An Efficient Method to Estimate Pronunciation from Multiple Utterances

Tofigh Naghibi, Sarah Hoffmann, Beat Pfister

Speech Processing Group, ETH Zurich, Switzerland
{naghibi,hoffmann,pfister}@tik.ee.ethz.ch

Abstract

Given K utterances of a word and a set of sub-word units one may need a generalization of the conventional one-dimensional Viterbi algorithm to jointly decode them in order to derive their underlying word model (pronunciation). This extension is called *k*-dimensional Viterbi. However, as the number of utterances increases, the complexity of the *k*-dimensional Viterbi algorithm exponentially increases causing prohibitive computational burden. Here, we propose an approximation algorithm for the *k*-dimensional Viterbi which efficiently uses the available utterances to estimate the pronunciation. In addition to automatic dictionary generation, it can be used in computationally expensive applications such as lexicon-free training and joint pattern alignment.

Index Terms: pronunciation, joint decoding, *k*-dimensional, viterbi

1. Introduction

In modern HMM-based speech recognizers sub-word units which are considered the basic acoustic elements, are modeled by a sequence of hidden Markov states. Word models (or even sentence models) can then be created by concatenating the HMMs corresponding to the sub-words. It is common to call these word models *pronunciations* especially if the sub-words are phonemes. However, in general, sub-word units do not necessarily need to be linguistically motivated elements and can simply be obtained by clustering the acoustic space. In this case they may be referred to as abstract acoustic elements (AAE). Given an observation sequence, the recognition problem then can be seen as selecting the most likely pronunciation (or a sequence of them) of the dictionary that can describe the observation sequence best. However, there are applications in which pronunciations cannot be obtained from the pronunciation lexicon. For instance, there is no available phonetic based lexicon for some languages or some of the standard pronunciations are not entirely correct for different dialects of a language. In addition, when AAEs are used as the sub-word units, the pronunciations are anyway not known in advance and have to be estimated. Therefore, a data-driven approach for pronunciation learning is necessary to address this problem.

The main challenge here is to generate a reliable pronunciation from the recorded data. Obviously, when there is only one utterance (also called realization or pattern in literature) of a word in the speech corpus, the word pronunciation can be achieved by computing its optimal sub-word sequence which is unreliable and probably not generalizable to other utterances of that word. However, the more interesting case is when several utterances of a word are available. The problem then is to find a pronunciation that best describes all utterances of the word. A variety of solutions have been proposed for this problem which are mainly heuristics. In [10][12][13] n-best hypotheses are generated for each word and the best candidate is selected based on different criteria. A* based algorithm has been proposed in [1] to search through the trellis. It uses a heuristic estimate of the likelihood to determine the order in which the search visits nodes in the tree. However, as all A* based algorithms, its main short-come is its heuristic estimate of the likelihood. A voting approach proposed in [5] is another commonly used solution of this problem. It considers the most frequent pronunciation as the best pronunciation of a word.

Unlike these heuristic methods, an approach based on maximizing the joint likelihood of multiple audio recordings of a word has been suggested in [6] to determine a word pronunciation. This algorithm is called *k*-dimensional Viterbi and is the base algorithm of this work. A very similar approach based on a different version of the multi-dimensional Viterbi algorithm has been suggested in [2] to determine a word pronunciation. However it uses the multi-dimensional Viterbi algorithm proposed by Nair et al. in [11] which does not provide a probabilistic framework and thus can not find the maximum likelihood solution. However, both of these algorithms have a complexity of $O(T^K)$ where T is the average length of utterances of a word and k is the number of utterances. The exponential increase of the complexity with the number of utterances prevents them from using more than a very few utterances and thus estimating a reliable pronunciation.

Here, we formulate the pronunciation estimation problem and present an approximation algorithm for the *k*-dimensional Viterbi that can efficiently be employed with a large number of utterances to generate a reliable pronunciation. This algorithm breaks down the difficult task of joint decoding of k utterances into $k-1$ applications of the two-dimensional Viterbi algorithm and thus its complexity is about $O((k-1)T^2)$.

Besides pronunciation estimation as the main aim of this approximation algorithm, the proposed approach provides a powerful tool for embedded training of AAEs. In this problem, sub-words are AAEs and consequently, there is no lexicon available to achieve the pronunciations. Furthermore, the training data is a set of sentences. Thus, a fast algorithm is needed to take information from all utterances of a word into account to construct a dictionary which is used for segmentation [7]. Another potential application for our algorithm is the well-known joint alignment problem appearing in various fields such as bioinformatics, video processing [4] and biology [9]. For instance, an immediate application for our algorithm is to align several frame sequences recorded at different time by independent moving cameras that follows a similar trajectory. This problem has been discussed in [4].

