JOINT ACOUSTIC UNIT DESIGN AND LEXICON GENERATION

M. Bacchiani *M. Ostendorf*

Electrical and Computer Engineering Department, Boston University, 8 St. Mary's St., Boston, MA, 02215, USA

ABSTRACT

Although most parameters in a speech recognition system are estimated from data by use of an objective function, the unit inventory and lexicon are generally hand crafted and therefore unlikely to be optimal. This paper proposes a joint solution to the related problems of learning a unit inventory and corresponding lexicon from data. The proposed algorithm performs comparably to a state-of-the-art phone-based system on a speaker independent read speech task with moderate vocabulary size.

1. INTRODUCTION

Large vocabulary speech recognition systems typically represent lexical entries in terms of sub-word units, for which acoustic models can be reliably estimated. Part of the system design is therefore to decide on a suitable unit inventory and define the mappings from lexical entries in the vocabulary to linear strings or networks of units (i.e. define the lexicon). This problem is simplified in most systems by using phone-based units and a hand-crafted lexicon. In order to limit the complexity of the recognition search, most systems use single linear strings for the majority of the lexical entries in the vocabulary and very few linear strings for the remaining entries. Although the parameters of the unit models are generally estimated from data using an objective function such as maximum likelihood (ML), no such function is used in the unit inventory and lexicon design. Given the lack of a clear objective in this part of the system design, the resulting unit inventory and lexicon are unlikely to be optimal in terms of the objective function used throughout the design of the rest of the system.

Even though the unit inventory and lexicon definition is suboptimal, the use of mixture distributions in the unit models together with parameter estimation techniques have proven successful in read speech tasks. When this system design algorithm is applied to spontaneous speech tasks however, severe performance degradation compared to read speech tasks is observed. Where mixture distributions in the unit models for read speech were able to capture the acoustic variability within the suboptimally defined units, the increased acoustic variability in spontaneous speech makes this approach problematic because of the increased confusability between units.

An alternative to manual derivation of a unit inventory and lexicon is to learn them from data. A unit derived in this way is generally referred to as an Acoustic Sub-Word Unit (ASWU). Over the last decade, a number of researchers have looked into this problem [1, 2, 3, 4] and found algorithms that automatically define model inventories and estimate unit model parameters. The related problem of defining a lexicon in terms of these ASWUs has also received attention, e.g. [5, 6]. To derive a lexicon of linear string pronunciations, all these approaches start by finding a number of candidate pronunciations for each lexical entry in the vocabulary based on the pronunciations seen across training tokens for the target lexical entry. These candidate pronunciations are then evaluated with all training tokens, and an objective function is used to determine the optimal candidate.

One problem with this approach is that the unit inventory and lexicon design problems are clearly related – acoustic models are no longer optimal after the pronunciations are restricted. This problem is easily addressed by iterative re-estimation of the acoustic model and the pronunciations, as in [7]. A bigger problem, not addressed in previous work, is that for tasks with a lot of pronunciation variability in the initial labeling of tokens, it is expensive to determine the optimal pronunciation among the large number of candidates and difficult to rule out cases when the vast majority occur only once. Furthermore, the optimal pronunciation may not even be among the observed candidates. Cases where a large number of candidate pronunciations per lexical entry can be expected include speaker-independent recognition, where pronunciation variability would be a consequence of speaker differences, and/or large vocabulary recognition where a large inventory of acoustic units (analogous to polyphonic HMM states) is needed.

Here we describe a joint unit inventory and lexicon design algorithm that addresses the problem of initial pronunciation variability by introducing lexical constraints into the unit inventory design. By designing the units and lexicon jointly, the derived unit models are matched to the lexicon. Section 2 describes the algorithm in detail. In section 3, experimental results on the DARPA Resource Management (RM) database are described. Finally, the results are discussed in section 4.

2. ALGORITHM

The two basic algorithmic steps of any unit inventory design algorithm are an acoustic segmentation followed by a clustering step to define the unit inventory, e.g. [1, 2, 7].

The key elements that differ in our approach are the use of pronunciation-related constraints in unit design, the consistent use of a maximum likelihood objective function, and progressive unit inventory and model refinement.

