Mapping context dependent acoustic information into context independent form by LVQ

Jyri Mäntysalo, Kari Torkkola†, and Teuvo Kohonen

Helsinki University of Technology
Laboratory of Information and Computer Science
Rakentajanaukio 2 C
SF-02150 Espoo
FINLAND

†IDIAP (Institut Dalle Molle d'Intelligence Artificielle Perceptive)
C.P. 609
CH-1920 Martigny
SWITZERLAND

Address for correspondence:
Kari Torkkola
IDIAP, C.P. 609
CH-1920 Martigny
SWITZERLAND

email: karit@idiap.ch
fax: +41-26-227818
phone: +41-26-227664

List of unusual symbols: none

List of abbreviations:

HMM	Hidden Markov model
VQ	Vector quantization
LVQ	Learning vector quantization
OLVQ	Optimized rate learning vector quantization

Number of pages:	26
Number of figures:	4
Number of tables:	3

Proposed running head: Modeling context by LVQ

Keywords:

- Context modeling
- Learning Vector Quantization
- Hidden Markov models
- LVQ in phonemic speech recognition
- Neural networks in speech recognition

Abstract

In the framework of phonemic speech recognition using Hidden Markov Models (HMMs) together with codebooks trained by Learning Vector Quantization (LVQ), a novel way to model context-dependencies in speech is presented. We use LVQ to map acoustic contextual data into context-independent phonemic form. The acoustic data is in the form of concatenated averages of successive short-time feature vectors. This mapping eliminates the need to employ context dependent phonemic, for example, triphone HMMs, and the difficulties associated therein. Instead, simpler context-independent discrete observation HMMs suffice. We report excellent results for a speaker dependent task for Finnish.

Zusammenfassung

Wir diskutieren ein neues Modell von Kontextabhängigkeiten in der phonemischen Spracherkennung. Unser Modell basiert auf den Methoden der Hidden-Markov-Modelle (HMM) und der lernende Vektorquantisierung (LVQ). Wir benutzen LVQ, um Informationen über den akustischen Kontext in eine kontextunabhängige phonemische Form abzubilden. Die Kontextinformation besteht aus der Aneinanderkettung von Durchschnitten von aufeinanderfolgenden Kurzzeitmerkmalsvektoren. Durch diese Abbildung ist es dann nicht mehr notwendig, kontextabhängige phonemische HMM (wie, z. B., triphonale HMM) zu verwenden, und wir können die Probleme vermeiden, die mit solchen Methoden verbunden sind. Stattdessen können einfachere, kontextunabhängige HMM benutzt werden. Wir haben mit der Methode gute Ergebnisse für ein sprecherabhängiges Spracherkennungsproblem erhalten.

Résumé

Nous présentons, dans le cadre de la reconnaissance de la parole basée sur les phonèmes utilisant des modèles de Markov cachés (HMM, Hidden Markov Models) en conjonction avec des dictionnaires entraînés par quantification vectorielle supervisée (LVQ, Learning Vector Quantization), une nouvelle manière de modéliser les dépendances contextuelles. Nous utilisons les LVQ pour transformer les données acoustiques contextuelles en une forme phonémique indépendante du contexte. Nous faisons la moyenne de plusieurs vecteurs caractéristiques successifs, vecteurs résultant de l'analyse à court-terme, et

ensuite nous concaténons les moyennes ainsi obtenues pour modéliser les données contextuelles. Cette transformation élimine le besoin d'utiliser des modèles de Markov dépendants du contexte, ainsi que les difficultés associées. Au contraire, des modèles simplifiés indépendants du contexte des observations discrètes sont suffisants. Nous avons obtenu par cette méthode d'excellents résultats pour une tâche dépendante du locuteur en Finnois.

1 Introduction

In certain languages, such as Finnish, recognition of phonemes from the acoustic signal is a good basis for further processing, mainly due to small number of phonemes, and due to a simple correspondence between orthography and phonemic transcriptions. Other languages similar to Finnish in this sense are Spanish, Italian, and romanized Japanese. A further attraction in producing a stream or a lattice of phonemes lies in the enormously reduced bandwidth for later processing stages in speech recognition. However, being abstract linguistic units, phonemes do not directly correspond to clearly distinct acoustic segments. Phonemes have varying lengths, they may consist of several partly overlapping acoustic segments, and their acoustic appearances vary according to their context, and also across speakers. Even the same speaker is not able to produce exactly the same acoustic signal for the same utterance. The task of transcribing speech into phonemes is thus far from trivial. In this paper we address this task for the Finnish language.

To tackle the variability induced by context, many succesful speech recognition systems reported in the literature employ context-dependent subunit models in the Hidden Markov Model (HMM) formalism, phoneme models among others [1, 16]. Due to large variations in acoustic realizations of phonemes caused by context, usually a separate model is constructed for a sub-unit in each context that causes its realization to be different. Construction of such models is a tedious task and requires considerable linguistic knowledge and/or sophisticated methods. Also, because of the large number of emerging sub-unit models, sparse training of the models may be a problem with a limited amount of training data. Tying of parameters may, however, alleviate some of these problems [24].