2. Problem Specification

A hidden Markov model can be defined by a pair of stochastic process: Markov state process and observable process where the

observable process is an incomplete observation of the Markov state process. In an HMM of a word, the word pronunciation together with the HMM topology constitute a Markov state process that can generate different sample paths (different hidden state sequences). For instance, let a pronunciation $\boldsymbol{\omega}$ be a sub-word sequence $\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3$ where $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_3$ represent some arbitrary sub-word units (phonemes or AAEs). Moreover, assume it has a left-right Markov topology with no skipping of sub-words permitted, a common model topology in HMM-based systems. Some sample paths of this stochastic process are then $\mathbf{A}_1\mathbf{A}_1\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3$, $\mathbf{A}_1\mathbf{A}_1\mathbf{A}_2\mathbf{A}_2\mathbf{A}_3$, $\mathbf{A}_1\mathbf{A}_2\mathbf{A}_2\mathbf{A}_2\mathbf{A}_3\mathbf{A}_3$. We may denote a set of all sample paths generated by a pronunciation $\boldsymbol{\omega}$ with $\mathbb{S}_{\boldsymbol{\omega}}$.

When a pronunciation is not available, it can be estimated from one utterance of the word by finding the most probable pronunciation that has generated the given utterance, though it is very probable that it will not be the optimal sub-word sequence for other utterances of that word. Therefore, in pronunciation estimation all available utterances of a word should be taken into account. This problem can be defined more formally as follows:

Given K utterances $\mathbf{X}^1_{1:T_1}, \cdots, \mathbf{X}^K_{1:T_K}$ of a word with lengths $T_1 \cdots, T_K$ and a set of M sub-word HMMs, $\mathbf{A}_1, \cdots, \mathbf{A}_M$, find a pronunciation that with the highest probability produces the given utterances. The pronunciation estimation problem can be stated as:

$$\boldsymbol{\omega}_K = \underset{\boldsymbol{\omega}}{\operatorname{argmax}} \prod_{i=1}^{K} \max_{\mathbf{S}^i} P(\mathbf{X}^i_{1:T_i}, \mathbf{S}^i|\lambda) \qquad (1)$$
$$\text{subject to:} \quad \mathbf{S}^i \in \mathbb{S}_{\boldsymbol{\omega}} \quad \forall i \in \{1, \cdots, K\}$$

where $\mathbf{S}^1, \cdots, \mathbf{S}^K$ are the hidden state sequences (sample paths) corresponding to $\mathbf{X}^1_{1:T_1}$ to $\mathbf{X}^K_{1:T_K}$, respectively. Here, λ denotes the loop of sub-word HMMs that are connected in parallel and $\boldsymbol{\omega}_K$ is the pronunciation estimated from the K given word utterances. The constraint in (1) implies that all hidden state sequences are sample paths of the same pronunciation. A solution for this optimization problem for a case when $\boldsymbol{\omega}$ has a left-to-right Markov state topology, has been suggested in [6].

However, we may further investigate this problem by pointing out that provided the Viterbi approximation is sufficiently precise [3, p. 44], the solution of the problem in (1) will be the maximum likelihood (ML) estimation of the pronunciation.

It can readily be proved by noting the fact that the ML estimate of a pronunciation can be found by maximizing $P(\mathbf{X}^1_{1:T_1}, \cdots, \mathbf{X}^K_{1:T_K}|\boldsymbol{\omega}) = \prod_{i=1}^{K} P(\mathbf{X}^i_{1:T_i}|\boldsymbol{\omega})$ over all valid pronunciations, $\boldsymbol{\omega}$. By approximating the $P(\mathbf{X}^i_{1:T_i}|\boldsymbol{\omega})$ terms with their maximum probability state sequences (Viterbi approximation) we obtain the formulation in (1).

A direct corollary of the above discussion is that for $K \to \infty$ the estimation will converge to the exact pronunciation following the fact that when the sample size tends to infinity, ML estimators are unbiased and converge in probability to their true values. This suggests that provided there are enough repetitions of a word in a database it is more precise to estimate the pronunciations from the data even in applications where they can be obtained from a lexicon.

