Optimization on Decoding Graphs by Discriminative Training

Shiuan-Sung LIN, *François YVON*

GET/ENST and CNRS/LTCI, UMR 5141

lin@tsi.enst.fr, yvon@enst.fr

10.21437/Interspeech.2007-487

Abstract

The three main knowledge sources used in the automatic speech recognition (ASR), namely the acoustic models, a dictionary and a language model, are usually designed and optimized in isolation. Our previous work [1] proposed a methodology for jointly tuning these parameters, based on the integration of the resources as a finite-state graph, whose transition weights are trained discriminatively. This paper extends the training framework to a large vocabulary task, the automatic transcription of French broadcast news. We propose several fast decoding techniques to make the training practical. Experiments show that a reduction of 1% absolute of word error rate (WER) can be obtained. We conclude the paper with an appraisal of the potential of this approach on large vocabulary ASR tasks.

Index Terms: discriminative training, decoding graph, weighted finite-state transducer

1. Introduction

Most of the ASR research effort focuses on improving the performance of one specific component of the system, with the hope that it will improve the overall performance. This approach, does not take into account the dependency between the various knowledge sources. For instance, the design and estimation of acoustic models critically depends on the word pronunciation(s) that actually occur in the dictionary: a small dictionary might bear with simple models, while a large vocabulary system will require complex ones. Language modeling is performed separately from other resources using different, and often much larger, corpora. In addition, most modeling approaches perform parameter estimation separately for each resource, assuming that all the other parameters are fixed. The interdependency among knowledge sources is thus ignored, yielding under optimal performance.

Reliable estimation procedures for the various model parameters remain therefore a key issue for obtaining good performances. In the literature, the most commonly used estimation strategy is maximum-likelihood estimation (MLE). This approach attempts to estimate the parameters such that the likelihood of the training data is maximized. The principles of MLE rely on the availability of large training samples; however, the improvement in training does not always translates into better decoding performance [2]. This observation has led researchers to explore other estimation techniques, notably such as discriminative training (DT) techniques. In contrast to MLE, DT aims at optimizing the separation between good and bad hypotheses on the training samples. It is performed by formulating an objective function that, in some ways, penalizes parameters that are liable to confuse correct and incorrect words.

In the past years, various DT criteria such as maximum mutual information (MMI) [3] and minimum phone error (MPE) [4] have greatly improved the estimation of acoustic models. Discriminative techniques also have been shown to yield significant improvements in language modeling, such as minimum classification error (MCE) [2], minimum sample risk (MSR) [5], and reranking techniques based on the perceptron algorithm [6].

Recent advances in ASR systems have also promoted the use of weighed finite-state transducers (WFST), as a common underlying formalism for representing homogeneously the various knowledge sources [7]. A WFST is a type of finite state machine, whose state transitions carry input symbols, output symbols and arbitrary weights. A transition sequence from the initial state to the final state of a WFST represents a weighted mapping from inputs to the corresponding outputs. Using WFSTs to represent the various ASR components, different resources can be easily integrated in a single graph. Various optimization algorithms, such as determinization[1] and minimization, can be applied off-line, prior to decoding. These optimization techniques eliminate redundant arcs and states to yield an equivalent but more efficient graph.

Our previous work [1] proposed to combine these two techniques into a unified training framework: WFST are used to combine various sources into a finite-state graph, whose parameters are trained using DT. Our positive results on a simple name recognition task were recently confirmed in [8], which reports significant error rate reduction on a large vocabulary application.

In this paper, we extend our previous work to a much more complex large-vocabulary task, and analyze the achieved performance from several different perspectives. We detail several decoding techniques that had to be introduced in order to make training practical. In addition, we also investigate the strengths and shortcomings of this approach and discuss the new directions it opens.

