Discriminative Training of Finite State Decoding Graphs

Shiuan-Sung LIN, François YVON

GET/ENST and CNRS/LTCI, UMR 5141

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

Abstract

Automatic Speech Recognition systems integrate three main knowledge sources: acoustic models, pronunciation dictionary and language models. In contrast to common practices, where each source is optimized independently, then combined in a finite-state search space, we investigate here a training procedure which attempts to adjust (some of) the parameters after, rather than before, combination. To this end, we adapted a discriminative training procedure originally devised for language models to the more general case of arbitrary finite-state graphs. Preliminary experiments performed on a simple name recognition task demonstrate the potential of this approach and suggest possible improvements.

1. Introduction

Large-vocabulary automatic speech recognition (ASR) systems integrate three main resources: a set of acoustic models, a pronunciation dictionary, and a statistical language model. Acoustic models, usually based on the Hidden Markov Model (HMM) formalism, match a stream of acoustic parameters and predefined statistical models of acoustic units. The dictionary contains a deterministic or probabilistic mapping between sequences of acoustic units and words. The language model defines a probability distribution over word sequences, which encodes the most common local syntactic regularities. A major breakthrough has been the development of a methodology for combining these resources in a unified framework, based on the formalism of Weighted Finite-State Transducers (WFSTs, see e.g. [1] and the reference therein). In this formalism, decoding is performed using standard heuristic search procedures in an optimized finite-state transducer, yielding fast and accurate decoders.

One issue remains unsettled though, which relates to the specification of these knowledge sources. Each resource involves the estimation of myriad of parameters, which, according to common practices, are estimated in isolation, using separate training corpora. This means, for instance, that the definition of dictionary entries is performed independently from the acoustic modeling and the same goes for language models. As a result, many “small” decisions made during specification of these sources, regarding, for instance, the number and topology of HMMs, the modeling of pronunciation variants and the complexity of the language model, have to be experimentally validated, yielding a significant tuning overhead every time a system has to be set up. Another downside of this approach is that it yields unnecessarily large graphs, which are yet difficult to prune. We contend that a better integration of resources is needed during the model estimation step and propose a discriminative training methodology to achieve this goal.

Discriminative training (DT) is a general estimation technique which aims at setting model parameters so as to directly optimize the performance of a model or a function thereof, rather than the likelihood of training data. The availability of very-fast (sub real-time) decoders makes this learning strategy feasible. In the context of ASR, this methodology has been successfully applied to the estimation of acoustic [2,3] and linguistic models [3,4,5].

In this paper, we try to take the idea of DT one step further, and apply this methodology to adjust the various parameters of **the integrated decoding graph**. By working directly on a representation that includes all the knowledge sources, we expect to eventually come up with a set of parameters which will play its role better, i.e. improve the efficiency of the search procedure. Another benefit of this approach is to build search graphs which are beyond the reach of the standard combination procedure: for instance graphs where the probability of pronunciation variants depends on the language model history.

The integrated decoding graph contains all the resource parameters, and many more[1]: trying to simultaneously optimize all of them may well prove infeasible. In this paper, we thus make the simplifying assumption that acoustic model parameters are fixed and we only attempt to optimize the arc transition weights adjusting the graph to increase the discrimination capacity of the network in areas where the acoustic confusability is high. Starting with an initial graph configuration, our algorithm iteratively updates the graph parameter so as to increase the recognition rate, until convergence is reached.

This paper is organized as follows: we first introduce our discriminative training procedure, based on the minimum classification error (MCE) criterion and detail our own implementation. We then describe the task used to test this methodology and report experimental results. We finally discuss some open issues and draw perspectives for future work.

2. Discriminative training

2.1. The MCE model

In the MCE approach, the estimation of the model parameters aims at optimizing a function of the classification error rate [2]. Since the resulting optimization program does not lend itself to an analytical resolution, estimation is performed through an iterative parameter optimization procedure. In this section, we present this model, which extends the ideas originally introduced in [5], and from which we borrow the notations.

We assume that G is an integrated finite-state decoding graph, devised according to the procedures introduced in [1]. G contains two kinds of parameters: acoustic model parameters, associated with HMMs states Gaussian densities, and state transitions weights.