As mentioned before, the suggested solution in [6] to solve this optimization problem is essentially an extension of one-dimensional Viterbi algorithm to *k* dimensions. However, the complexity of the algorithm grows exponentially with respect to the number of utterances which prevents from using it with more than a very few utterances. Based on our experiments, the algorithm is not tractable for $K > 3$ in real applications. To address this problem, in the following section an approximation algorithm is suggested that can efficiently use information of all available utterances to estimate the pronunciation.

3. K-dimensional Viterbi Approximation

The *k*-dimensional Viterbi problem is actually a bi-linear problem involving the product of two variables: the joint alignment and the optimal sequences. Given the optimal joint alignment of the utterances, the problem reduces to a linear programming type problem that can be solved by the Viterbi algorithm.

The approximation algorithm proposed here utilizes this idea to first estimate the optimal joint alignment and use this alignment to estimate the pronunciation. This can be done in $K-1$ iterative steps. At each step, we align a new utterance with a virtual utterance constructed from the previously used utterances by means of the two-dimensional Viterbi and output an updated virtual utterance. A virtual utterance basically is a sequence of elements which we call buckets. Each bucket contains a group of aligned features. Moreover, all features of a bucket are emitted from the same hidden state. The concept of virtual utterance and buckets are discussed more profoundly in the next subsection.

We denote the observation sequence of the i^{th} utterance by $\mathbf{X}^i_{1:T_i} = x^i_1 x^i_2 \cdots x^i_{T_i}$ where x^i_l is the feature vector of the i^{th} utterance at time frame l, the state-j emission probability or the observation probability of the x^i_l with $b_j(x^i_l)$, the transition probability between state i and j in HMMs with a_{ij} and finally, a virtual utterance constructed by i number of utterances with $\mathbf{Z}^i = z^i_1 z^i_2 \cdots z^i_{T_{Z_i}}$, where T_{Z_i} is its length and z^i_n is the n^{th} bucket in the bucket sequence $\mathbf{Z}^i$. Using these notations, the algorithm can be described as follows:

1. Initialize the first virtual utterance $\mathbf{Z}^1$ by putting each feature of the first utterance in one bucket, i.e., $T_{Z_1} = T_1$ and $z^1_l = \{x^1_l\} \quad \forall l \le T_1$.
2. for $i = 2 : K$
   (a) Align the i^{th} utterance $\mathbf{X}^i_{1:T_i}$ with $\mathbf{Z}^{i-1}$ by means of the two-dimensional Viterbi algorithm.
   (b) Construct the new virtual utterance $\mathbf{Z}^i$.
3. Construct the pronunciation from the optimal state sequence of $\mathbf{Z}^K$.

The above lines show the general framework of the algorithm. However, we still need to elaborate the iteration steps 2a and 2b. In the two following subsections we try to investigate and clarify these two steps. The step 2b is explained first.

3.1. Virtual utterance

A virtual utterance is a sequence of buckets as opposed to a sequence of feature vectors in real utterances. Each bucket is a set of features and has to satisfy three conditions:

1. All features in a bucket should be emitted from the same hidden state.
2. In each bucket there exist i sub-sequence of features (with at least size one): one from each of the i utterances.
3. Buckets can not be split. They only can be merged. This condition is actually essential to make the algorithm tractable. It means at each iteration we can only merge the buckets resulting in losing some degrees of freedom. Therefore, the utterances are sorted by length (longest first) to start with the maximum number of buckets.

Moreover, there is an additional constraint on the virtual utterance forcing each feature of the utterances that construct the virtual utterance to be in one and only one bucket. The last constraint is just to ensure that all available feature vectors are used, and only used once.

In step 2b of the algorithm, we construct a new virtual utterance $\mathbf{Z}^i$ by including the features of the new utterance to the buckets and merging some of the buckets. This procedure is clarified in the following example.

Consider a virtual utterance $\mathbf{Y} = y_1y_2y_3y_4y_5y_6$ and a real utterance $\mathbf{X} = x_1x_2x_3x_4x_5$, given their hidden state sequences as $\mathbf{S}^y = S_1^yS_2^yS_2^yS_2^yS_3^yS_3^y$ and $\mathbf{S}^x = S_1^xS_1^xS_2^xS_2^xS_3^x$ (which are available from the two-dimensional Viterbi in step 2a), several valid virtual utterances can be constructed from them. One example can be $\mathbf{Z} = z_1z_2z_3$ with z_1=$\{y_1, x_1, x_2\}$, z_2=$\{y_2, y_3, y_4, x_3, x_4\}$, z_3=$\{y_5, y_6, x_5\}$.