2. Discriminative training on decoding graphs

2.1. MCE criterion

We follow here the notations of our previous work [1]. Assume G is an integrated finite- state graph. G contains two kinds of parameters: state transition weights and acoustic model parameters, which are associated with HMM states Gaussian densities. Given a word string W, a set of acoustic model Λ, a set of transition weights Γ and an observation sequence X, the conditional log-likelihood of X is approximated as the score of the best path in G for input X

[1] Determinization of finite-state transducers is not a well-defined notion [7]: in conformance with the literature, we will use the term here in a rather loose sense to denote exact and heuristic techniques aiming at removing duplicate paths on the input type of the transducer.

and output W. This score combines the individual acoustic log-likelihoods and transition weights along the decoded path:

$$g(X,W,\Lambda,\Gamma) = a(X,W,\Lambda) + b(W,\Gamma) \quad (1)$$

where $a(X,W,\Lambda)$ is the sum of acoustic log-likelihood and $b(W,\Gamma)$ is the sum of transition weights along the path from the starting state to the final state. Speech decoding consists of finding a word hypothesis W_{hyp} which maximizes g over all possible word sequences. If W_{ref} is the correct word sequence, the performance of the recognizer can be expressed as a function of the score difference between the reference and the best hypothesis. For a given input vector, the misclassification function is defined as:

$$d(X,\Lambda,\Gamma) = -g(X,W_{ref},\Lambda,\Gamma) + g(X,W_{hyp},\Lambda,\Gamma) \quad (2)$$

An erroneous recognition hypothesis simply translates into a positive value of $d(X,\Lambda,\Gamma)$, meaning that the correct word sequence is not the top ranking one according to g. The next step is to define a continuously differentiable loss function, $l(X,\Lambda,\Gamma)$, integrating the misclassification measure $d(X,\Lambda,\Gamma)$:

$$l(d(X,\Lambda,\Gamma)) = \frac{1}{1+\exp(-\gamma d(X,\Lambda,\Gamma)+\theta)} \quad (3)$$

where γ and θ are the parameters which control respectively the slope and the shift factor of the sigmoid function. Using generalized probabilistic descent (GPD) algorithm, a standard iterative procedure can be defined, based on the following parameter update rule for the transition weights:

$$\Gamma_{t+1} = \Gamma_t - \varepsilon \nabla l(X,\Lambda,\Gamma_t) \quad (4)$$

Assuming that the acoustic model parameters are fixed, the loss function needs to be differentiated only with respect to the transition weights. The derivation of (4) yields:

$$\nabla l_i(X,\Lambda,\Gamma_t) = \frac{\partial l_i}{\partial d_i}\frac{\partial d_i(X,\Lambda,\Gamma_t)}{\partial \Gamma} \quad (5)$$

Viewing Γ as a transition weight vector and taking partial derivatives of $d_i(X,\Lambda,\Gamma)$ with respect to Γ, the derivation of (5) finally yields:

$$\frac{\partial l_i}{\partial d_i} = \gamma l(d_i)(1-l(d_i)) \quad (6)$$

$$\frac{\partial d_i(X,\Lambda,\Gamma_t)}{\partial \Gamma} = -I(W_{ref},s) + I(W_{hyp},s) \quad (7)$$

where $I(W,s)$ represents the number of transition weight s on the best decoding path for W.

2.2. Parameter update rule

In MCE training, the parameters that have to be carefully selected are γ and ε, which respectively represent the slope of sigmoid function, that transfers a misclassification measure to the 0-1 domain, and a scale factor to update the parameters. Also recall that training is performed on a weighted finite-state graph, where each word is represented by a sequence of phones. A first requirement is thus to recover, through alignment, the complete (i.e. phone level) best path for the reference. In addition, implementing the update rule (7) requires to address two issues: 1) the choice of n-gram terms from the hypothesis and the reference word strings and 2) the arc position for transition weight update.

Since a correctly decoded word may follow an incorrect word, meaning that parameter update is based on an incorrect word history, we focus on the relationship between two consecutive words rather than considering a certain word history. The choice of the updated arc has a great influence on the overall transition weight distribution. In this implementation, the arc for parameter update is randomly chosen by random sampling with uniform probability amongst all possible candidates.

In addition, the value of γ is selected by relating the effect on the parameter adjustment to the score difference. Our experiments suggest that γ=0.02 is a suitable choice. The learning rate ε is determined dynamically for each training sample, using the line-search technique, so as to give an effective parameter update while keeping stable convergence.

2.3. Implementation

The decoding graph is built by compiling knowledge sources from typical ASR systems, namely the dictionary, the acoustic models and the language model, as weighted finite-state transducers (WFST). Using WFST techniques, the composition allows different level's representations (from a word to pronunciation sequence(s) and to associated HMMs) to be combined in a single graph. The determinization is used to reduce the graph size by eliminating the redundant paths, yielding an equivalent, yet more efficient graph. An optional silence (short pause model) is added at word boundaries; finally, a word insertion penalty (WIP) is added to balance the length of output word sequences.