To jointly solve the unit inventory and lexicon design problem, the unit design algorithm described in [4] is modified such that it operates under a set of constraints associated with the lexicon. Assuming a linear pronunciation model, the unit inventory design is constrained to associate the same sequence of units with every token of a particular lexical entry. In particular, two constraints are introduced. After the acoustic segmentation, pronunciation length consistency is enforced such that all training tokens of a lexical entry contain the same number of segments. Then before clustering, a pronunciation consistency constraint is introduced by grouping the segments in the different training tokens of a lexical entry according to their position in the lexical entry, taking advantage of the pronunciation length consistency. After completion of the clustering step, the lexicon is implicitly defined, since the data from different training tokens representing the same position within a lexical entry are assigned to a single cluster. In addition, since the maximum likelihood objective function is used, the acoustic model parameters are also defined as a result of clustering. Section 2.1 describes the segmentation and clustering with the introduction of constraints in more detail.

Progressive unit refinement is important for at least three reasons. First, once data is clustered, the segmentation that the initial units were based on may no longer be appropriate. Section 2.2 describes how the unit model inventory and corresponding segmentation can be refined further using retraining. Second, even phone-based systems benefit from iterative clustering techniques for increasing the acoustic model complexity, and the analogous solutions for ASWUs are described in Section 2.3. Finally, there is the issue of matching the temporal resolution of units to the size of the unit inventory, which is addressed in Section 2.4.

2.1. Initial unit inventory and lexicon design

The initial inventory and lexicon design is a three step process. The first two steps provide a segmentation in which all training tokens of a lexical entry contain the same number of segments. Finally in the third step, the segments are clustered using a pronunciation consistency constraint to define the unit inventory and lexicon. The segmentation and clustering algorithms are implemented for polynomial mean trajectory segment models in general [4, 8], but for simplicity the experiments and equations given here correspond to the special case of a hidden Markov model, i.e. a constant mean trajectory.

The first step is **unconstrained acoustic segmentation**. The acoustic segmentation functions as an initialization of the algorithm. Taking an approach similar to that in [5], the maximum likelihood segmentation of the training data is found by use of dynamic programming (DP). Let x_t be a d-dimensional observation vector, such as a vector of cepstral coefficients representing a window of speech at time t. The unconstrained acoustic segmentation algorithm involves recursive updating for every time t and every allowable number of segments n:

$$\delta(t,n) = \max_{t-l_{max} \le \tau \le t-l_{min}} [\delta(\tau-1, n-1) + \log \mathrm{p}(x_\tau, \ldots, x_t | \mu_{\tau,t}, \Sigma)], \quad (1)$$

where l_{min} and l_{max} denote minimum and maximum segment lengths. In addition to updating $\delta(t,n)$ the index τ that maximizes equation 1 is stored allowing the most likely segmentation to be found in the end by tracing back. The (generalized) likelihood of the segments during the DP is computed using a multivariate Gaussian model with a single diagonal covariance Σ used for all the segments. This covariance matrix can be estimated either on a per utterance basis or from the entire training corpus. The mean parameter of the Gaussian model is computed from the hypothesized segments; the constant mean model corresponds to the assumption that speech is piecewise stationary. During segmentation, the likelihood increases monotonically with the number of allowable segments. We control the average number of segments by setting a fixed threshold on the average likelihood per frame, which controls the temporal resolution (average state duration) of the resulting system.

Next, we introduce a **pronunciation length constraint.** The acoustic segmentation is aligned with segmentation times of lexical entries. Each acoustic segment is assigned to the lexical-entry token with which it overlaps most. For each lexical entry, the median of the number of acoustic segments over all training tokens is used to define the pronunciation length. The training data is then segmented again using DP (eqn. 1) under the constraints that:

- Each lexical entry boundary coincides with an acoustic segment boundary.
- The number of acoustic segments for a lexical item is equal to the median pronunciation length.

In the resulting segmentation, all training tokens of a lexical entry have the same number of segments.

The final step is **maximum likelihood clustering.** The segments resulting from the second step of the algorithm are clustered to define the unit inventory. The clustering algorithm used here differs from that used in [2, 5, 7] in that maximum likelihood is used as an objective rather than minimum Euclidean distance. Specifically, the repartition step involves computing the likelihood of segments given the model parameters of a cluster, i.e. a negative log likelihood "distance". The cluster re-estimation procedure consists of finding the ML parameter estimates of a Gaussian distribution from the data contained in the cluster. Cluster centroids therefore directly represent unit models and clustering addresses both the inventory and model design problems, whereas in [2, 5, 7] unit model parameters had to be estimated in a separate step from the data partition defined by clustering.

