MINIMUM EXACT WORD ERROR TRAINING

G. Heigold, W. Macherey, R. Schlüter, H. Ney

Lehrstuhl für Informatik 6 - Computer Science Dept.
RWTH Aachen University, Aachen, Germany
{heigold,w.macherey,schlueter,ney}@cs.rwth-aachen.de

ABSTRACT

In this paper we present the *Minimum Exact Word Error* (exactMWE) training criterion to optimise the parameters of large scale speech recognition systems. The exactMWE criterion is similar to the *Minimum Word Error* (MWE) criterion, which minimises the *expected* word error, but uses the *exact* word error instead of an approximation based on time alignments as used in the MWE criterion. It is shown that the exact word error for all word sequence hypotheses can be represented on a word lattice. This can be accomplished using transducer-based methods. The result is a word lattice of slightly refined topology. The accumulated weights of each path through such a lattice then represent the exact number of word errors for the corresponding word sequence hypothesis. Using this compressed representation of the word error of all word sequences represented in the original lattice, exactMWE can be performed using the same lattice-based re-estimation process as for MWE training. First experiments on the *Wall Street Journal* dictation task do not show significant differences in recognition performance between exactMWE and MWE at comparable computational complexity and convergence behaviour of the training.

1. INTRODUCTION

Due to improved optimisation procedures and increased computational power, discriminative methods have become an important means of estimating the parameters of *Hidden Markov Models* (HMM) in many state-of-the-art speech recognition systems. Since the first successful application of the *Maximum Mutual Information* (MMI) criterion to large scale speech recognition tasks [1], there has been a growing interest in a class of error minimising discriminative training criteria, like for example the *Minimum Word Error* (MWE) and the *Minimum Phone Error* (MPE) criterion [2, 3]. In contrast to the MMI criterion, which directly maximises the posterior probability of the training utterances, MWE and MPE aim at minimising the (expected) word and phoneme error on the training data. The MWE and MPE criterion were shown to significantly outperform the MMI criterion on many tasks [2, 3, 4].

As the standard evaluation measure of most speech recognition systems is the word error rate (WER), one would ideally like to minimise the *empirical* word error in training. However, the implementation is not easy for several reasons. First, the objective function is a discrete valued function and therefore has to be approximated by a smoothed objective function to allow an efficient optimisation. In this work, the objective function will be the *expected* word error, on which both MWE and exactMWE are based. Clearly, [3] introduced a misnomer but we have used this terminology to emphasise the similarity of our proposed algorithm with MWE. Second, the key quantity of the expected word error is the word error. The calculation of the exact word errors requires an alignment, which so far has been solved only on N-best lists. Finally, the discriminative statistics need to be accumulated efficiently. Typically this comprises a summation over the word sequences.

In the literature, different approximations are proposed to tackle the minimisation of the expected word error. In [5], N-best lists are used in conjuction with the exact word error, whereas in [6] the word error is approximated by consensus lattices which are expanded to lists for accumulation. In contrast, [3] and [4] use an approximation of the word error, which allows to work on lattices in all stages of the training. [4] also describes a more refined approximation of the word error, which is misleadingly termed 'exact MWE'. It is, however, not clear how much recognition performance is lost by these approximations.

In this paper, we propose an algorithm to *exactly* solve the minimisation of the expected word error on word lattices in all stages of the training. This enables quality assessments of the various approximations of the word error and comparisons of N-best lists with lattices. The algorithm might be also useful in other applications where, for instance, no time alignment is available, which is needed for some approximations of the word error (e.g. [3]).

The calculation of word errors on lattices is based on weighted transducers. In the literature, similar problems have been discussed.

- The edit-distance of two unweighted automata may be calculated with a transducer-based approach [7].
- [7] also described an algorithm to calculate the edit-distance of two weighted automata, which may be used for re-scoring according to the Bayes risk with the exact word error in speech recognition. However, the feasibility of this algorithm has not been demonstrated yet.

2. TRANSDUCER-BASED CALCULATION OF EDIT-DISTANCES

The minimisation of the expected word error requires a summation over the word sequences and, thus, is usually performed on word lattices since this is a compact way to store many word sequences. As a consequence the word error of any sentence in the word lattice and the spoken sentence has to be calculated. For this purpose, word lattices here are considered unweighted automata where each path of the automaton represents a word sequence. To calculate the word errors efficiently, we will take advantage of the compact transducer representation.