Our decoder performs the search by using the general token passing [9] over a finite-state network. To speed up token propagation over ε-transitions, we use a look-up table to store the ε-closure relationship between states. This table is built off-line by traversing the graph and recording ε paths. Using a look-up table, a token is easily re-directed to its destination states without having to re-search the graph.

Alignment is different from decoding: the search procedure must find in the graph all the path which output a given word string W. Conceptually, this amounts to intersecting the output language of the WFST with W: our approach thus uses the general principle of graph inversion, which exchanges the input and output symbols on every transitions. By searching for W in this inverted graph, the relevant phonetic sequences can be computed in advance. The extracted graph generally consists of hundreds of states and arcs, no matter how large the original graph is.

Weights in the WFST can be distributed in many ways, which still result in an equivalent WFST. In typical ASR systems, it has been demonstrated that if the probability is properly redistributed, both the pruning efficiency and the search speed can be improved [10]. Our weight pushing algorithm is conceptually similar to the "tropical semiring" algorithm introduced in [7] and proceeds as follows: the graph is first reversed (i.e. all transitions are reversed); for all the states, the maximum transition weight among the incoming arcs is pushed towards the word boundary; finally the graph is reversed again. Some caution is taken to prevent weights from becoming too small, which would penalize too strongly the corresponding transition. This algorithm has a time complexity comparable with the algorithm of [10] but is more space efficient, as it is linear in the number of states.

3. Experiments

3.1. Experimental setup

Our experiments are carried out on a French radio broadcast news database ESTER [11]. Approximately 62.88 hours of manually transcribed data (TRAINSet) was used for training the parameters of acoustic models and a 3-gram language model (LM). The development set (DEVSet) and test set (TESTSet) respectively contains 5.3 and 2.95 hours of audio. Transcriptions containing out-of-vocabulary (OOV) words were removed from TRAINSet and DEVSet.

Our original HMM set contains 21466 acoustic models. Additionally, 1369 models are synthesized according to the decision tree by state-tying. Each model consists of 3 states, except the short pause model (one-state model) used to model the short inter-word silence. There are a total of 6238 distinct states, each of which is associated with a 39-dimensional probability density function taking the form of a mixture of 32 Gaussians, assuming a diagonal covariance.

Two 3-gram LMs are used to construct the decoding graphs: one is the above-mentioned 3-gram model; the other is a much larger 3-gram obtained by linear interpolation of several models trained on archives of the newspaper *LeMonde*, covering approximately 400M words. Both LMs contain 65k words of vocabulary. The resulting graphs respectively contain 824,845 states for 1,655,723 arcs and 6,997,044 states for 19,676,349 arcs. The so-called "fudge" factor α is used to balance the acoustic and language model scores in the log domain. Using empirically determined values for α, γ and WIP, the baseline word error rate for each decoding graph is shown in the following table:

LM	2-gram	3-gram	perplexity	Baseline
Graph1	248739	102461	158.92	45.9
Graph2	5052090	4033834	86.42	37.9

Table 1. *The number of n-grams, perplexity and baseline WER of individual decoding graphs.*

The decoder used in our experiments runs respectively at 0.7×RT and at 3×RT on Graph1 and Graph2 on a 3.6Ghz Xeon processor[1]. The alignment procedure runs in 0.05×RT. Faster decoders have been reported in the literature (eg. [12]), but these decoders only need to keep track of the word history of hypotheses. In these cases, crucial information for DT, such as the complete state transition sequence, cannot be recovered. Our decoder has to store the full state history of the top 20k candidate hypotheses at each time frame, yielding an extra-layer of book-keeping activity: this explains why we could not achieve real-time decoding speed on very large graphs.

3.2. Results

3.2.1. Deterministic versus random update

Consecutive words extracted from the reference and hypothesis word strings are represented as sequences of phones, and the MCE criterion does not precisely prescribe the exact transition weight to update. This leaves us with several options: 1) update all the weights along the transition paths, or 2) select (deterministically or randomly) one arc to update. 3) for paths which contain a back-off transition, an option would be to leave the weight unchanged when an update would yield a higher back-off probability than the *n*-gram probability.