In this paper, we present experiments whose purpose is to demonstrate that context effects can be handled at a level lower than HMMs. To this end, we use the Learning Vector Quantization (LVQ) methods, including the latest developments [10, 11, 12, 14]. The idea is to employ LVQ to map acoustic data, *including context in time*, into context-independent phonemic form, and to interpret the symbolic phonemic data further by using *context-independent* HMMs. Our basic configuration is thus LVQ combined with discrete observation HMMs. As many other research groups have used the same configuration, the novelty of our work can be better appreciated after a review of related research. This will be given in Section 2. The outline of the rest of this paper is as follows. Section 3 presents experiments in finding the best-performing representation of context data for the mapping. In Section 4 we introduce a new way to use LVQ to produce additional information, useful to phoneme-level segmentation. Section 5 demonstrates how discrete observation multiple stream HMMs can be applied to information produced by LVQ to accomplish our final task, transcribing speech into phoneme sequences. Finally, in Section 6 we draw some conclusions and compare our work to that of others.

2 Previous work on LVQ/HMM combinations

The role of conventional vector quantization (VQ) algorithms in speech recognition, such as the Linde-Buzo-Gray algorithm, is to *represent* speech feature vectors with the smallest possible distortion. This is not the case with the LVQ-methods, which try to aim at *discrimination* of pattern classes, whatever they may be. Codebook vectors directly define the class borders in the feature space according to nearest-neighbor rule. LVQ modifies the codebook vectors adaptively so that the borders between classes, that are midpoints between codebook vectors of different classes, will approximate Bayes' decision surfaces. Quantization error as the representation error is of secondary interest. However, in this work we try to make use of the quantization error but rather as a distance measure to a pattern class. For a more detailed description of LVQ, the reader is referred to [8, 13, 12].

In speech recognition literature, the most common way of using LVQ is as a substitute to conventional vector quantization. Short time feature vectors have been used as the basis for classification, usually to phoneme classes. Training has been done either by single frames [7, 23], or by concatenating several frames together to represent some context [5, 6, 23, 25]. Further, the whole (phoneme) token, all of the frames or successive shifts of the concatenated frames have been used as training examples.

To combine local phoneme classification decisions heuristic duration-based rules have been used [15], or, in phoneme token recognition, LVQ outputs have been integrated over the duration of the whole phoneme token [18]. A better solution is statistical; the convenient and well studied framework of HMMs can be exploited. The local classifications made by LVQ can be treated just as VQ-codebook indices using conventional *discrete* observation HMMs (dHMM) [5, 6, 23, 25, 7].

When the task has been speaker dependent and phoneme oriented, significant improvements have been observed due to LVQ when compared against ordinary VQ [6, 23]. In speaker independent vowel token recognition no improvement was attained by LVQ over conventional vector quantization [7]. On the other hand, in this work the basis for phoneme classification was a single-frame feature vector, which obviously was not a suitable representation for the task. Where the task has been word or phrase recognition, rhe results have been slightly controversial: no improvement in [25], but significant improvements in [5].

In all of the above mentioned work, LVQ-codebooks were trained to distinguish phoneme classes regardless of whether the final task was word recognition or phrase recognition, with the hope that phoneme distinctions are important to the final task. To remedy this kind of a mismatch, McDermott et al pointed out that LVQ is a special case of more general *probabilistic gradient descent* [19, 20]. They showed how to define a loss function that better reflects, for example, the word error rate, which is more important to the actual task, and how to minimize this loss function (and the word error rate) by using gradient descent.

Since LVQ is only applicable to *static* patterns (patterns that have a fixed dimensionality), Ramesh et al developed a way to normalize words of different length [22]. They show how using HMMs having the same number of states, words of different length can be normalized and presented as a concatenation of the centroid vectors of the states. Because all the words are now represented as fixed length vectors, LVQ can now be applied to train and to recognize these vectors

Since our work is targeted towards the task of "phonetic typewriter", where phoneme errors *do* count, it is a natural choice to employ LVQ for phoneme classification. The key point in the current work is to explore different kind of representations of context as the basis for LVQ-classification. The aims are twofold. 1) To maximize phoneme recognition accuracy by taking the context into account. 2) To derive other useful information for the task, especially related to phoneme-level segmentation. We shall elaborate a little on the second aim. In addition to using classification results provided by LVQ, we try to exploit vector quantization errors as an aid to the decoding process. In contrast to all of the related work cited above, we are training the codebook vectors by samples constructed from phoneme centers only. We do not use any other portions of phonemes at all. By this we aim at being able to produce high quantization errors at transitions between phonemes. Subsequently, the local minima and maxima of the quantization error contain valuable information for phonemic segmentation. In addition, we have applied LVQ to a segmentation task, whereby the context vectors are classified into two classes only, representing phoneme center locations and transitions. The output labels from both of the two LVQ-codebooks can be interpreted as observations of discrete-distribution HMMs with multiple observation streams [4]. The models also include quantization errors as a separate observation stream.

3 Finding the best form of context data

To construct a mapping from acoustic data into phonemic form, we first made experiments to find the best performing *representation* of context data. By the best performance we mean here the highest accuracy as a phonemic classifier.

As it is well known that context has a notable effect on phonemes, mapping based on short-time feature vectors may be accurate enough only for some simple cases. Often the context and/or the time evolution of the feature vectors needs to be taken into account. This is usually done at the level of HMMs. Here we are trying to account for it at an earlier stage. A crude but straightforward way to do this is to concatenate feature vectors into longer ones, as done in much of the previous work. In this work, the pattern vectors are formed by both averaging and concatenating short-time feature vectors within a time domain window. Since there exist no known theoretical way of deciding the combination that will give the best performance with the given data, we have tried to find a good representation experimentally.