Before clustering, the data is grouped to ensure pronunciation consistency. This grouping is implemented by computing a sufficient statistic for each collection of segments originating from different training tokens in the same position within a lexical entry. The sufficient statistics for the constant mean model are the sample mean μ_p and covariance Σ_p and the total number of vector observations

contained within the group N_p. These sufficient statistics are stored for each unique position within each unique lexical entry: if there are V entries in the vocabulary and the median pronunciation length is R, the data is grouped into VR groups. These sufficient statistics will be referred to as *atomic group sufficient statistics*. As the sufficient statistic representations of these atomic groups cannot be split in clustering, this grouping ensures the pronunciation consistency.

Let a group of K segment observations $\mathcal{X}_p = \{X^1, \cdots, X^K\}$, of lengths $\{L_1, \cdots, L_K\}$, have sufficient statistics (μ_p, Σ_p, N_p) where $N_p = \sum_{i=1}^{K} L_i$. The likelihood distance of the group with respect to the cluster with parameters μ_c and Σ_c is computed as

$$\mathcal{L}(\mathcal{X}_p|\mu_c, \Sigma_c) = \frac{N_p}{2}\left[D\log(2\pi) + \log(|\Sigma_c|) + \right.$$
$$\left. \text{tr}\left(\Sigma_c^{-1}\Sigma_p\right) + (\mu_p - \mu_c)^T\Sigma_c^{-1}(\mu_p - \mu_c)\right], \quad (2)$$

where the superscript T denotes a transposition.

Once observations are assigned to a cluster, the ML parameter estimate given P sets of observations are

$$\mu_c = \frac{1}{\sum_{q=1}^{P} N_q}\sum_{p=1}^{P} N_p\mu_p \quad (3)$$

$$\Sigma_c = \frac{1}{\sum_{q=1}^{P} N_q}\left[\sum_{p=1}^{P} N_p\left(\Sigma_p + \mu_p\mu_p^T - \mu_c\mu_c^T\right)\right] \quad (4)$$

Starting from a single cluster, the cluster inventory is increased by binary divisive clustering. Iteratively, the cluster with the lowest average likelihood per frame is selected to be split. Two new clusters are defined by first obtaining an initial split by perturbing the cluster mean, and then running a number of binary K-means clustering iterations using only the data that was contained in the original cluster before splitting. Clusters with fewer observations than a minimum occupancy threshold are not considered for splitting. After the cluster inventory is increased up to a heuristically determined inventory size, a number of K-means iterations using all the data and the full cluster inventory are run. If any cluster during this stage has fewer observations than the minimum occupancy threshold, the cluster is removed from the inventory and the data previously held within that cluster is repartitioned over the remaining clusters. The clustering algorithm derives the final unit model inventory by alternating between divisive and K-means iterations, increasing the number of clusters in stages.

The final data partition over the cluster inventory defines the lexicon by virtue of the data grouping. When neighboring segments within a lexical entry are assigned to the same cluster, the segments are collapsed into a single entry in the lexicon. As the units assume a constant mean uni-modal Gaussian distributions, repetitions of the same unit label in the lexicon are equivalent to a single instance of the label. Segments that were found distinct in acoustic segmentation can be found equivalent after the quantization introduced by clustering. During this step, some temporal resolution is lost relative to the temporal resolution of the acoustic segmentation.

2.2. Re-training

The initial acoustic segmentation was optimal given an unconstrained model inventory (size S for S segments), as model parameters in acoustic segmentation are derived separately for each instance of a segment in the DP search. After clustering, the acoustic space is quantized into C models with $C << S$, making acoustic segmentation suboptimal. To achieve a matched condition between the unit model inventory and the segmentation, a retraining algorithm can be used, either Viterbi or Baum-Welch. The Viterbi training algorithm, used here, iteratively re-segments the data to find the optimal segmentation given the current unit model inventory and then re-estimates unit model parameters using the new segmentation. Given a lexicon consisting of linear pronunciation strings derived by the algorithm described in 2.1, a lexical-entry-level transcription can be expanded into an S-length unit-level transcription $\{u_1, \cdots, u_S\}$. The segmentation step involves recursive updating for every time t and every unit index $i \in \{1, \cdots, L\}$ of

$$\Psi(t, i) = \max_{1 \le \tau \le t-1} \Psi(\tau - 1, i - 1) + \log \text{p}(x_\tau, \cdots, x_t|\mu_i, \Sigma_i) \quad (5)$$

where μ_i and Σ_i denote the mean and covariance of unit i. In addition to updating $\Psi(t, i)$ the index τ that maximizes equation 5 is stored allowing the most likely segmentation to be found in the end by tracing back. The unit model parameters are re-estimated from this new segmentation using standard ML estimation.