However, in this work, we add a few additional constraints (which are not necessary and can be replaced with other constraints) to remove the grouping ambiguity and obtain exactly one virtual utterance. Perhaps the two most important ones are first, a bucket size (number of features in a bucket) should be as small as possible or equivalently, the number of buckets should be as large as possible and second, features should be distributed as evenly as possible in buckets.

By applying these new constraints, a valid virtual utterance in the above example will be $\mathbf{Z} = z_1z_2z_3z_4$ where z_1=$\{y_1, x_1, x_2\}$, z_2=$\{y_2, y_3, x_3\}$, z_3=$\{y_4, x_4\}$, z_4=$\{y_5, y_6, x_5\}$. This example is illustrated in Figure 1.

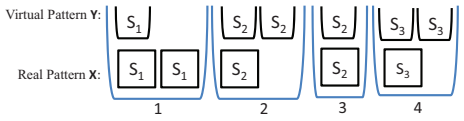

Figure 1: A virtual utterance Z, sequence of four buckets, constructed from a virtual utterance $\mathbf{Y}$ and a real utterance $\mathbf{X}$.

3.2. Aligning a virtual utterance with a real one

In step 2a, the two-dimensional Viterbi can be employed to obtain the optimal state sequences of the virtual utterance $\mathbf{Z}^{i-1}$ and the real utterance $\mathbf{X}^i_{1:T_i}$. Since the two-dimensional Viterbi guarantees that state sequences are from the same pronunciation (generated by the same sub-words sequence), the time alignment can be readily achieved. However, the observation probabilities and state transition probabilities have to be computed differently for virtual utterances than real utterances. As in the previous subsection, we use an example to clarify this point.

Consider an utterance $\mathbf{X} = x_1x_2x_3x_4x_5x_6$ with a hidden state sequence $S_1S_2S_2S_2S_3S_3$ and assume $\mathbf{Z} = z_1z_2z_3$ is a virtual utterance constructed from this utterance such that z_1=$\{x_1\}$, z_2=$\{x_2, x_3, x_4\}$, z_3=$\{x_5, x_6\}$. Since some of the buckets contain more than one feature, the observation probability of a bucket is the product of the observation probabilities of all the features in a bucket. For instance $b_j(z_2)$=$b_j(x_2)b_j(x_3)b_j(x_4)$. In general, the state j emission probability of a bucket z_l^i, can be calculated as

$$b_j(z_l^i) = \prod_{x_n^m \in z_l^i} b_j(x_n^m) \tag{2}$$

Furthermore, the transition probabilities should also be modified in accordance with the number of features in each bucket. In our example, the transition probability from a state n at time 1 (assume z_1 is emitted at this time point) to state m at time 2 (z_2 is emitted) consists of a product of two terms. The first term which we may call cross transition probability is a_{nm} as in a real utterance. However, the second term or self transition probability comes from the fact that z_2 contains more than one feature and since we assumed those features have been emitted from the same state, they introduce a hidden stay-in-state probability, equal to $a_{mm}^{|z_2|-1}$=a_{mm}^2 where $|z_2|$ is the number of features in the bucket z_2. Generalizing this example to a case where a virtual utterance has been constructed from i utterances is straightforward. In this case, the transition probability from the state n at time l−1 to state m at time l will be:

$$a_{nm}(z_l^i) = a_{nm}^i a_{mm}^{|z_l^i|-i} \tag{3}$$

Since there are i sub-sequences and in total $|z_l^i|$ features in a bucket, $|z_l^i|-i$ of them need to be modeled by a_{mm} and i of them by a_{nm} resulting in $a_{nm}^i a_{mm}^{|z_l^i|-i}$. It is noteworthy that unlike real utterances, the transition probabilities are bucket dependent in virtual utterances. Thus the argument, z_l^i has been utilized in $a_{nm}(z_l^i)$ to emphasize on this fact.

We do not describe the two-dimensional Viterbi algorithm here. However, for the purposes of this paper, it suffices to know that its outputs are two optimal states sequences generated with the same pronunciation. Having all HMM parameters at hand, we can use the two-dimensional Viterbi to align the virtual pattern and real pattern in step 2a.

4. Experiments

Based on the proposed algorithm, we carried out several isolated-word speech recognition experiments and compared the proposed approximation algorithm with the *k*-dimensional Viterbi algorithm and an n-best solutions approach. For our experiments we used TIMIT data [8] to train two kinds of subword units: phonemes and acoustic abstract elements (AAE). Both systems employ 13 MFCC and their delta and delta-delta coefficients. We used the phoneme-based recognizer to compare the quality of the estimated pronunciations with the lexicon pronunciations and used the AAE-based recognizer to compare the different pronunciation estimation approaches with each other. The following test scheme was used in the first two experiments to generate statistically reliable results.