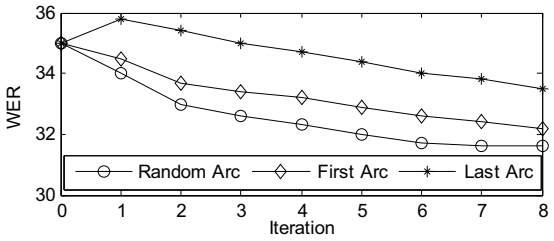

Figure 1: *Performance of different updating schemes.*

Our implementation treats back-off transitions as regular ones, and considers only one arc from the transition path for parameter adjustment. Experiments are performed using Graph1, starting from the baseline (WER=35) on the DEVSet. Two deterministic updating schemes are compared: update on the first arc and update on the last arc (see Figure 1). The random update scheme not only converges more quickly but also achieves a lower WER, confirming our previous findings [1]. This strategy is used in all the following experiments.

3.2.2. Error rate reduction on a large graph

A decoding graph contains a large number of parameters. Some of the parameters, notably the transition weights, are not entirely independent. This makes the training process difficult to reach the minimum error rate. However, a large graph usually provides a better baseline for DT. Therefore, given the same amount of training data, we compare two graphs of different sizes: DT is performed using DEVSet on Graph1 and Graph2. TESTSet is used to evaluate the performance of decoding graph after parameter adjustment. After 8 training iterations, we get the results displayed in Table 2.

	DEVSet	TESTSet
Graph1	35→31.6 (-3.4)	45.9→44.9 (-1)
Graph2	28.5→25.1 (-3.4)	37.9→37 (-0.9)

Table 2. *WER reduction on the development set and on the test set, using graphs of different size.*

The improvement on the DEVSet is similar for both graphs, with Graph2 starting from a much better baseline. On the TESTSet, WER reduction is smaller (:1% absolute), half of which is obtained after the 1st iteration itself.

In practical applications, a graph may be constructed from a language model containing a large number of *n*-grams, while the number of word pairs in the development set, which is used to optimize the graph, is comparatively much smaller. This means that DT updates only a small portion of the transition weights (no matter the number of iterations). In the next section, we use a larger training set for DT, so as to better evaluate the potential of this approach.

3.2.3. Training with a larger data set

These experiments are carried out using Graph2. One single training iteration is performed using the entire 62 hours of

[1] These figures are obtained using pre-computed acoustic likelihoods. We estimate that the on-line computations of likelihoods would increase the run-time by less than 0.7×RT for decoding and by about 0.1×RT for alignment.

TRAINSet for DT. This dataset contains approximately 11 times more word pairs, out of which 7 times more *different* word pairs than DEVSet. Results are presented in Table 3. With this larger dataset, decoding results on the test set give a total of 1.1% absolute WER reduction *in the 1st iteration*, approximately 3 times more than that obtained using DEVSet. Based on our experiments, it is likely that running additional training iterations with the large graph could bring us another 1% absolute WER reduction on the test set.

	Sub.	Del.	Ins.	WER
DEVSet	23.1	12.0	2.4	37.5
TRAINSet	22.8	11.2	2.8	36.8

Table 3. *WER reduction using different training data.*

4. Discussion and Improvements

Our experiments have shown that DT has the potential of effectively reducing the WER, even for large-vocabulary tasks. A first worry is the stability of our iterative optimization procedure: a close examination of the training log reveals that the number of updates is gradually reduced on both graphs; yet, even after 8 training iterations, the number of parameter updates performed during one iteration remain substantial (in the tens of thousands), suggesting that some transition weights do not completely stabilize.

We have explained that DT reduces the training errors by re-distributing the transition weights, yielding similar output strings to the reference, and smaller score differences between the reference and the best hypothesis. This is confirmed by the analysis of the score difference between the reference and the best hypothesis of the DEVSet: after 6 training iterations, the score of the reference has increased for almost all the training samples; and 217 originally erroneous training samples (3.34% of the training sentences) are completely corrected (reference and hypothesis are fully identical).