3.1 Speech data and preprocessing

The same Finnish speech database was used here as in [23]. The database contains four repetitions of a set of 311 words spoken by three male Finnish speakers. Each set consists of 1737 phonemes. In the original Finnish language, there are only 21 different phoneme classes: 8 vowels and 13 consonants. Four additional phonemes have been adopted with loan words from other languages, but none of those were represented in the database. In phoneme token recognition experiments unvoiced plosives /k,p,t/ are considered to consist of two parts, representing the closure part and the plosive release. The actual number of pre-segmented tokens in each set is thus 1903. It must be also noted that we consider each diphthong to be a concatenation of two vowels, rather than to be a phoneme of its own. This is in contrast to the common practice in phonemic speech recognition with, the English language, for example. This can also be interpreted as working with LVQ at a level slightly below phonemes.

Three of the repetitions were used each time for training, and the remaining one for testing. Four independent runs were made by leaving one set at a time for testing. Thus all recognition results presented in this paper are speaker-dependent and averages of 12 runs. As features computed every 10 milliseconds we use 20-component mel-scale cepstra computed in the way described in [3]. The cepstral vectors are thereafter liftered with raised sine [23].

3.2 LVQ training procedure

We have first run an extensive amount of preliminary experiments to search for good LVQ parameters for this specific task. Public domain software package *Lvq_Pak* [14] was used throughout these experiments. The following learning scheme proved to give the best results on the average.

Codebook vectors are initialized with samples from training data, so that each class is assigned an identical number of codebook vectors. Candidate samples are tested against all other samples; if classification with K-nearest-neighbor –method is correct, the candidate will be accepted as an initial codebook vector. This guarantees, that all the initial vectors reside inside the correct class decision borders. Codebook vectors are trained iteratively by the OLVQ1-algorithm using 10000 iterations and the initial learning rate value $\alpha(0) = 0.3$ (an iteration denotes here a presentation of a single example vector). We employed the LVQ1 for the final training using $\alpha(0) = 0.2$. The learning rate decreased linearly to zero during the training process. In the latter stage, the number of iterations was 100 times the number of codebook vectors.

3.3 Concatenation and averaging experiments

This subsection presents detailed results of our studies to find a good context representation. We have restricted ourselves to two operations only on the successive feature vectors: averaging and concatenation, and combinations of them. One could imagine overlapping and weighting schemes, but to keep the search space reasonable we adopted these restrictions. All experiments in this section are done with pre-segmented phoneme tokens. In Section 5 we discuss how these results can be extended into continuous speech.

First, we experimented with concatenation of successive short-time feature vectors into one high-dimensional context vector. Fig. 1 plots the results with four different codebook sizes (these are indicated at the right-hand side of the plots). Y-axis displays the recognition rate, and x-axis the number of successive feature vectors that have been concatenated into a long “context vector”.

Figure 1 about here.

Two things are apparent in the plot. Firstly, conventionally used codebook sizes are too small for high dimensional context vectors. In this case the codebook size ought to be around 1000. Secondly, when a small codebook (200 codevectors) is used, there is a limit after which adding extra context begins to impair the recognition accuracy. With larger codebooks (500, 1000, or 2000 codevectors) such a limit is not yet visible. Both of these observations are due to the following. By including a wide amount of context in the vectors, we obtain numerous different patterns that represent the same phoneme class. Subsequently, to represent the wealth of all these contextual variations, a large codebook is needed.

To explore the limit of useful context width, and to simultaneously keep the context vector dimension low, we computed averages of two successive feature vectors, and concatenated such averages. Figure 2 plots these results with the same codebook sizes.

Figure 2 about here.

X-axis displays now the number of averages that have been concatenated. The time scale of Fig. 2 extends two times as far as in Fig. 1. This plot reveals now some asymptotic behaviour that was not yet visible in the previous one. The limit of useful amount of context can be observed clearly when codebooks of 200, 500, and 1000 codevectors are used, but this limit is still not reached using 2000 codevectors. However, since the curve is almost flat we decided not to extend the context any further. The difference between using 1000 and 2000 codevectors is not large, but it seems to be increasing as the the amount of context increases.

It is obvious, that the farther we move from the phoneme centers the more irrelevant information the speech signal contains to classify this particular phoneme. Also, since the durations of phonemes vary, representations as above, that must be sharply time-aligned, do not probably work in the best possibly way. It seems thus reasonable to average wider slices of context farther away from the centers. Table 1 lists the results of these experiments, where averages of four have been used outside the phoneme center. For comparison, some results of Figs. 1 and 2 are duplicated, together with results of concatenating averages of three vectors.

Table 1 about here.

The confidence intervals (with 99% confidence) of the figures are fairly tight due to a large number of tests ($\approx$23000): $\pm$0.16% for the best figure in the table, and $\pm$0.56% for the worst.

Concentrating on those representations that cover a wide context (the bottom four rows), we can observe that among the representations that cover about the same amount of context, the performance is almost equal. The ones using wider averages outside centers are slightly better with smaller codebooks, and have also the advantage of smaller dimensionality. The context vector form corresponding to the best results (the last row of Table 1) was chosen as the basis for further experiments. Compared to a single short-time cepstral vector, we can see the best context data representation improves phoneme token classification accuracy as much as from 87% to 99%. This a 92% reduction of the error rate!

4 What information to aid segmentation can be derived using LVQ ?

In the previous section we saw that the mapping of contextual data into phonemic form works well for phoneme center locations. However, there is no guarantee that it is usable with continuous speech, with other parts of phonemes but exact centers. This can only be verified with experiments on the actual task, producing phonemic transcriptions of spoken utterances. These will be described in Section 5. Before going into the experiments, we would like to discuss what other information useful to the task might be derivable using LVQ.