2.1. Edit-Distance

Let Σ be a finite alphabet of distinct symbols, and let $\Sigma^\star$ denote the set of all possible strings given Σ. The set of local edit operations is defined as the set $\mathcal{E} = \Sigma \times \Sigma \cup \Sigma \times \{\epsilon\} \cup \{\epsilon\} \times \Sigma$ and each local edit operation is assigned costs $c : \mathcal{E} \rightarrow \mathbb{R}^+$. Furthermore, an element $w \in \mathcal{E}^\star$ is called an *alignment* of the strings x and y if $h(w) = (x, y)$ for the corresponding morphism $h : \mathcal{E}^\star \rightarrow \Sigma^\star \times \Sigma^\star$. Then, the edit-distance of these two strings is

$$d(x, y) := \min_{w \in \mathcal{E}^\star : h(w) = (x,y)} \sum_i c(w_i),$$

where w_i are the local edit operations of w. The classical edit-distance is obtained if the costs are set to 1 except for matches which have zero costs [8]. The edit-distance of two strings is efficiently solved by dynamic programming.

2.2. Transducer Representation

The alignments including the costs may be represented as transducers in a natural way. To build the alignments, it is convenient to introduce the *edit-distance transducer* as illustrated in Fig. 1. The arcs are labelled with the triple "input:output/weight", each describing a local edit operation with the corresponding costs (e.g. "ϵ : *a*/1" represents an insertion). ϵ denotes the empty symbol.

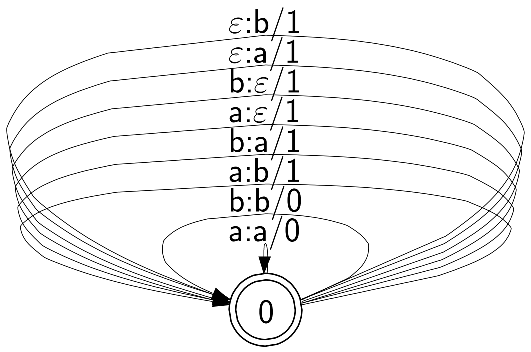

Fig. 1. *Edit-Distance transducer for the alphabet* $\{a,b\}$.

Given two unweighted automata, A_1 and A_2, and the edit-distance transducer L, the weighted automaton

$$A_1 \circ L \circ A_2$$

($\circ$: composition) contains all alignments of any string in A_1 and any string in A_2 [7]. The edit-distance of A_1 and A_2[1], for example, is efficiently calculated by means of a single-source shortest-path algorithm such as best from FSA [9] in combination with the tropical semiring $\mathcal{T}$, i.e.,

$$\mathsf{best}_\mathcal{T}(A_1 \circ L \circ A_2) \tag{1}$$

returns the alignment with the lowest costs [7]. In the case of speech recognition, A_1 represents the spoken sentence. If A_2 contains only the recognised sentence, then Eq. (1) produces the word error of these two sentences. If A_2 is set to the word lattice, Eq. (1) can be used to calculate the graph error rate (GER).

Note that this concept is general: any weighted transducer without cycles with negative weight might be substituted for the edit-distance transducer. As a variant of the classical edit-distance, for example, the weights of the edit-distance transducer can be set to the values, which have been estimated from a stochastic model for edit-distance [10].

[1] $d(A_1, A_2) := \min_{x \in A_1, y \in A_2} d(x, y)$

2.3. Algorithm

A similar problem involves finding the edit-distances for all strings in A_2 (e.g. word sequences in a word lattice) given the reference(s) in A_1 (e.g. correct word sequence). More formally: given two unweighted automata, A_1 and A_2, find a weighted automaton with arc weigths $w[a]$ such that the weight of any path $\pi \in A_2$ with string x satisfies the constraint

$$w[\pi] = d(A_1, x),$$

i.e., the arc weights are distributed over the automaton so that the accumulated weights provide the edit-distance for each string of A_2 given the reference(s) in A_1. The weight of path π is obtained by summing up the corresponding arc weights. A solution to this problem is to use standard weighted transducer algorithms. An overview of the relevant transducer algorithms is found in Table 1 [9].