Given a word string W, a set of acoustic models Λ, a set of transition weights Γ and an observation sequence X, the

[1] The language model weight, the word insertion penalty…

conditional likelihood of X is approximated as the score of the best path in G for input X and output W. This score combines the individual acoustic likelihoods and transition weights according to:

$$g(X,W,\Lambda,\Gamma) = a(X,W,\Lambda) + b(W,\Gamma) \tag{1}$$

where *a(X,W,Γ)* is a sum of acoustic likelihoods and *b(W,Γ)* is a sum of transition weights. Speech decoding consists in finding the word sequence W_1 maximizing *g* over all possible word sequences.

If W_0 is the known correct word sequence, the performance of the recognizer can thus be expressed as a function of the difference between the score of the correct sequence and that of the best hypothesis. For a given input vector, we thus define the misclassification function as:

$$d_f(X,\Lambda,\Gamma) = -g(X,W_0,\Lambda,\Gamma) + g(X,W_1,\Lambda,\Gamma) \tag{2}$$

An erroneous recognition hypothesis thus simply translates into a strictly positive value for d_f, meaning that the correct word sequence is not the top ranking one according to *g*. To formulate an optimization procedure based on the misclassification error function l_f, the differentiable class loss function is introduced as:

$$l_f(X,\Lambda,\Gamma) = l(d_f(X,\Lambda,\Gamma)) = \frac{1}{1+\exp(-\gamma d_f(X,\Lambda,\Gamma)+\theta)}$$

where γ and θ are parameters which control respectively the slope and the shift factor of the sigmoid function. A standard iterative gradient procedure can then be defined, based on the following update rule for the set of transition weights:

$$\Gamma_{t+1} = \Gamma_i - \varepsilon.\nabla l_f(X,\Lambda,\Gamma_t) \tag{4}$$

If we consider that acoustic model parameters are fixed, the loss function needs only be differentiated with respect to the weight transition probabilities. Given that *b(W,Γ)* is a mere sum of transition weights, the mathematical derivation exactly follows that of [5], yielding:

$$\nabla l_f(X,\Lambda,\Gamma_t) = \frac{\partial l_f}{\partial d_f}\frac{\partial d_f(X,\Lambda,\Gamma_t)}{\partial \Gamma} \tag{5}$$

where $\frac{\partial l_f}{\partial d_f} = \gamma l(d_f(X))(1-l(d_f(X))$.

We continue to work out the mathematics by taking partial derivatives of the term $d_f(X,W,\Gamma)$ with respect to the transition weight vector Γ, finally yielding:

$$\frac{\partial d_f(X,\Lambda,\Gamma_t)}{\partial \Gamma} = -I(W_0,s) + I(W_1,s) \tag{6}$$

where *I(W,s)* represents the number of occurrences of the transition weight *s* on the best decoding path for *W*. This procedure can further be generalized by considering the N-best hypotheses, rather than the single best one (again, refer to [5] for the details).

Altogether, the training procedure consists in iteratively scanning the training corpus until convergence, using for each training sentence the update rule in (5).

2.2. Implementation

The training data only contains the correct output word sequence: the corresponding reference path is computed using a forced alignment procedure. If a transition is more frequent in the reference path than in the hypothesis path, its weight is increased; otherwise, it has to be decreased. For transitions which exist with the same frequency in the reference and in the hypothesized path, no update is performed.

Based on this general principle, many training regimes can be considered: in the experiments reported above, we made the following choices: (i) in all cases, the HMM internal transitions are frozen and are not updated: this allowed us to limit the graph expansion at the phone level; (ii) amongst the remaining transitions, we only update the ones whose output label is not empty. Two different update strategies were considered: in the first one, we do not update transitions which appear (albeit with different counts) in the reference and hypothesis path. In the second one, we update all the transitions having a different frequency, according to the update rule of equation (6).

All the experiments reported above were performed using our own decoder: a full search (involving no pruning) runs in about 0.8RT on a 3.0 GHz Pentium IV. The decoding graphs were prepared using the FSM Toolkit [7].

3. Experiments

The preliminary experiments reported hereafter were carried out on a simple name recognition task. Our main purpose was to get a better grasp of the behavior of the training procedure and of the influence of the various parameters on a well-understood task, using a relatively simple decoding graph. We first present the task and the database, before reporting the results of these experiments.