2.3. Increasing system complexity

One way to derive a high complexity system is to derive a large unit inventory by a single divisive clustering run, starting from the segment boundaries derived by acoustic segmentation. Alternatively, one could run Viterbi training after an intermediate size unit inventory is designed, re-estimate the atomic group sufficient statistics, and continue divisive clustering to increase the inventory using these new statistics. In preliminary experiments, we found that better results were obtained using this second approach. Once one has the intermediate size inventory and new sufficient statistics, the system complexity can be increased in three ways, as described next.

1. High complexity ASWU system

After convergence of the Viterbi training algorithm, new lexical position-dependent atomic group sufficient statistics are computed based on the new segment boundaries. The ML clustering processes can then be started using the unit model inventory derived by Viterbi training as the initial cluster representatives. The unit inventory can be expanded using ML clustering (divisive followed by K-means) and refined using Viterbi training. Iterative application of ML clustering and Viterbi training appears to give better performance, keeping the segment boundaries near optimal.

2. Unconstrained CD-ASWU system

Given the success of explicitly modeling context in a phone-based system, another approach to increase the complexity of an ASWU-based system is by estimating parameters of

context dependent ASWU (CD-ASWU) models. Given the segmentation of the low complexity system, atomic group sufficient statistics can be computed for each unit in each unique left and right context. A high complexity system can then be derived as in the ASWU case by iterative ML clustering and Viterbi training.

3. Center constrained CD-ASWU system

This approach is very similar to the unconstrained CD-ASWU system in that atomic group sufficient statistics are computed for each unit in each unique left and right context. The difference is that only parameter sharing among unit models with the same center unit are allowed. This constraint is equivalent to the approach used in phone-based systems where only parameter sharing among the same state of context dependent models with the same center phone are allowed. The center constrained CD-ASWU training is implemented by running an ML clustering run for each center unit. The unit inventory is increased by use of divisive clustering until the average likelihood per frame exceeds a heuristically set threshold or all cluster occupancies fall below the minimum occupancy threshold. Then a number of K-means iterations are run removing clusters that have fewer observations than the minimum occupancy.

2.4. Temporal resolution refinement

The iterative ML clustering and Viterbi training approach is beneficial as it allows retaining near optimal segment boundaries in the unit inventory design algorithm. A problem it introduces however is that throughout the unit design, the temporal resolution of the system decreases. When neighboring units within a lexical entry are clustered in the same cluster, they are merged to form a single unit causing loss of temporal resolution. Note that segments merged given a low complexity system unit inventory might not have been merged given a high complexity system unit inventory.

One approach to avoid this problem is to set the threshold that controls the temporal resolution in the first unconstrained acoustic segmentation so that a very high temporal resolution is obtained, compensating for the loss in temporal resolution incurred during the unit inventory design algorithm. A possible problem with this approach is that it might result in poor decisions on parameter sharing during the clustering stage. Another approach to circumvent the problem of the loss of temporal resolution is to increase the temporal resolution by splitting the segments derived after a Viterbi training stage. By use of acoustic segmentation, as described in section 2.1, each segment in the Viterbi training segmentation can be split. New atomic group sufficient statistics can then be computed for the new segmentation, and a new lexicon can be defined by running one or more K-means clustering iterations (i.e. partition the new set of atomic group sufficient statistics over the existing unit model inventory). Successive identical units will be merged as before, so the effective increase in pronunciation length is much less than a factor of two.

3. EXPERIMENTS

Experiments were conducted on the DARPA Resource Management (RM) database [9], which is a read speech corpus with a 1000 word vocabulary. Although the proposed algorithm is developed to attack the increased acoustic variability problem in spontaneous speech corpora, initial experiments were conducted on the smaller RM corpus to investigate the viability of the algorithm and explore different options at a lower experimental cost.