Proposition *If A_2 is acyclic, the weighted automaton*

$$\mathsf{det}_\mathcal{T}(\mathsf{rm}\epsilon_\mathcal{T}(\mathsf{proj}_2(A_1 \circ L \circ A_2)))$$

is well-defined and satisfies $w[\pi] = d(A_1, \pi)$, $\forall \pi \in A_2$.
Proof. First, $A_1 \circ L \circ A_2$ is acyclic since A_2 is acyclic by assumption. According to [11] any acyclic weighted automaton has the twins property and, thus is determinisable, i.e., the weighted automaton is well-defined. Second, the determinisation produces a deterministic weighted automaton which is equivalent to the input automaton under the given, say, tropical semiring. A deterministic automaton has the properties [12]:

- unique initial state;
- no two transitions leaving any state share the same input label.

This definition implies that any string in the deterministic automaton is unique. From these observations, the correctness of the algorithm follows. □

The edit-distance transducer has a single state, but, it has $|\Sigma|^2$ arcs if $|\Sigma|$ is the alphabet size. In speech recognition with large vocabularies this is prohibitive. For this reason the vocabulary is reduced to the words occuring in A_1, and an "out-of-vocabulary" word is introduced onto which all words of A_2 which do not appear in A_1 are mapped. Thereby, different word sequences may be mapped onto the same word sequence. For training, however, all word sequences of the word lattice are required. So, the word sequences of the word lattice are recovered afterwards with an algorithm, which performs similarly to composition.

To estimate the complexity of this algorithm, it is assumed that the automata are connected, i.e., $|Q| \leq |E|$ where $|Q|$ and $|E|$ are the number of states and arcs. The transducer $A_1 \circ L \circ A_2$ has $\mathcal{O}(|E_1||E_2|)$ arcs. In the straighforward approach, this transducer might be calculated in $\mathcal{O}(|E_1|^2(|E_1| + |E_2|))$. The edit-distance transducer might be factorised so that this step takes only

Table 1. *Weighted transducer algorithms from the transducer toolkit FSA [9]. Complexities are given for connected automata.*

algorithm	description	complexity
$\circ$	composition	$\mathcal{O}(\lvert E_1\rvert\lvert E_2\rvert)$
proj$_2$	discard input labels	$\mathcal{O}(\lvert E\rvert)$
rmϵ	ϵ-removal	$\mathcal{O}(\lvert V\rvert\lvert E\rvert)$
det	produce equivalent deterministic automaton	$\mathcal{O}(\lvert E\rvert \log\lvert E\rvert)$ (acyclic)

$\mathcal{O}(|E_1||E_2|)$ [7]. As will be shown, this reduction has no impact on the overall complexity. Using the complexities of the individual algorithms from Table 1 then leads to the overall complexity

$$\mathcal{O}(|E_1|^2|E_2|^2 \log|E_1||E_2|).$$

Clearly the bottleneck of this algorithm is the determinisation performed by rmϵ (ϵ is not a special symbol in det) and det. Compared with N-best lists, this algorithm is generally more efficient because A_2 might represent up to $\mathcal{O}(B^{\frac{|E_2|}{B}})$ different strings (B: branching factor of A_2).

A simple optimisation might be to first minimise A_1 and A_2 which significantly speeds up the algorithm in the context of discriminative training. Several other optimisations are possible (e.g. pruning), which, however, do not guarantee the exactness of the algorithm in general.

3. MINIMISATION OF THE EXPECTED WORD ERROR

Let x_1^T be the feature vector, and v_1^M the correct word sequence. The acoustic and the language model probability are denoted by $p_\theta(x_1^T|v_1^M)$ and $p(v_1^M)$, respectively. The language model probabilities are supposed to be given. Hence, the parameter θ comprises the set of all parameters of the acoustic model. The parameter estimation problem then consists of finding $\hat{\theta}$ that optimises the objective function $\mathcal{F}_\theta(v_1^M, x_1^T)$. In speech recognition, the objective function ideally equals the empirical word error on the training corpus. This objective function, however, is not differentiable in θ, and, thus, has to be approximated by a differentiable function to allow gradient-based optimisation of the parameters.