3.1. Task and database

The task consists in recognizing isolated sequences consisting of a proper name followed by its spelling, as illustrated by:

frana F-R-A-N-A

The recognizer's output is accordingly composed of two parts: a sequence of phone labels, followed by a sequence of letter labels. No dictionary look-up is performed to match the output with existing names; performance (WER) is simply computed as a function of the number of corrected recognized symbols (phones and letters) in the output. The main motivation for experimenting with this small vocabulary tasks was (i) to assess the influence of the various parameters involved in the procedure and (ii) to be able to analyze the output decoding graph and get a better understanding of the benefits of this training procedure.

Data for this task was extracted from the 'spelled word' category of the Swiss-French Polyphone database [8]. After cleaning invalid[2] entries, the database was randomly split into a discriminative training set (8427 names) and a testing set (1150 names), representing respectively 14.16 and 1.95 hours of recordings. Performance measurement being based on a phonetic match between hypotheses and references, each orthographic name was automatically converted into a sequence of phones, using a pronunciation dictionary whenever applicable and automatic pronunciation procedures otherwise.

For these tasks, acoustic models of varying sizes were estimated using the 'phonetic-rich' category of Polyphone (totaling 49.79 hours of speech): for each of our 39+2 phonetic units, context-independent models containing from 16 to 64 Gaussian mixtures per HMM state were considered.

The initial decoding graph was set up as follows: two language models were separately trained, one for phone sequences and one for letter sequences. The former was trained using automatic phonetic transcript of the 'phonetic-rich' category items in Polyphone: this yielded a phone-based language model encoding a general distribution of possible

[2] Entries for which one subpart is missing or which contain non-conventional spelling directives.

phone sequences. In contrast, due to lack of data, the letter language model was built using the same dataset that is used for discriminative training, i.e. spellings extracted from the ‘spelled-name’ category: this model already integrates some knowledge regarding the typical sequences of letters that occur in names. Each of these models was trained conventionally, using Maximum Likelihood estimation and standard smoothing procedures. Table 1 gives a quantitative description of the phone and letter language models, including the total corpus size used for estimation, their number of parameters and perplexity (PP).

	Corpus	Bigram		Trigram	
		Size	PP	Size	PP
Phone	1,411,699	1237	29.69	16865	10.77
Letter	77,185	1078	13.33	5027	9.51

Table 1: Phone and letter language models

Each language model is then turned into a weighted finite state acceptor (WFSA). The letter WFSA is then composed with a FST mapping letters to their spelling. The resulting WFST is further determinized[3] and concatenated with the phone WFSA to produce a phone-based decoding graph. This construction is illustrated on Figure 1. The bigram graph contains 516 states and 2,198[4] arcs; for the trigram graph these numbers are respectively 8,154 and 32,467.

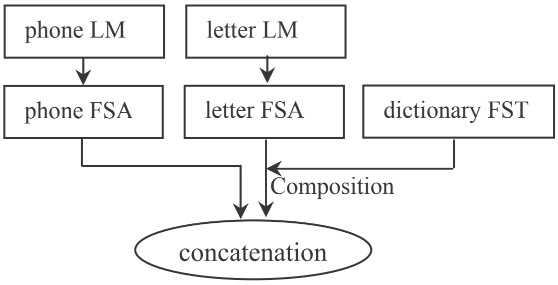

Figure 1: Baseline graph construction

3.2. Parameters selection

Before experimenting with the discriminative training procedure, we performed a number of experiments aiming at setting two parameters of the procedures: γ, which controls the slope of the sigmoid function, and ε, which is the increment parameter of the gradient descent. For the purpose of this study, we assumed that α, the language model weight, and the word insertion penalty, δ, are fixed, taking values α = 0.23, δ = 0.6 for the phone part and δ = 0.4 for the letter part. We also set θ=0.

The rationale for finding a reasonable parameter set is based on the following remarks. When the difference d_f, between reference and hypothesis scores is large, the loss function tends to one, and the corresponding update tends to zero (see equation (5). A first decision was made to set an upper bound on the value of this difference: sentences whose misclassification score exceed this threshold are not considered during training. Based on an analysis of the distribution of for d_f, in the training data, this upper bound was fixed so as reject only a small portion of the training data (about 3% of the sentences). Given the range of possible values for for d_f, we then choosed γ = 0.01 so as to control the value of the gradient. We finally set ε=10 to get a total increment factor of 0.1 for the gradient descent. With these parameters, the update factor for a transition lies between 0.011 and 0.025 on a log scale.