It is possible to gain additional information for phonemic segmentation by the merit of training using phoneme centers only. Because the chosen context vector covers a relatively long period of time domain, we can presume that transitions, or sections close to transitions, will appear substantially different from center parts of phonemes. This will in turn result in high LVQ quantization errors at phoneme transitions, since the codebook vectors represent phoneme centers.

There is also another conceivable solution to make sure that LVQ produces reasonable classification results even with continuous speech, not only with center points. As exemplars of each phoneme class, successive shifts of the whole phoneme duration could be used. However, this might lead into another problem: portions close to transitions might be extremely confusable and might resemble very much other parts of other phonemes. After preliminary experiments we abandoned this strategy and continued with the formerly described one.

Our solution is thus to slide the fixed time domain window, that covers a period of 220 milliseconds, (Fig. 3) over all of the speech signal and to compute phoneme classification by LVQ for its center position. This results in a so-called quasiphoneme sequence with a phoneme symbol and the corresponding quantization error every 10 ms. The main source of information for decoding by HMMs is the quasiphoneme sequence, but the error can also be used as an aid to phonemic segmentation.

Figure 3 about here.

To obtain even more explicit information about phoneme locations, we have experimented with training another codebook for a two-class problem: to discriminate phoneme centers from transitions between phonemes. For this codebook, the context vector form and the number of codevectors were kept the same as in the most succesful phoneme recognition experiment, i.e., the context vector of Fig. 3, and 2000 codevectors. The codebook is trained using exactly the same procedure, but with examples from phoneme centers and phoneme borders. Table 2 displays these results with different context vector forms and codebook sizes.

Table 2 about here.

This segmentation task indeed stresses the advantage of employing wide context. As expected, with only short-time cepstral vectors, the accuracy was as low as 75.4%. The best accuracy we obtained in this phoneme/border classification problem was 94.6% using context vectors. The confidence intervals (with 99% confidence) of the figures are the following ($\approx$46000 tests): $\pm$0.26% for the best figure in the table, and $\pm$0.50% for the worst. Using the largest codebook, we cannot thus say that the representation corresponding to the last row outperforms the shorter one on the previous row. However, we chose the wider representation for this task as well, due to the convenience of using the same representation.

Figure 4 about here.

Figure 4 illustrates all this information available for phonemic decoding. We can see that (at least in this example) the label sequence produced by the phoneme classifier LVQ-codebook is almost perfect, and that the corresponding quantization error indicates quite clearly phoneme class centers and borders. Segmentation classifications (at the bottom of the figure) also match well phoneme centers and transitions. However, the quantization error appears like a time derivative of the quantization error of the phoneme classification codebook, and does not seem to be as useful as the label sequence itself.

The remaining problem, how to combine all this information to produce phonemic transcriptions of uttered speech, will be addressed in the next section.

5 Decoding by HMMs

The classification results produced by LVQ can be conveniently interpreted as output symbols of discrete observation HMMs. In our approach, each phoneme is modeled as a separate context independent Markov source. A simple four-state left-to-right topology has been used. Output symbol distributions have been attached to state transitions within the phoneme models and they are trained using the standard maximum likelihood approach [21].

To allow arbitrary phoneme sequences, the decoding algorithm is allowed to loop back from the final state of a phoneme model to the initial state of any other model. Probabilities for the transitions between phonemes (phoneme bigram probabilities) were estimated separately from a large text corpus.

The main differences between a conventional discrete observation (vector quantization) HMM–approach and the approach we are using here are the following:

1. LVQ-codebook is used to map contextual data into context independent form, or equivalently, LVQ is used to compute frame-synchronous phoneme *classification*, not to *represent* the speech signal with minimum distortion.
2. LVQ-codebook quantization errors are included in the HMMs and modeled as a separate input stream, either discrete or continuous-valued. To guarantee the usefulness of this information, we train LVQ only by context windows centered at phonemes.
3. An additional LVQ-codebook is used for segmentation, and its label stream is modeled as yet another input stream of the HMMs.

Concentrating on classification, not on representation, at the item 1 is the basic LVQ-strategy that has also been used previously together with HMMs in [6, 5, 25, 7, 23]. To this basic strategy we have added a carefully chosen representation of context. Items 2 and 3, are novel to our knowledge. Thus, we are using three separate parallel streams with the HMMs. For the quantization errors, both 8-level quantized values and continuous-density probability distributions with large constant variance were tried.

It may be useful to recapitulate that in the Finnish language that we are using, phonemic transcriptions almost correspond to text as such. In fact, although we are talking about phonemic transcription accuracies in these experiments, the figures refer as well to the accuracy of produced text, measured in letters. Table 3 presents the results of phonemic transcription experiments divided in four main groups:

1. Using short-time cepstral vectors as the basis of phoneme classification by LVQ. The codebook size was 500.
2. Using a large LVQ-codebook of 2000 codevectors to map high-dimensional context vectors into phoneme classes. Quantization error of this codebook is modeled as a separate stream, either as a discrete or as a continuous density distribution.
3. In addition to the above, using a large segmentation LVQ-codebook also with high-dimensional context vectors to produce a third parallel stream of information that consists of only two symbols, one for phoneme centers, another for transitions.
4. Combine the two-stream information of the two above mentioned large codebooks into a single stream in the following way. For each phoneme, two alternative symbols will be used, one symbol for the center part, which is outputted whenever the segmentation codebook produces the center symbol, and another symbol for the transitional part, outputted whenever the segmentation codebook produces the transition symbol. This doubles the amount of possible symbols for the phoneme classification stream also doubling the number of parameters to be estimated. Of course, at the same time the segmentation stream is discarded. As it consists only of two possible symbols, the net effect is to increase the number of the parameters.