3.1. Formulation

The objective function under consideration is the expected word error

$$\mathcal{F}_\theta(v_1^M, x_1^T) = \sum_{w_1^N} p_\theta(w_1^N|x_1^T) L[v_1^M, w_1^N],$$

where $p_\theta(w_1^N|x_1^T)$ is the sentence posterior probability and can be rewritten in terms of the acoustic and language model probabilities. $L[v_1^M, w_1^N]$ denotes the loss between the spoken sentence v_1^M and a word sequence hypothesis w_1^N of the word lattice, e.g. the word error. If this function is differentiable w.r.t. θ, the gradient can be built

$$\begin{aligned}\frac{\partial \mathcal{F}_\theta(v_1^M, x_1^T)}{\partial\theta} &= \sum_{w_1^N} p_\theta(w_1^N|x_1^T)\ \frac{\partial \log p_\theta(w_1^N, x_1^T)}{\partial\theta} \\ &\quad \cdot \Big[L[v_1^M, w_1^N] - \mathcal{F}_\theta(v_1^M, x_1^T)\Big] \qquad (2)\end{aligned}$$

and the *Extended Baum Welch* (EBW) algorithm, for instance, may be used for optimisation. The optimisation can be carried out if (2) can be (efficiently) calculated for each parameter of θ. This includes the implicit or explicit calculation of all $L[v_1^M, w_1^N]$.

Note that the smoothness of the objective function in θ can be controlled by the overall scaling factor of the probabilities. Usually the value is set so that the acoustic and language model scaling factors are the inverse of the language model scaling factor from recognition and 1, respectively. This has proven to be a robust choice. In the limit of an infinitely large scaling factor, the posterior probability mass concentrates on the word sequence with maximum a posterior probability, i.e., the objective function coincides with the empirical word error [4].

The next subsection summarises the approximate approach [3].

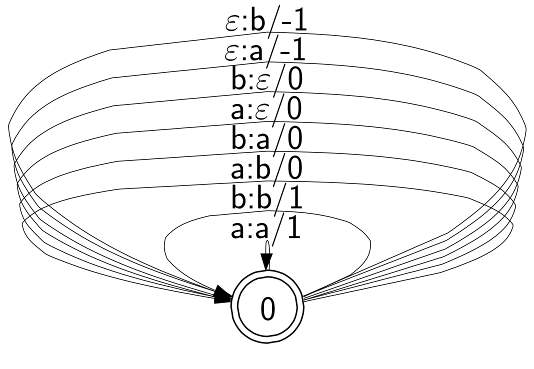

Fig. 2. *Accuracy transducer for the alphabet {a,b}.*

3.2. Word Accuracy

In [3], $L[v_1^M, w_1^N]$ is chosen as the word accuracy $A(v_1^M, w_1^N)$, which here is defined as the number of spoken words M minus the number of word errors. From this definition it follows that the expected word accuracy and the expected word error have the same local optima. Like the word error, the word accuracy can be modelled by a weighted transducer with arc weights 1, 0 and -1 for matches, substitutions/deletions and insertions, respectively, cf. Fig. 2.

To simplify and speed up the calculation of the word accuracies, the exact arc accuracies $A(q)$ may be replaced with approximate accuracies

$$A(q) = \max_z \begin{cases} -1 + 2e(q,z) & \text{if } z = q \\ -1 + e(q,z) & \text{otherwise,} \end{cases} \qquad (3)$$

where $e(q, z)$ is the relative overlap in time of the hypothesis arc q w.r.t. the correct arc z [3]. This approximation is based on the time alignments of the word sequences. It has the advantage that no edit-distance alignment is required and that the accuracies may be written in the original word lattices without changing the topology. In [3], a recursion formula is presented to calculate the corresponding gradient (2) on word lattices.