Results on the test set are positive, albeit by a smaller margin, suggesting that our algorithm is not generalizing well. In fact, a study of the word pairs that occur both in the test and in the training set reveals that about 40% of the word pairs in the former also occur in the latter. The situation is in fact much worse: even if an update is performed on a transition u→v during training, it might still not be useful during testing due to the difference in language model history (e.g. *wuv* appears in the training set and *w'uv* in the test set). Finally, only about 2.5% of the total number of word pairs in the test set are updated during training, clearly demonstrating the fact that our parameter update procedure can only provide with a limited improvement on this dataset.

Throwing in more training data is an effective, albeit computationally demanding, remedy to this situation. The improvements to our training procedure that are really needed should make the parameter updates less local: this is in fact what happens with traditional LM estimation procedures: any new instance of *uvw* will increase the likelihood of *w* in the context of *uw*; it will also increase, by a smaller margin, the likelihood of *w* following *v*. Two ways to make parameter update less "local" are being explored: one is to consider the top-N hypotheses rather than just the best one (see [13]); the second is to consider updating arcs *on every possible alignment* of the reference, rather that just the best one.

5. Conclusion and perspectives

DT on large decoding graph has been shown to be both computationally tractable and effective. In this training framework, parameter optimization is performed on a static decoding graph, whose transition weights are iteratively adjusted. We have presented the results of experiments on large vocabulary tasks which confirm the findings of [8] and discussed the strength and shortcomings of our DT procedure. Two other improvements are foreseen: 1) to get a fine-grained control of the effect of parameter updates. On a compact WFST, updating of one parameter changes the score of all the paths that use this transition, which can prove detrimental to the overall performance. 2) to create (and update) new transitions as needed, so as to minimize this effect; the downside being an increase of non-determinism in the graph.

Finally, we have thus far consider the acoustic parameters to be fixed; for some training samples, though, the acoustic mismatch is so bad that updating the transition weights only cannot recover the reference, or would do so at the price of unreasonable weight changes. Better ways to interleave acoustic and linguistic training are definitely needed to accommodate this kind of situation.

6. References

[1] S.S. Lin and F. Yvon, "Discriminative Training of Finite State Decoding Graphs," Proc. InterSpeech, pp. 733-736, 2005.

[2] Z. Chen, M.J. Li and K.F. Lee, "Discriminative Training on Language Model," Proc. ICSLP, 2000.

[3] L. R. Bahl, F. Jelinek and R. L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", IEEE Trans. on Pattern Analysis and Machine Intelligence., vol(5), pp. 179-190, 1983.

[4] D. Povey and P. C. Woodland, "Minimum Phone Error and I-Smoothing for Improved Discriminative Training," Proc. ICASSP, pp.105-108, 2002.

[5] J. Gao, H. Yu, W. Yuan and P. Xu, "Minimum Sample Risk Methods for Language Modeling," Proc. HLT/EMNLP, 2005.

[6] B. Roark, M. Saraclar and M. Collins, "Corrective Language Modeling for Large Vocabulary ASR with the Perceptron Algorithm," Proc. ICASSP, 2004.

[7] M.Mohri, "Finite-State Transducers in Language and Speech Processing," Computational Linguistics, 23:2, pp. 269-311, 1997.

[8] H. K. Jeff Kuo, B. Kingsbury and G. Zweig, "Discriminative Training of Decoding Graphs for Large Vocabulary Continuous Speech Recognition," Proc. ICASSP 2007.

[9] S.J. Young, N.H. Russel, and J.H.S. Thornton, "Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems," Technical Report, Cambridge University: 1989.

[10] M. Mohri and M. Riley, "A Weight Pushing Algorithm for Large Vocabulary Speech Recognition," Proc. EuroSpeech, pp. 1603-1606, 2001.

[11] G. Gravier, J.-F. Bonastre, S. Galliano, E. Geoffrois, K. McTait and K. Choukri. "The ESTER evaluation campaign of Rich Transcription of French Broadcast News", Proc. LREC, 2004

[12] G.Saon, D.Povey, and G.Zweig, "Anatomy of an Extremely Fast LVCSR Decoder," Proc. InterSpeech, pp. 549-552, Lisbon, 2005.

[13] H.-K. Jeff Kuo, E. Fosler-Lussier, H. Jiang and C.-H. Lee, ``Discriminative Training of Language Models for Speech Recognition", Proc. ICASSP'02, Orlando, Florida, 2002.