The pre-defined 109 speaker training set of approximately 228 min of speech was used as training material for unit inventory and lexicon design. The pre-defined February 1989 test set containing 300 utterances was used as a development test set. The 4 available test sets (feb89, oct89, feb91 and sep92) were used for some of the trained systems, and the average of the 4 results will be referred to as the *full test set* performance. The word-pair grammar provided with the RM database was used in the search.

Feature vectors were computed every 10 ms and included 12 Mel-warped cepstral coefficients and normalized energy and their first and second order differences (39 dimensions in total). The speech signal was windowed using a Hamming window of 25 ms, and a first order pre-emphasis filter (0.97) was applied.

The word recognition performance is derived from a DP word-level string alignment of the recognizer output to the reference transcription. Denoting the number of correct labels as H, the number of insertions as I and the total number of labels in the reference transcription as N, the % correct figure is defined as $H/N \times 100\%$ and the % accuracy figure is defined as $(H - I)/N \times 100\%$.

3.1. Phone systems

For performance comparisons, a phone-based HMM system was trained using the HTK toolkit [10]. The 48 phone set and lexicon provided with the RM database were used. The HMMs had a 3 state left-to-right topology without allowing skips. Starting from context-independent (CI) model parameter estimates derived from the TIMIT database, 4 Baum-Welch (BW) training iterations were performed to derive a 145 state CI system. These models were then cloned for each unique phone context (expanded to a triphone system), and 2 BW training iterations were run to train the 2254 state system. Two parameter sharing techniques were used on the 2254 triphone models. One involved agglomerative clustering, resulting in 1875 distributions, and the other used tree-based clustering, which gave 1557 distributions.

The recognition performance on the February 1989 test set is given in table 1. The performance of the CI phone system on the full test set was 75.6% accuracy. The performance of both the agglomerative-clustered and tree-clustered triphone systems on the full test set was 89.1% accuracy.

3.2. ASWU systems

To derive an ASWU unit model inventory and lexicon, the algorithm described in section 2.1 was applied. The threshold in the unconstrained acoustic segmentation was set so that the resulting segmentation contained approximately 3 acoustic segments per phone. This gives a temporal resolution comparable to the phone-based systems, and preliminary experiments using coarser segmentations showed performance degradation. Variances for the acoustic seg-

System	Number of states	Performance %acc. (%cor.)
Context Independent	145	75.6 (77.9)
Triphones	2254	87.5 (89.6)
Agglomerative clustered triphones	1875	90.2 (90.9)
Tree-based clustered triphones	1557	90.0 (90.5)

Table 1. HMM system results on the February 1989 test set for different system configurations.

mentation were computed on a per utterance basis. During the ML clustering stage, the minimum cluster occupancy was set to 100 frames; clusters with fewer observations were removed.

One set of experiments used progressive refinement to obtain a series of unit inventories of different sizes. First, a 124 unit model inventory and corresponding lexicon was derived in 5 stages of alternating divisive and K-means clustering, followed by 3 iterations of Viterbi training. Then, the size of the unit inventory was increased up to 646 units by 2 iterations of ML clustering (5 and 7 stages of alternating divisive and K-means clustering) and Viterbi training (2 iterations). The temporal resolution was then increased by splitting all segments in the final Viterbi segmentation in two and reclustering, as described in section 2.4. After low frequency clusters were removed, the size of the unit inventory was 635. This unit inventory and the corresponding lexicon were then used in 2 iterations of Viterbi training. Finally the size of the unit inventory was further increased to 1385 by one more ML clustering and Viterbi training step. The recognition performance in terms of %correct and %accuracy for the different system configurations are depicted in figure 1.