3.3. Exact Solution on Lattices

In the previous section, an algorithm has been presented to calculate the word errors of an unweighted transducer and store the results in form of a weighted transducer. To allow an efficient accumulation of the discriminative statistics, the word lattices need

Table 2. *Corpus statistics and vocabulary size on the North American Business (*NAB*) corpus.*

corpus	NAB-20k / NAB-65k		
	train	dev	eval
acoustic data [h]	81:23	0:48	0:53
# speakers	284	20	20
# sentences	37474	310	316
# running words	642074	7387	8193
# lexicon words	15013	19978 / 64735	

to be modified so that the word errors can be incorporated into the lattices without losing the information used for the acoustic rescoring (pronunciations, word start and end times, across word contexts).

First, the tokens used to evaluate the word errors are not identical to the pronunciations stored in the word lattice. The corresponding mapping is accomplished by composing the lattices with a suitable transducer.

Then, the word lattice with the word errors is obtained by composing the original word lattice and the weighted transducer containing the word errors. As the composition is based on the state mapping (q_1, q_2) and $(q_1', q_2') \rightarrow ((q_1, q_2), (q_1', q_2'))$ the times etc. may be recovered easily. It is important to avoid sentence hypotheses in the resulting word lattice, which have identical word sequence *and* time alignment, since this would affect the posterior probabilities entering the accumulation. To ensure this, the weighted transducer with the word errors has to be determinised beforehand. Observe that, in general, the composition may split states with the result that the resulting word lattices increase. This effect has been investigated and turned out to be uncritical, cf. next section. For the accumulation, the same recursion formula as for the approximate approach, i.e., MWE can be used [3].

4. EXPERIMENTAL RESULTS

Experiments were conducted on two settings of the *Wall Street Journal* (WSJ) corpora [13]. Table 2 summarizes some corpus statistics. The settings for the training are basically the same as in [2].

The Nov. '94 *North American Business* (NAB) training corpus consists of the 84 speakers of the WSJ0 corpus plus 200 additional speakers from the WSJ1 corpus. Tests were performed on the NAB Nov. '94 *Hub-1* evaluation corpus. Both the 20k and the 65k recognition systems use 7000 decision-tree based gender independent across-word triphone states plus one state for silence. The system employs Gaussian mixture distributions with a total of 412k densities and one globally pooled diagonal variance matrix. 16 cepstral features together with their first derivatives and the second derivative of the energy are used. Each three consecutive observation vectors are concatenated and projected onto a 32 dimensional feature vector via an LDA. The ML trained recognizer obtaines a WER of 11.43% for the 20k system and 9.24% for the 65k system on the evaluation corpus (cf. Table 3).

In all discriminative experiments, the ML trained system was used to generate word-conditioned word lattices. The resulting word graph densities are shown in Table 4. Note that the graph error is zero because the hypotheses of the spoken word sequence are merged into the word lattices. This is done to keep the results comparable with those of other training criteria such as *Minimum Classification Error* (MCE). On average a lattice contains 6×10^9 distinct sentence hypotheses (different pronunciation variants and different alignments of the same word sequence are counted only once). The calculation of the word errors, including the integration of the word errors into the raw lattices takes 2-3 real time (2 GHz pentium machine). The memory requirements for this postprocessing step are a few hundred MB. The lattice density increases modestly by 13%, see Table 4.

Table 4. *Word graph densities for training lattices, "with error" refers to the lattices after incorporating the word errors into the "raw" lattices.*

corpus	WSJ0+WSJ1	
	raw	with error
avg. #arcs per spoken word (WGD)	59	67
avg. #arcs per timeframe	31	35

For all iterations of the discriminative training, the hypotheses encoded in the word lattices were re-aligned within their boundary times (the Viterbi segmentation points) as determined in the initial recognition phase. The convergence behaviour of the exact and the approximate approach is similar. All results reported here for discriminative training correspond to the word error rate obtained after five iterations, which yields the best results on the development corpus.

To assess the quality of the approximate accuracy (3) in the context of expected word error minimisation, the approximate accuracies are replaced with the exact accuracies and the system is re-trained. The results from this experiment are summarised in Table 3. In conclusion, the performance of the exact and the approximate approach on this corpus is the same.