Choosing a smaller value for γ would have the effect of increasing the average update factor, yielding a faster convergence of the procedure, at the risk however of rendering the process unstable. While more experimental work is required to fine tune these parameters, the values used for our experiments seemed to yield a reasonable convergence rate.

3.3. Results

Baseline results are obtained with the 32-mixture acoustic models before starting the discriminative training procedure. After each training iteration, the graph is dumped on disk and used for testing. For each of our experiments, 5 training iterations were performed.

Figure 2 plots the evolution of recognition performance after each training iteration for the first training regime (see Section 2.2). Identical results were obtained with the second, more costly, training regime. Additional control experiments carried out with 16 and 64-mixture models also yielded a similar decrease pattern.

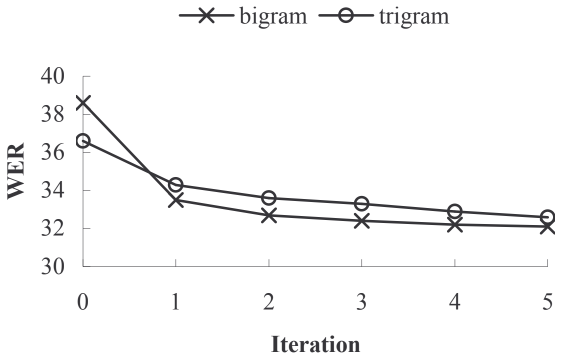

Figure 2: Evolution of the WER

For the bigram curve, most of the performance increase is achieved after one iteration, and the remaining iterations only bring a small improvement. Overall, the performance increases by 6.5 points for the bigram model, by 4 points for the trigram model. One iteration of training already brings a very significant increase (4% absolute for the bigram graph) in performance. The discriminatively trained bigram graph significantly outperforms the baseline trigram graph, even though it contains about ten times less parameters. This illustrates the uselessness of many parameters in the trigram baseline graph. After five iterations of training, the trigram graph is still lagging behind. This may be explained by the fact that this graph contains a larger number of parameters, resulting in a slower convergence rate: after 5 iterations, the performance continues to increase, albeit at a small pace.

Another perspective on the convergence of the algorithm is given by a closer examination of the update pace: for the bigram graph, after five iterations of training, the number of updates stops its decrease: the number of updates per iterations remains very high (about 120,000), suggesting that the same weights are repeatedly increased and decreased.

[3] Control experiments were ran without determinization and show no significant difference on this task.

[4] The fully expanded network would thus include about three times more states: the graph is here only expanded at the level of phones.

This stabilization is not observed for the trigram graph. Again, more experimental work is required to confirm these preliminary observations.

To investigate in more details the effect of discriminative training, we examined the 30 most frequent confusion pairs in the bigram baseline system. This tracking was performed independently on the phone part and on the letter part of the decoded utterances. Figure 3 and 4 display the evolution of the number of confusions for these pairs before and after training. As clearly appears on these figures, discriminative training manages to significantly reduce these frequently occurring confusions. For the phone part, only a handful of pairs show an increase in confusion. For most of the remaining ones, we get an improvement, which is all the more substantial as the related phones have a different distributional pattern: this is the case, for instance, of the pairs ɑ/a, a/t, i/t… In contrast, some pairs which are both acoustically and distributionally very similar remain difficult to discriminate: the top remaining errors in the phone part are: e/ɛ; o/ɔ; ɔ/o; b/d; t/p...

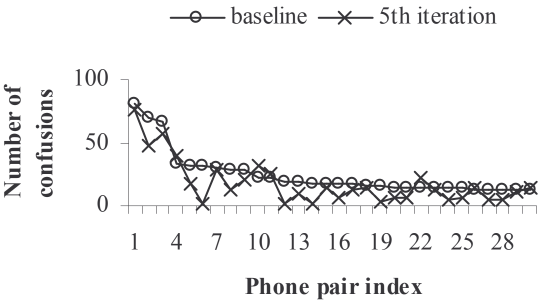

Figure 3: Evolution of the top 30 phone confusion pairs.