In principle, the information contained in separate observation streams ought to be statistically independent to allow observation probability combination by multiplication. Strictly speaking this is not the case in our experiments. In spite of this, improvements were observed.

Table 3 about here.

The 99% confidence limits in Table 3 are $\pm 0.3\%$ for the lowest error figure, and $\pm 0.5\%$ for the worst (≈ 21000 tests).

To comment the table, all of the new combinations are significantly better than the baseline, LVQ-codebook trained for phoneme classification using short-time feature vectors (STF). Replacing short-time feature vectors by context vectors (FON) appears to halve the error rate. Adding now quantization error to the HMMs as a continuous-observation stream, reduces again significantly the error rate (cQE), but the same information in discrete form does not (dQE). The segmentation codebook further adds useful information (SEG), although the information can be seen to be somewhat redundant with the quantization error of the phoneme classification codebook, especially when the latter is presented as a discrete observation stream. All in all, the best structure, combining labels from the two streams into a single stream, modelling quantization errors as a continuous observation stream, reduces the error rate from 9.5% to 2.8%, a reduction of about 70%.

It should be mentioned that manually segmented starting and ending points of the utterances are supposed to be known here. In these experiments, we did not include yet another codebook for the energy of the speech signal, which would have facilitated beginning and endpoint detection.

6 Discussion

In this research we have applied the Learning Vector Quantization methods, including the latest developments [10, 11, 12, 14], to the task of Finnish speaker-dependent speech recognition. Our aim was to produce phonemic transcriptions of spoken utterances. LVQ was used to map contextual acoustic information into context-independent phonemic form, further decoded by context-independent discrete observation hidden Markov models. We experimented with various ways to form a high-dimensional context vector both by averaging and concatenating short-time feature vectors within a time domain window. This method of taking local phonemic context into account was shown to give excellent results when applied to classification of Finnish phoneme tokens. Compared to using short-time feature vectors, classification accuracies increased from 87% up to 99% with the best form of context vector.

We observed that the sizes of LVQ-codebooks must be increased in accordance with the dimensionality and the amount of context included in the vectors. It seems to be obvious that in this way we are able to capture some context-dependent phoneme variations in a large codebook automatically without the need to impose any further linguistic knowledge about phonemes in different contexts. Although a considerable amount of context is used, we do not have to specify which contexts have similar (or different) effects on the appearance of each phoneme. On the contrary, the contextual data is simply all fed to LVQ !

Compared to previous work on LVQ and HMM combinations, our work differs in several respects. Instead of just concatenating an arbitrary number of frames together, we made a careful study to find out the best input representation as a concatenation of averages of successive frames. Instead of using all possible successive shifts over a phoneme token to train the LVQ-codebooks, we position our context window only at the center of each phoneme. By training with centers only, we aimed at producing high quantization errors at phoneme borders. The error then carries useful information for phoneme-level segmentation, and can be modeled as a separate HMM input stream. In addition, we used LVQ to produce two-class segmentation information by training another LVQ-codebook by examples of phoneme centers and transitions. This label stream can be included as yet another input stream to HMMs.

In a way, our approach resembles segmental approaches [2, 17] in which the segment (for example, phone or phoneme) classification is based on the whole duration of the segment. The speech signal is considered as a concatenation of such segments, and this concatenation is found out by Viterbi-decoding. Segment boundaries may then be suggested to Viterbi-decoding by other means that are easy to compute, for example, by separate HMMs. In contrast to classifying segments only between hypothesized segment begin and end points as in [2], we perform segment classification (mapping of context vectors into phonemes) at each discrete time step. This makes our decoding process simpler, since segment borders need not be hypothesized for the classifier. However, the computational burden is now moved to the classifier. Real-time use of the largest codebook that we experimented with requires $D \times N \times R$ multiply/accumulate operations, where D = dimension of the context vector (140), N = number of codebook vectors (2000), and R = frame rate (100). This means 56 MFLOPS, which is within reach of one or two modern signal processor chips.

It is conceivable, that in the speaker independent case the codebook sizes should be even larger to be able to represent the variations without getting “saturated”. How much larger, and how much improvement could be gained, can only be determined by experiments. The previous work either has used single short-time vectors, as the basis of classification [7], or used codebooks of a relatively small size [7, 25], or did not use as much context as we did [5].

The whole approach of transcribing speech into phoneme sequences first may not be a convenient one for languages with a large number of different phonemes, and with irregular and complex correspondence between the orthography and phonemes. Finnish and romanized Japanese are well suitable for this approach [9]. Good candidates could also be Italian or Spanish.

In subunit based speech recognition, an occasional concern is whether improvements in subunit recognition carry out to word or phrase recognition performance. The previous work using LVQ/HMM-hybrids in word and phrase recognition is slightly controversial [5, 25]. It is admittedly the case, that the error criterion used in training the parameters of the recognizer should reflect the final task as closely as possible. Approaches like the one described in [20] might then be close to the optimum. Since our work is directed towards a phoneme-based dictation machine, phoneme errors certainly are important, and it is a sensible approach to try to minimize them.

In our final task, transcribing spoken utterances into phoneme sequences (or phoneme spotting, as it may also be called), the lowest error percentage was approximately 3%. This figure presents a significant improvement when compared to our baseline, the 9.5% error rate obtained in real-time operation using short-time feature vectors as the basis for LVQ classification.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments and suggestions to make this paper readable.