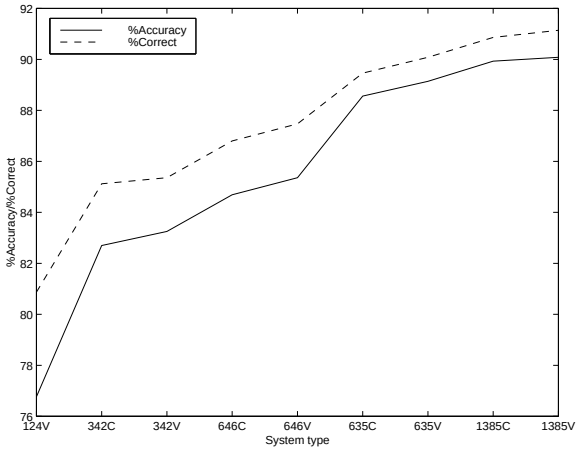

Figure 1. Recognition performance on the February 1989 test set using unit inventories and lexicons derived at different stages of progressive refinement. System type indicates the number of units and is appended with either a C for the system after clustering or a V for the system after Viterbi training.

In another thread of the same experiment, we looked at the effect of changing the point of temporal splitting in the progressive refinement steps. The 646 unit inventory was expanded, without temporal splitting, to a size of 1147 units by an additional ML clustering and Viterbi training step. The temporal resolution was then increased by splitting the segments in two through acoustic segmentation, and 5 iterations of K-means clustering were run. The resulting inventory, which included 1098 units after low frequency clusters were removed, was then refined using 2 passes of Viterbi training. Using the 1147 unit model inventory and lexicon for recognition of the February 1989 test set resulted in a 86.9% accuracy (88.8% correct), while the final 1098 unit inventory and lexicon resulted in a 90.4% accuracy (91.3% correct), confirming that temporal refinement is an important step. In addition, the 1098 unit inventory gave performance comparable to the 1385 unit inventory, suggesting that temporal splitting need not be early in the refinement process.

The performance of the 124 and 1385-unit Viterbi trained ASWU systems on the full test set was 74.1% accuracy and 89.3% accuracy, respectively.

3.3. CD-ASWU systems

Three CD-ASWU experiments were conducted to test the effect of unconstrained vs. constrained clustering and size of the base inventory. The base inventory (either 124 or 635 units) was expanded by considering "tri-units" to be units conditioned on the unit label of their left and right neighbor. Atomic group sufficient statistics were computed for these context-dependent units for use in subsequent clustering. As in previous experiments, clusters with less than 100 observations were eliminated in all ML clustering steps.

First, a 124-unit ASWU system was used as the starting point for an unconstrained CD-ASWU system. After context expansion, 6331 unique (atomic) units in context were found. The unit inventory was increased to 1519 units by 3 iterations of ML clustering and Viterbi training. The recognition performance on the February 1989 test set is depicted in figure 2. In this case, Viterbi training actually hurt performance for the largest inventory. The best system gave 86.3% accuracy.

The unconstrained clustering system with 1519 units can be compared to a system designed using center-constrained clustering, again starting from the 124-unit ASWU system. The resulting 1262 unit inventory was based on divisive clustering, followed by 5 iterations of K-means clustering, followed by 2 iterations of Viterbi training. The recognition performance of the final system on the February 1989 test set was 84.8% accuracy. Extrapolating from the performance of the unconstrained case, there appears to be no advantage to center-constrained clustering and possibly some performance degradation.

Finally, the 635-unit ASWU system was used as the starting point for a unconstrained CD-ASWU system. After context expansion of the 635 unit inventory segmentation, 12943 unique units in context were found. The unit inventory was increased to 1382 units by 2 iterations of ML clustering and Viterbi training. The recognition performance of the final system on the February 1989 test set was 89.3% accuracy, which outperforms the unconstrained case starting from 124 units. The key differences that might explain

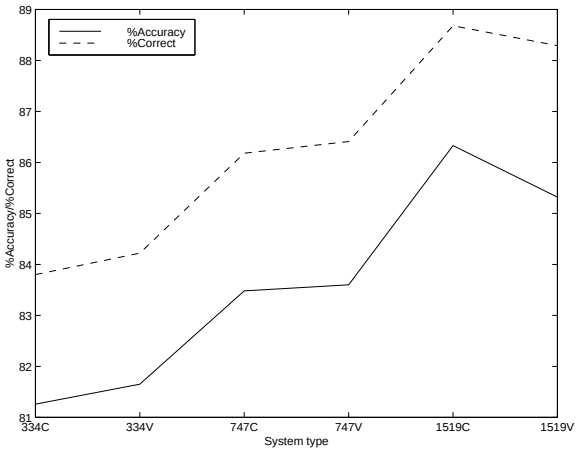

Figure 2. Recognition performance on the February 1989 test set using unit inventories and lexicons derived at different stages of the unconstrained CD-ASWU training algorithm. System type indicates the number of units and is appended with either a C for the system after clustering or a V for the system after Viterbi training.

the improved performance are that the start from 635 units has incorporated temporal refinement and the number of atomic groups is larger when starting from 635 units. The context-dependent 1382-unit system can be compared to the 1385-unit system where the atomic groups are dependent on lexical position rather than unit context. Lexical position dependence gives better performance, 90.1% vs. 89.3% accuracy, but this may be explained by the larger degrees of freedom in clustering by having 30k vs. 13k atomic groups.