For pronunciation estimation, we collected a pronunciation learning set comprising 210 distinct words with more than 10 utterances for each from the training set. This set was used to construct our dictionary (pronunciation estimation step). In both experiments, for 10 times we randomly selected k utterances (k from 3 to 10) of each word and constructed a dictionary containing all these 210 pronunciations. Each dictionary was then divided to 21 distinct test dictionaries containing a set of 10 words. For each test dictionary, we ran the recognition test on a held out test set containing 60 examples, i.e., 6 utterances for each word. The recognition rate was then the average of the recognition rates of all these 10 × 21 tests. This test scheme guarantees that the results are statistically meaningful and comparable.

For phoneme-based speech recognizer in our fist experiment, we used 46 phones from the standard TIMIT phone set and each phone was modeled by a three-state, left-to-right HMM with six independent Gaussian mixtures. The HMMs then were trained using HTK tools [14]. In this experiment, we first assumed the pronunciations were unknown and estimated

Methods	Number of utterances							
	3	4	5	6	7	8	9	10
k-dimensional Viterbi	4.12 ± 0.26	-	-	-	-	-	-	-
Approximation	4.40 ± 0.28	4.47 ± 0.28	3.69 ± 0.28	3.78 ± 0.28	3.24 ± 0.22	3.65 ± 0.28	3.04 ± 0.20	3.46 ± 0.20
n-best	8.01 ± 0.38	8.39 ± 0.40	8.93 ± 0.44	9.27 ± 0.40	9.77 ± 0.44	8.91 ± 0.40	9.95 ± 0.48	10.35 ± 0.46

Table 1: Comparison of three different pronunciation estimation approaches. WER of the AAE-based recognizer for each experiment and its standard deviation are shown in this table.

them with our approximation algorithm for $3 \leq k \leq 10$. The word error rates (WER) of these experiments as well as their confidence intervals have been illustrated by the dash line in Figure 2. As can be seen the recognition rate significantly improves by taking more utterances for pronunciation estimation into account. We again ran the recognition test with the reference pronunciations obtained from the lexicon and depicted the result with the solid line in Figure 2. As expected, even using four utterances to estimate the pronunciation leads to lower WER than using the reference pronunciations. It actually follows the fact that the k-dimensional Viterbi is the optimal estimation of the pronunciation in a sense of maximum likelihood. The variations of the word error rates are because of the statistical nature of the proposed algorithm but the general trend is toward decreasing the WER. They also show that the estimation has not yet converged to the optimal pronunciation and using more utterances may result in more reduction of error rates.

The AAE-based speech recognizer was used for the second experiment. In this test, we used 64 AAEs, each consisting of a 1-state HMM made up of 16 mixtures trained on the TIMIT data. Details of the embedded training approach used to train them can be found in [7]. The goal of the experiment was to compare the pronunciations obtained by our approximation algorithm with those obtained from k-dimensional Viterbi and n-best approach which is a commonly used algorithm in this context. In the n-best approach, n pronunciation hypotheses were generated from the available utterances and were scored against other utterances to find the most likely pronunciation. Table 1 shows the comparison. The n-best complexity is $O(T^2)$, i.e., almost the same as our algorithm with $O((k-1)T^2)$. However, as can be seen, its performance is significantly worse than the proposed algorithm. As expected, the k-dimensional Viterbi was not tractable for more than three utterances. However, for three utterances it obtains a slightly better recognition rate than the approximation. Although with some variations, the results reported in both Table 1 and Figure 2 show the general trend of lower WER as the number of utterances increases. The standard deviations reported for the experiments statistically support this conclusion.

Unlike the previous tests, in the last experiment we used a dictionary containing 210 words. The results of the phoneme-based speech recognizer for both scenarios of the reference and the estimated pronunciations have been illustrated in Figure 3. We also included the AAE-based recognizer results. Its results are comparable and even better than the utilized phoneme-based recognizer. As before, the estimated pronunciations resulted in lower error rates than the reference pronunciations.