References

[1] Paul Bamberg and Laurence Gillick. Phoneme-in-context modeling for DRAGON's continuous speech recognizer. In *Proceedings of the DARPA Speech and Natural Language workshop*, pages 163–169, Hidden valley, Pennsylvania, USA, 1990.

[2] Y.M. Cheng, D. O'Shaughnessy, V. Gupta, P. Kenny, M. Lenning, P. Mermelstein, and S. Parthasarathy. Hybrid segmental-LVQ/HMM for large vocabulary speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP92)*, volume I, pages 593–596, San Francisco, CA, USA, March 23-36 1992.

[3] Stephen B. Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, August 1980.

[4] V. N. Gupta, M. Lennig, and P. Mermelstein. Integration of acoustic information in a large vocabulary word recognizer. In *Proceedings of the IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP87), volume 2, pages 697–700, Dallas, Texas, April 1987.

[5] H. Iwamida, S. Katagiri, and E. McDermott. Speaker independent large vocabulary word recognition using an LVQ/HMM hybrid algorithm. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP91)*, volume 1, pages 553–556, Toronto, Canada, May14-17 1991.

[6] H. Iwamida, S. Katagiri, E. McDermott, and Y. Tohkura. A hybrid speech recognition system using HMMs with an LVQ-trained codebook. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP90)*, volume 1, pages 489–492, Albuquerque, NM, USA, April 3-6 1990.

[7] Donald G. Kimber, Marcia A. Bush, and Gary N. Tajchman. Speaker-independent vowel classification using hidden Markov models and LVQ2. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP90)*, volume 1, pages 497–500, Albuquerque, NM, USA, April 3-6 1990.

[8] Teuvo Kohonen. Learning vector quantization for pattern recognition. Technical Report TKK-F-A601, Helsinki University of Technology, 1986.

[9] Teuvo Kohonen. The 'Neural' Phonetic Typewriter. *IEEE Computer*, 21(3):11–22, March 1988.

[10] Teuvo Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin-Heidelberg-New York-Tokio, 3rd edition, 1989.

[11] Teuvo Kohonen. Improved versions of learning vector quantization. In *Proceedings of the International Joint Conference on Neural Networks*, volume I, pages 545–550, San Diego, June 1990.

[12] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[13] Teuvo Kohonen, György Barna, and Ronald Chrisley. Statistical pattern recognition with neural networks: Benchmarking studies. In *Proceedings of the IEEE International Conference on Neural Networks*, pages I 61–68, San Diego, July 1988.

[14] Teuvo Kohonen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. LVQ_PAK: A program package for the correct application of Learning Vector Quantization algorithms. In *Proceedings of the International Joint Conference on Neural Networks*, volume I, pages 725–730, Baltimore, June 1992. IEEE.

[15] Teuvo Kohonen, Kari Torkkola, Makoto Shozakai, Jari Kangas, and Olli Ventä. Phonetic Typewriter for Finnish and Japanese. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP88)*, pages 607–610, New York City, USA, April 1988.

[16] Kai-Fu Lee. Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(4):599–609, April 1990.

[17] H.C. Leung, I.L. Hetherington, and V.W. Zue. Speech recognition using stochastic segmental neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP92)*, volume I, pages 613–616, San Francisco, CA, USA, March 23-36 1992.

[18] Erik McDermott and Shigeru Katagiri. Shift-invariant, multi-category phoneme recognition using Kohonen's LVQ2. In *Proceedings of 1989 International Conference on Acoustics, Speech and Signal Processing (ICASSP89)*, volume 1, pages 81–84, Glasgow, Scotland, 23-26 May 1989. IEEE.

[19] Erik McDermott and Shigeru Katagiri. Prototype-based discriminative training for various speech units. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP92)*, volume I, pages 417–420, San Francisco, CA, USA, March 23-36 1992.

[20] Erik McDermott and Shigeru Katagiri. Prototype-based MCE/GPD training for word spotting and connected word recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP93)*, volume II, pages 291–294, Minneapolis, MN, USA, April 27-30 1993.

[21] Lawrence R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[22] Padma Ramesh, Shigeru Katagiri, and Chin-Hui Lee. A new connected word recognition algorithm based on HMM/LVQ segmentation and LVQ classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP91)*, volume 1, pages 113–115, Toronto, Canada, May14-17 1991.

[23] Kari Torkkola, Jari Kangas, Pekka Utela, Sami Kaski, Mikko Kokkonen, Mikko Kurimo, and Teuvo Kohonen. Status report of the Finnish phonetic typewriter project. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN-91)*, pages 771–776, Espoo, Finland, June 24-28 1991. North-Holland.

[24] S.J. Young. The general use of tying in phoneme-based HMM speech recognisers. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP92)*, volume I, pages 569–572, San Francisco, CA, USA, March 23-36 1992.

[25] George Yu, William Russel, Richard Schwartz, and John Makhoul. Discriminant analysis and supervised vector quantization for continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP90)*, volume 1, pages 685–688, Albuquerque, NM, USA, April 3-6 1990.