4. DISCUSSION

The proposed automatic unit inventory and lexicon design algorithm performs comparable to or better than a phone-based system. At low complexity, the 124-unit ASWU system is comparable to the 128-unit system described in [7] and to the 145-state CI phone-based HMM system. At high complexity, the best ASWU system performs comparable to the best triphone system. Using fewer constraints in the unit inventory and lexicon design algorithm when increasing the complexity of the system results in better performance. For both the unconstrained ASWU as well as the CD-ASWU systems, increased temporal resolution resulted in better system performance, indicating the importance of the temporal resolution in the system design. The particular stage in inventory design for increasing temporal resolution did not appear to be an important factor.

The comparable performance of the low complexity system with the results of [7] demonstrates that the constrained unit design approach is competitive with previous ASWU work. The performance at high complexity, comparable to a triphone system, shows the robustness of the algorithm for large unit inventories. Given that phone-based systems represent the state-of-the-art in read speech tasks, especially at high system complexity, the comparable performance of the ASWU-based system shows the viability of the proposed algorithm. In spontaneous speech recognition tasks, the ASWU-based system is expected to perform better than a phone-based system using standard baseforms, since speakers use canonical pronunciations much less frequently in casual speech. It remains to be seen, however, whether the use of ASWUs will outperform phone-based systems with complex pronunciation models.

Acknowledgments

This work was supported by ATR Interpreting Telecommunications Laboratory.

REFERENCES

[1] C.H.Lee, B.-H Juang, F. K. Soong and L. Rabiner, "Word recognition using whole word and subword models," *Proc. Int. Conf. on Acoust., Speech and Signal Proc.*, vol. 1, pp. 683-686, 1989.

[2] T. Svendsen, K.K. Paliwal, E. Harborg and P.O. Husøy, "An improved subword based speech recognizer," *Proc. Int. Conf. on Acoust., Speech and Signal Proc.*, vol. 1, pp. 108-111, 1989.

[3] L. R. Bahl, P.F. Brown, P. V. de Souza, R. L. Mercer and M. A. Picheny, "A Method for the Construction of Acoustic Markov Models for Words," *IEEE Trans. Speech and Audio Processing*, vol. 1, no. 4, pp. 443-452, 1993.

[4] M. Bacchiani, M. Ostendorf, Y. Sagisaka and K. Paliwal, "Design of a Speech Recognition System based on Non-Uniform Segmental Units," *Proc. Int. Conf. on Acoust., Speech and Signal Proc.*, vol. 1, pp. 443-446, 1996.

[5] K.K. Paliwal, "Lexicon building methods for an acoustic sub-word based speech recognizer," *Proc. Int. Conf. on Acoust., Speech and Signal Proc.*, vol. 2, pp. 729-732, 1990.

[6] T. Svendsen, F. K. Soong and H. Purnhagen, "Optimizing baseforms for HMM-based speech recognition," *Proc. European Conf. on Speech Commun. and Technology*, vol. 1, pp. 783-786, 1995.

[7] T. Holter and T. Svendsen, "Combined Optimisation of Baseforms and Model Parameters in Speech Recognition Based on Acoustic Subword Units," *Proc. IEEE Workshop on Automatic Speech Recognition*, pp. 199-206, 1997.

[8] A. Kannan and M. Ostendorf, "A comparison of constrained trajectory models for large vocabulary speech recognition," *IEEE Trans. Speech and Audio Processing,* to appear May 1998.

[9] P. Price, W. M. Fisher, J. Bernstein, D.S.Pallett, "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition," *Proc. Int. Conf. on Acoust., Speech, Signal Proc.*, vol. 1, pp. 651-654, 1988.

[10] HTK, version 2.1, Cambridge University, 1997.