often have the form

$$< /p/,\ IsStressed,\ \ldots,\ PhraseFinal > .$$

Is there some way to make use of this factorial structure?

Consider applications where the feature space is mapped on some acoustic variable, such as segmental duration. There is a set $F = \{1, \ldots, N\}$, for some N, of *factors*. For each $i \in F$, the factor F_i is a set $\{F_1^i, \ldots, F_{\zeta_i}^i\}$ of $\zeta_i = |F_i|$ distinct *levels* or *features*. For example, one factor might be the phone itself. The levels would be the set of possible phones. Another factor might be whether the phone is stressed, and the levels would be the set of possible stress values. The *feature space* $\mathcal{F}$ is defined by $\mathcal{F} = F_1 \times \cdots \times F_N$. Each individual phonetic segment occurrence ρ corresponds to a feature vector $\vec{f}(\rho) = (f_1, \ldots, f_N) \in \mathcal{F}$, where $f_i \in F_i$ for $1 \le i \le N$.

One way this mapping may be characterized is as a sum of *terms*, each of which depends on one or more of the factors, but typically not on all factors. Specifically, let K be a subset of 2^F such that the duration (for example) of a feature vector $(f_1, \ldots, f_N)$ can be predicted by a sum of parameters:

$$D(f_1, \ldots, f_N) = \sum_{I \in K} S_I(f_{I_1}, \ldots, f_{I_{|I|}}) + \mu \qquad (1)$$

where $I = \{I_1, \ldots, I_{|I|}\}$ for $I \in K$; μ is some constant. In the *Analysis of Variance* [4], the S_I terms have no particular form, but are subject to the zero sum constraint:

$$\sum_{f_{I_j} \in F_{I_j}} S_I(f_{I_1}, \ldots, f_{I_{|I|}}) = 0,\ 1 \leq j \leq |I|,\ \forall I \in K. \tag{2}$$

In *sums-of-products models*, the S_I terms have the form:

$$S_I(f_{I_1}, \ldots, f_{I_{|I|}}) = \prod_{j=1}^{|I|} S_{I,I_j}(f_{I_j}),\ \forall I \in K. \tag{3}$$

Regardless of the specific form of these equations, the S_I terms in Eq. 1 allow modeling of *interactions.* Two factors are said to interact when the effect of one factor (as measured by comparing two different levels on that factor holding all other factors constant) depends on other factors. A well-known example is the effects on vowel duration of postvocalic voicing and phrasal location: in phrase-final locations, the lengthening effect of postvocalic voicing is much larger (on a log scale) than in phrase-medial locations.

Not all factors interact. For example, the effects of phrase-final lengthening are the same for stressed and unstressed syllables, when measured on a log scale.

It is obvious that in order to estimate the interaction terms from data, our training data must contain all combinations of levels on the factors that occur in a single interaction term. For example, if $F = \{1, 2, 3, 4\}$ and $K = \{\{1, 2, 3\}, \{2\}, \{2, 4\}\}$, then we need all combinations of factors $\{1, 2, 3\}$, and $\{2, 4\}$, but not of $\{1, 2, 3, 4\}$, $\{3, 4\}$, and $\{1, 4\}$.

A sufficient condition for *estimability* of interaction terms (more about which in the next section) is that the training data cover the entire feature space, but it is obviously not necessary. After all, the numbers of estimated parameters are typically small (e.g., in the above example assuming 5 levels per factor, we have to estimate fewer than 200 parameters) compared to the number of data points (typically 50 phone durations per sentence). Hence, we should be able to estimate parameters on far fewer sentences than are required to cover the entire feature space.

We propose that in situations where one has a *good sense of which interactions should be of no concern*, the following procedure can be used:

1. Transcribe each phone in the text corpus by the usual vector, $\vec{f}(\rho) = (f_1, \ldots, f_N)$.
2. Generate from each individual vector a set of sub-vectors, where each sub-vector corresponds to some S_I term. In the above example, the sub-vectors would be (f_1, f_2, f_3), (f_2), and (f_2, f_4).
3. Proceed with the usual greedy algorithm, applied to these sub-vectors.

This idea was applied to Mandarin Chinese [5]. It was found that the 8,233 distinct sub-vectors occurring in a data base of 15,630 sentences could be covered completely in just 427 sentences. Coverage of the original feature space would have required thousands of sentence. The training data were found to be sufficient for parameter estimability. In fact, standard errors of estimate were quite small, due to the even distribution of data points over parameters and to the large ratio of observations to parameters (better than 100:1). Statistical analysis of the data using sums-of-products models yielded reliable parameter estimates, and resulting predicted segmental durations were found to be quite accurate.

5. MODEL-BASED GREEDY SELECTION.

The sub-vectorization method works well in practice and is extremely easy to implement, but it has two formal weaknesses. First, it is mathematically quite possible that even when all sub-vectors of the types (f_1, f_2, f_3), (f_2), and (f_2, f_4) are covered, parameters of the corresponding model still cannot be estimated. The reason is that for estimability one needs the presence of some combinations of factors across these three groups. Thus, covering all sub-vectors is not sufficient for parameter estimability.

Second, it is not necessary either, because under the constraints described in Eqs. 2 and 3, we do not need all combinations of all levels of factors contained in the interaction term.