5. CONCLUSIONS

In this paper, we presented a transducer-based algorithm to efficiently calculate the exact word error of all word sequence hypotheses represented by a word lattice. This algorithm was then used to augment the word lattices with the *exact* word errors. This allowed us to carry out the minimisation of the expected word error without approximations and using word lattices in all phases of the discriminative training. Experiments conducted on the NAB corpus have shown that the difference in recognition performance of this exact approach (exactMWE) and the approach using an approximation of the word errors (MWE) is not significant. However, tests are planned to verify if the proposed exact solution for expected word error minimisation improves the recognition performance on more complex corpora.

Acknowledgements This work was partly funded by the European Union under the Integrated Project "TC-STAR - Technology and Corpora for Speech to Speech Translation" (IST-2002-FP6-506738, http://www.tc-star.org).

6. REFERENCES

[1] P. C. Woodland and D. Povey, "Large Scale Discriminative Training of Hidden Markov Models for Speech Recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 25–48, 2002.

[2] W. Macherey, L. Haferkamp, R. Schlüter, and H. Ney, "Investigations on Error Minimizing Training Criteria for Dis-

Table 3. *Error rates [%] on the North American Business (*NAB*) corpus for the approximate (MWE) and the exact (exactMWE) approach.*

corpus	NAB-20k						NAB-65k					
	dev			eval			dev			eval		
	del	ins	WER	del	ins	WER	del	ins	WER	del	ins	WER
ML	1.41	2.02	11.36	1.57	2.12	11.43	1.66	1.16	9.14	1.56	1.21	9.24
MWE	1.31	2.03	11.17	1.38	2.02	10.83	1.53	1.19	8.85	1.46	1.22	8.88
exactMWE	1.30	1.97	11.10	1.39	2.04	10.90	1.53	1.18	8.85	1.49	1.23	8.99

criminative Training in Automatic Speech Recognition," in *European Conference on Speech Communication and Technology (Interspeech)*, Lisbon, Portugal, Sept. 2005, pp. 2133–2136.

[3] D. Povey and P. C. Woodland, "Minimum Phone Error and I-Smoothing for Improved Discriminative Training," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, vol. 1, Orlando, FL, May 2002, pp. 105 – 108.

[4] D. Povey, "Discriminative Training for Large Vocabulary Speech Recognition," Ph.D. dissertation, Cambridge, England, 2004.

[5] J. Kaiser, B. Horvat, and Z. Kacic, "A Novel Loss Function for the Overall Risk Criterion Based Discriminative Training of HMM Models," in *Proc. Int. Conf. on Spoken Language Processing*, vol. 2, Bejing, China, Oct. 2000, pp. 887 – 890.

[6] V. Doumpiotis and W. Byrne, "Pinched Lattice Minimum Bayes Risk Discriminative Training for Large Vocabulary Continuous Speech Recognition," in *Proc. Int. Conf. on Spoken Language Processing*, Jeju Island, Korea, Oct. 2004, pp. 1717 – 1720.

[7] M. Mohri, "Edit-Distance of Weighted Automata: General Definitions and Algorithms," *International Journal of Foundations of Computer Science*, vol. 14, no. 6, pp. 957 – 982, 2003.

[8] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics - Doklay*, vol. 10, no. 10, pp. 707 – 710, 1966.

[9] S. Kanthak and H. Ney, "FSA: An Efficient and Flexible C++ Toolkit for Finite State Automata Using On-Demand Computation," in *Proc. of the Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, July 2004, pp. 510 – 517.

[10] E. S. Ristad and P. N. Yianilos, "Learning String Edit Distance," *IEEE Transactions PAMI*, vol. 20, no. 5, pp. 522 – 532, 1998.

[11] C. Allauzen and M. Mohri, "Efficient Algorithms for Testing the Twins Property," *Journal of Automata, Languages and Combinatorics*, vol. 8, no. 2, 2003.

[12] M. P. Schützenberger, "Sur une variante des fonctions séquentielles," *Theoretical Computer Science*, vol. 4, no. 1, pp. 47 – 57, 1977.

[13] D. S. Pallett, J. G. Fiscus, W. M. Fisher, and J. S. Garofolo, "Benchmark Tests for the DARPA Spoken Language Program," in *ARPA Human Language Technology Workshop*, Princeton, NJ, March 1995, pp. 7–18.