The same pattern of improvement is observed for spelled letters: some confusions are substantially reduced (e.g. between E (/ø/) and 2 (/dø/), or between A (/a/) and K (/ka/); some pairs nonetheless remain difficult to discriminate and continue to account for a large number of errors: B (/be/) vs. D (/de/), L (/ɛl/) vs. N (/ɛn/), M (/ɛm/) vs. N, S (/ɛs/) vs. F (/ɛf/)...

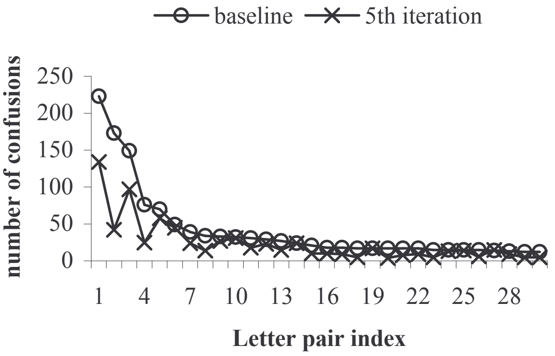

Figure 4: Evolution of the top 30 letter confusion pairs.

4. Conclusions

In this paper, we presented a discriminative training procedure aiming at adjusting the various parameters of a decoding graph so as to directly optimize the recognition performance. Results obtained on a simple name recognition task demonstrate the effectiveness of this methodology, illustrated by a 6.5% absolute improvement of the word error rate on a bigram graph. A close examination of the optimized graph reveals that discriminative training has the expected effect of facilitating, whenever possible, the discrimination between acoustically confusable phone and letter pairs. We are currently experimenting with alternative training regimes, such as considering all the available transitions for update, or taking the N-best recognition hypotheses into account. We are also trying to revise the graph update procedure so as to ensure that updates have a “local” impact: to see why this is important, consider the situation when the updated transition is a transition leaving a back-off state: its increase (or decrease) will impact the score of many paths which should not be changed. The idea is thus to introduce new states in the graph so as to guarantee that updates only affect the score of current reference and hypothesis paths. Additional experiments using richer acoustic models and larger vocabulary and language models are also required to assess more precisely the benefits of this new training strategy.

5. References

[1] Mehryar Mohri, Fernando C. Pereira and Michael Riley, ``Weighted Finite-State Transducers in Speech Recognition” *Computer Speech and Language, 16(1):69-88, 2002*.

[2] Biing-Hwang Juang, Wu Chou and Chin-Hui Lee, ``Minimum Classification Error Rate Methods for Speech Recognition”, *IEEE Transactions on Speech and Audio processing, 5:3, pp. 266-277, 1997*.

[3] Ralf Schluter, Wolfgang Macherey, Boris Muller and Herman Ney, ``Comparison of Discriminative Training Criteria and Optimization Methods for Speech Recognition”, *Speech Communication. Vol. 34, pp. 287-310, 2001*.

[4] Zheng Chen, Mingjing Li and Kai-Fu Lee, ``Discriminative Training on Language Model", *Proc. ICSLP’00, Beijing, China, 2000*.

[5] Hong-Kwang Jeff Kuo, Eric Fosler-Lussier and Hui Jiang, Chin-Hui Lee, ``Discriminative Training of Language Models for Speech Recognition”, *Proc. ICASSP’02, Orlando, Florida, 2002*.

[6] Brian Roark, Murat Saraclar and Michael Collins, ``Corrective Language Modeling For Large Vocabulary ASR With The Perceptron Algorithm”, *Proc. ICASSP 2004. Montreal, Canada, 2004*.

[7] Mehryar Mohri, Fernando C. Pereira and Michael Riley, ``General-purpose Finite-State Machines Software Tools”. http://www.research.att.com/sw/tools/fsm, AT&T research, 1997.

[8] Jean-Luc Cochard, Gérard Chollet, Philippe Langlais and Andrei Constantinescu. “Swiss-French Polyphone: a Telephone Speech Database to develop Interactive Voice Servers”, *Linguistic Databases*, CSLI Publications, John Nerbonne (Ed.), 1997.