We now describe a new algorithm that finds the smallest number of sentences that is guaranteed to be sufficient for parameter estimation.

5.1. Linear Parameter Estimability and Design Matrices

We briefly describe here standard theory of linear estimation for the Analysis-of-Variance model.

A feature vector $\vec{f}(\rho)$ corresponding to a given phonetic segment occurrence ρ can be uniquely represented as a compound row vector, in which each sub-vector encodes the level on the corresponding factor. One way to do this is to have the vector component corresponding to the level in the factor set equal to 1 and the remaining components equal to 0. Usually, the last level is represented as a vector of all -1's. For example, for $F_1 = \{$*primary stress, secondary stress, unstressed*$\}$, *primary stress* is mapped onto $(1, 0)^t$, *secondary stress* onto $(0, 1)^t$, and *unstressed* onto $(-1, -1)^t$,

The design matrix for a sentence s simply consists of the a matrix $X(s)$ whose rows are computed as indicated. The design matrix for a corpus C is a vertical stack of matrices $X(s)$, where s ranges over C.

If we let $\vec{D}(C)$ be the corresponding vector of observed durations, then it is known [4] that the parameter vector $\vec{\mathcal{P}}$ is estimable if and only if the matrix $X(C)$ is of full column rank, in which case the estimate of $\vec{\mathcal{P}}$ is given by:

$$\vec{\mathcal{P}} = \left(X(C)^t X(C)\right)^{-1} X(C)^t \vec{D}(C) \tag{4}$$

Note. The concept of design matrix cannot be used directly for sums-of-products models. We conjecture that it is possible to work out a similar solution using Jacobian matrix rank [1], but we have not proven this. In practical terms, however, we found it always to be the case that if data are sufficient for estimating the parameters of an Analysis-of-Variance model, then they are also sufficient for estimating the parameters of the corresponding sums-of-product model. But we do not doubt that counterexamples can be found.

5.2. The Optimization Problem

The optimization problem can now be formulated as follows: Find a subset of sentences $C' \subseteq C$ such that the corresponding design matrix still has full column rank, and such that C' is of minimal cardinality.

Elsewhere [2], we provide a detailed description of the algorithm. The algorithm is a greedy algorithm, and starts with the sentence whose corresponding design matrix has the largest rank. Once n sentences have been selected, for the next sentence we select the sentence whose design matrix produces the largest increase in rank when added to the stack of design matrixes constructed up to that point. Thus described, this process is extremely computationally expensive, because it necessitates rank computations of as many matrices as there are sentences at each step. Our algorithm, however, performs an incremental Gram-Schmidt orthogonalization procedure that obviates these rank computations.

Based on experiments on a 150 MHz R4400 processor on an SGI Challenge machine with 1 Gb of main memory, we estimate that large data sets (e.g., 5,000,000 sentences with each 50 phones, and 500 to-be-estimated parameters) can be computed in a few days of CPU time. Since this algorithm is typically used as part of a one-time off-line training procedure, this is not a practical problem.

The most important result obtained is that when applied to the same Mandarin Chinese data set as in section , *two* sentences were found to be sufficient for estimation of all (30) vowel parameters. Even when the standard error of estimate is bound to be large when so few data points are available per parameter, repeated application of the selection procedure can produce data sets that are still quite small (e.g., 50 sentences) yet produce uniformly small standard errors.

In summary, our intuition that for estimation of 30 parameters one should not need significantly more than one or two 50-phone sentences proved to be correct, and the proposed algorithm is sufficiently fast that large-scale applications are realistic.

6. CONCLUSIONS

This paper discussed methods for optimal text selection that can be used for training and assessing both TTS and ASR systems. In our lab, we now routinely use these methods.

The main contribution of this paper is two-fold. First, we hope to have made researchers more aware of the importance and feasibility of intelligent text selection procedures. Second, the linear model-based text selection procedure showed that extremely efficient text can be selected if one (1) is willing to make strong assumptions about model structure and (2) is able to mathematically tie the model to a greedy algorithm. In our algorithm, this tie was provided by the design matrix, which is a very simple concept. But it seems worth exploring if similar mathematical ties can be discovered in entirely different, perhaps more complicated types of models.

7. REFERENCES

1. D. Bamber and J. van Santen. How many parameters can a model have and still be testable? *Journal of Mathematical Psychology*, 29:443–473, 1985.
2. A. Buchsbaum and J. van Santen. Selecting training text via greedy rank covering. In *Proc. 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 288–95, 1996.
3. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachussetts, 1990.
4. C. R. Rao. *Linear Statistical Inference and its applications.* John Wiley & Sons, Inc., New York, 1965.
5. C. Shih and B. Ao. Duration Study for the AT&T Mandarin Text-to-Speech System. In J. van Santen, R. Sproat, J. Olive, and J. Hirschberg, editors, *Progress in Speech Synthesis.* Springer-Verlag, New York, 1996.
6. J. van Santen. Perceptual experiments for diagnostic testing of text-to-speech systems. *Computer Speech and Language*, 7:49–100, 1993.
7. J. van Santen. Combinatorial issues in text-to-speech synthesis. In *Proceedings*, Rhodos, Greece, 1997. Eurospeech.