5. Conclusion

In this work, we have presented a maximum likelihood based approach to learn the pronunciations of words. Our results showed that the proposed algorithm can outperform the common solutions of this problem while maintaining similar low

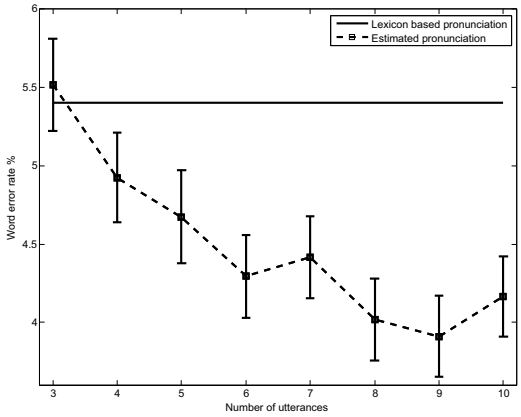

Figure 2: Word error rate of phoneme-based speech recognizer with estimated pronunciations and lexicon pronunciations

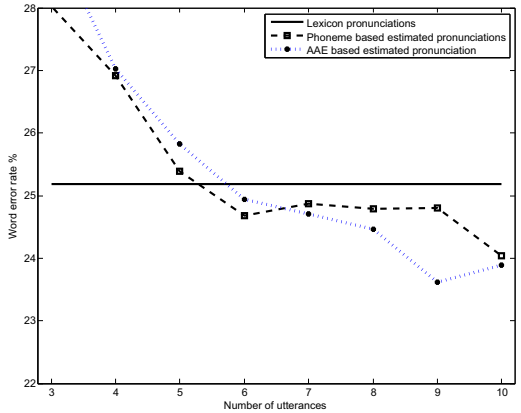

Figure 3: Comparison of phoneme-based recognizer and AAE-based recognizer on a test set with 210 distinct words without cross validation.

complexity. Moreover, our experiments showed that the estimated pronunciations lead to higher recognition rates than reference (lexicon) pronunciations and therefore, can be used for automatic lexicon generation. Much room for improvement remains, however. In this work, we assumed all utterances have equal weights. However, it is possible to adjust the weights of utterances based on prior information about intelligibility, etc. Furthermore, by making a few adjustments to the k-dimensional Viterbi algorithm, it is possible to generate a set of most likely pronunciations rather than just the most likely one.

6. Acknowledgments

This work was supported by the Swiss Innovation Promotion Agency CTI. The authors also wish to acknowledge the anonymous reviewers for their detailed and helpful comments.

7. References

[1] 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 Transactions on Speech and Audio Processing*, 1(4):443–452, oct 1993.

[2] D. Bansal, N. Nair, R. Singh, and B. Raj. A joint decoding algorithm for multiple-example-based addition of words to a pronunciation lexicon. In *IEEE International Conference onAcoustics, Speech and Signal Processing (ICASSP)*, pages 4293 –4296, april 2009.

[3] H. A. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[4] F. Diego, J. Serrat, and A.M. Lopez. Joint spatio–temporal alignment of sequences.

[5] J.G. Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *IEEE Workshop on Automatic Speech Recognition and Understanding.*, pages 347–354, dec 1997.

[6] M. Gerber, T. Kaufmann, and B. Pfister. Extended viterbi algorithm for optimized word hmms. In *IEEE International Conference onAcoustics, Speech and Signal Processing (ICASSP)*, pages 4932–4935, may 2011.

[7] S. Hoffmann, T. Naghibi, and B. Pfister. Embedded training for language-independent acoustic sub-word units. In *Proceedings of Interspeech*. Manuscript submitted for publication.

[8] et al. John S. Garofolo. Timit acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium, Philadelphia*, 1993.

[9] G. Lunter, I. Miklós, A. Drummond, L. Jensen, and J. Hein. Bayesian coestimation of phylogeny and sequence alignment. *BMC Bioinformatics*, 6:1–10, 2005.

[10] H. Mokbel and D. Jouvet. Derivation of the optimal set of phonetic transcriptions for a word from its acoustic realizations. *Speech Communication*, 29(1):49–64, 1999.

[11] N.U. Nair and T.V. Sreenivas. Viterbi algorithm for multi-pattern joint decoding. In *TENCON 2009 - 2009 IEEE Region 10 Conference*, pages 1 –5, jan. 2009.

[12] R. Singh, B. Raj, and R.M. Stern. Automatic generation of subword units for speech recognition systems. *IEEE Transactions on Speech and Audio Processing*, 10(2):89–99, feb 2002.

[13] T. Svendsen. Pronunciation modeling for speech technology. In *International Conference on Signal Processing and Communications (SPCOM)*, pages 11–16, dec. 2004.

[14] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book, version 3.4*. Cambridge University Engineering Department, Cambridge, UK, 2006.