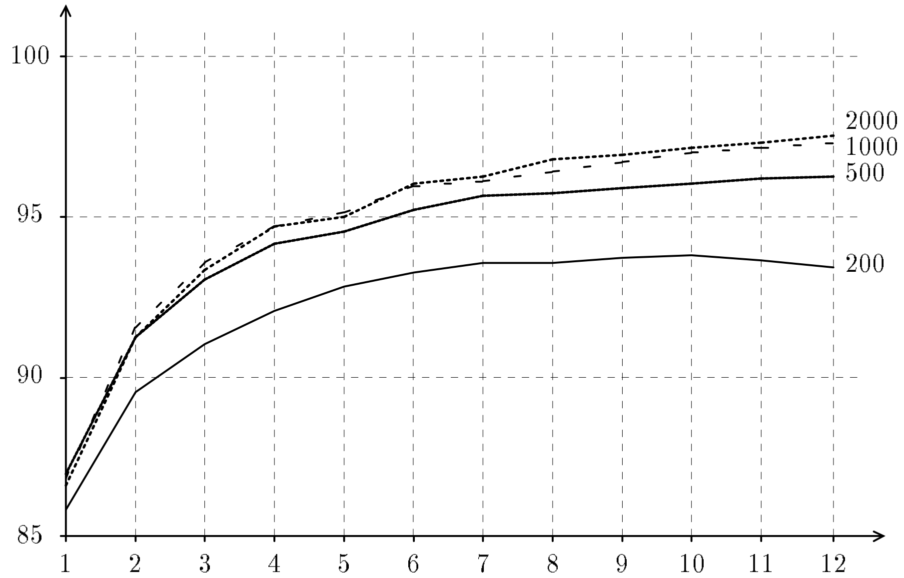

Figure 1:

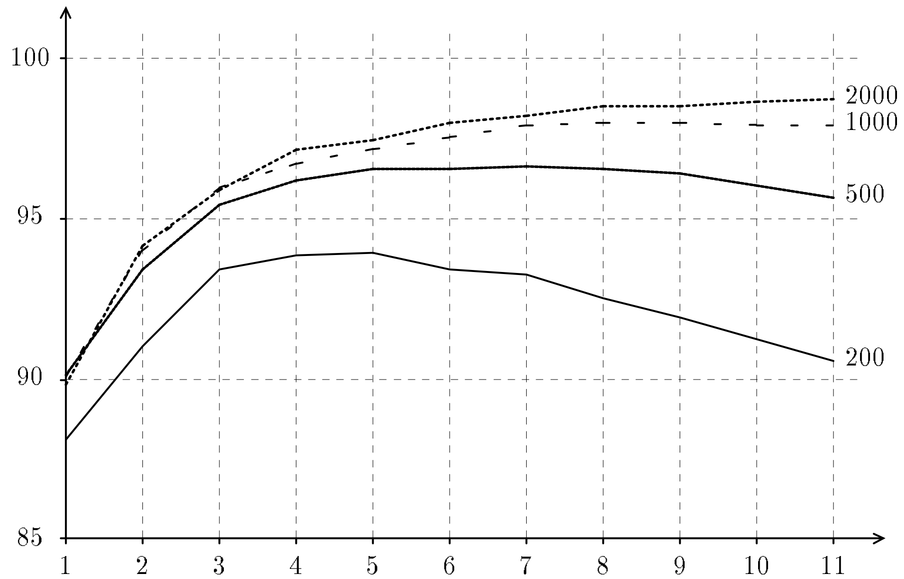

Figure 2:

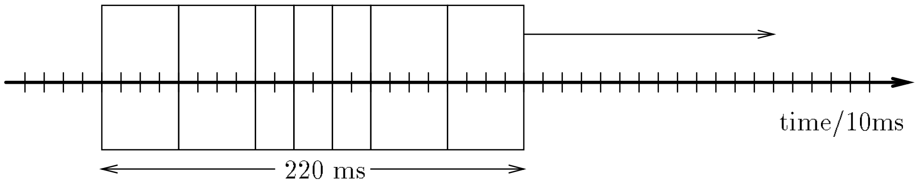

Figure 3:

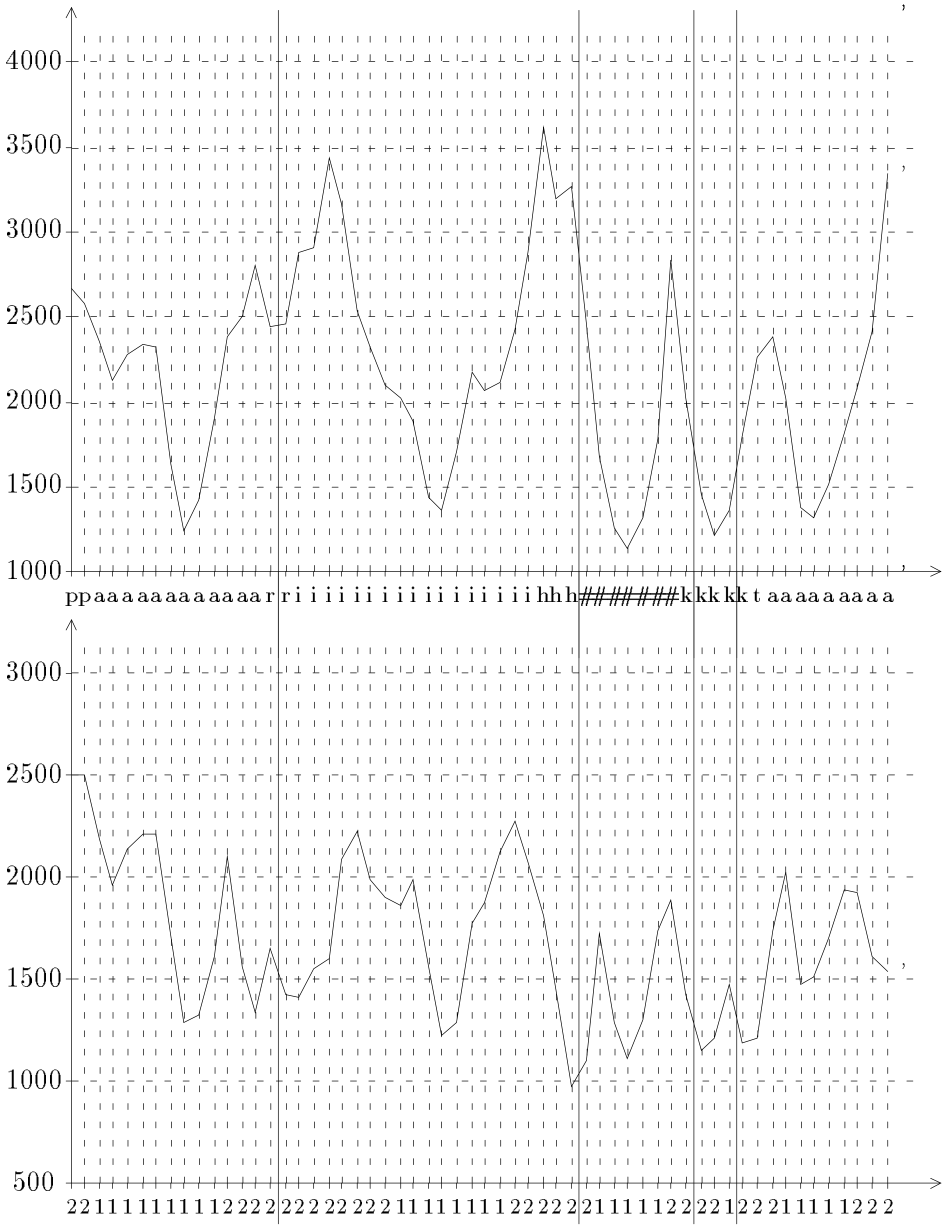

Figure 4:

Representation of context	time span (ms)	dim.	cb size 500 correct%	cb size 1000 correct%	cb size 2000 correct%
I	10	20	87.0	86.9	86.6
[....]	60	20	93.0	93.4	93.1
IIIIII	60	120	95.2	96.0	96.0
[] [] []	60	60	95.5	96.0	95.9
[] [] [] [] [] [] [] [] [] [] []	220	220	95.7	97.9	98.8
[.] [.] [.] [.] [.] [.] [.]	210	140	95.9	97.9	98.8
[..] [] [] [] [..]	140	100	97.0	97.9	98.2
[..] [..] [] [] [] [..] [..]	220	140	97.1	98.5	99.0

Table 1:

Representation of context	time span (ms)	dim.	cb size 500 correct%	cb size 2000 correct%
I	10	20	75.4	73.8
[]	20	20	77.0	76.0
[.]	30	20	80.8	80.5
[] [] []	60	60	89.8	91.8
[.] [.] [.]	120	60	90.5	92.9
[] [] [] [] []	100	100	91.5	93.9
[.] [.] [.] [.] [.]	150	100	91.3	94.0
[..] [] [] [] [..]	140	100	92.3	94.5
[..] [..] [] [] [] [..] [..]	220	140	91.9	94.6

Table 2:

Information included	correct%	errors%
STF	93.9	9.5
FON	97.1	4.6
FON, dQE	97.0	4.1
FON, cQE	97.6	3.8
FON, SEG	98.4	3.2
FON, SEG, dQE	98.1	3.2
FON, SEG, cQE	98.4	2.9
FON/SEG	98.5	3.0
FON/SEG, dQE	98.0	3.2
FON/SEG, cQE	98.5	2.8

Table 3:

Figure and Table captions

Figure 1. Phoneme token recognition accuracy as a function of the number of concatenated successive short-time feature vectors using four different codebook sizes.

Figure 2. Phoneme token recognition accuracy as a function of the number of concatenated averages of two short-time feature vectors using four different codebook sizes.

Figure 3. The chosen 140-dimensional context vector

Figure 4. Segmentation information for the Finnish word 'aika'. Top graph displays recognized quasiphonemes together with their quantization errors. Manual segmentations (not used in experiments) are shown as vertical lines (phonemes /a/, /i/, /k/, /a/). Character '#' denotes silence. Respective segmentation classifications are shown in the lower graph. '1' and '2' stand for stationary regions and transitions, respectively. Note that diphthong /ai/ is here considered as a concatenation of two vowels.

Table 1. Phoneme token recognition results using LVQ. `I` stands for a single short-time feature vector, `[]` denotes average of two, `[.]` average of three, `[..]` average of four, and so on. For example, `[..][]` denotes an average of four feature vectors concatenated with an average of two. "cb size" refers to codebook size.

Table 2. Phoneme center/border classification accuracy by LVQ. Context representations are explained in the caption of Table 1. "cb size" refers to codebook size.

Table 3. Results of transcribing speech into phoneme strings with HMMs using different LVQ-codebook variations. "FON" stands for phoneme classification codebook, "SEG" for segmentation codebook, "dQE" and "cQE" for quantization error in discrete and continuos form, respectively. "FON/SEG" denotes that the two code sequences have been combined to a single one using two different symbols for each phoneme, determined by the two possible symbols of segmentation codebook. For comparison, results obtained by using a short-time feature vector based LVQ-codebook ("STF") are given. The "errors"-column includes all substitution, deletion and insertion errors.