HILL CLIMBING ON SPEECH LATTICES: A NEW RESCORING FRAMEWORK

Ariya Rastrow[1], Markus Dreyer[1], Abhinav Sethy[2], Sanjeev Khudanpur[1],
Bhuvana Ramabhadran[2] and Mark Dredze[1]

[1]Human Language Technology Center of Excellence, and
Center for Language and Speech Processing, Johns Hopkins University
[2]IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
{ariya, markus, khudanpur, mdredze}@jhu.edu {asethy, bhuvana}@us.ibm.com

ABSTRACT

We describe a new approach for rescoring speech lattices — with long-span language models or wide-context acoustic models — that does not entail computationally intensive lattice expansion or limited rescoring of only an N-best list. We view the set of word-sequences in a lattice as a discrete space equipped with the edit-distance metric, and develop a hill climbing technique to start with, say, the 1-best hypothesis under the lattice-generating model(s) and iteratively search a local neighborhood for the highest-scoring hypothesis under the rescoring model(s); such neighborhoods are efficiently constructed via finite state techniques. We demonstrate empirically that to achieve the same reduction in error rate using a better estimated, higher order language model, our technique evaluates fewer utterance-length hypotheses than conventional N-best rescoring by *two orders of magnitude*. For the same number of hypotheses evaluated, our technique results in a significantly lower error rate.

Index Terms— Rescoring, Hill Climbing, Search Algorithm

1. SHORTCOMINGS OF N-BEST RESCORING

Due to the availability of large amounts of training data and computational resources, building more complex models with sentence level knowledge and longer dependencies has been an active area of research in automatic speech recognition (ASR) [1, 2, 3]. Yet, due to the complexity of the speech recognition task, integration of many of these complex and sophisticated knowledge sources into the first decoding pass is not feasible. Many of these long-span models cannot be represented as weighted finite-state automata (WFSA), making it difficult to incorporate them in even a lattice-rescoring pass. Instead, an *N-best rescoring* strategy is employed to (partially) realize their superior modeling power.

N-best rescoring, however, suffers from several known deficiencies and inefficiencies. As a thought experiment, one could sort all the hypotheses in a lattice using the simpler lattice-generating models, rescore each with the more complex models and ask where the highest-scoring word sequence *ranks*. N-best rescoring will suffer from search errors if N is smaller than this rank. But while considering a large number of hypotheses mitigates search errors, it is often computationally expensive, especially since most of the N options will score poorly using the complex model. The solution is to use the more complex models to aid hypotheses selection, as opposed to considering the N hypotheses chosen by the simpler models.

This work was partially supported by National Science Foundation Grant No 0963898 (IIS/RI). Frederick Jelinek contributed to this work and would be a co-author if he were available and willing to give his consent.

In this paper, we propose a hill climbing technique that operationalizes this idea. Hill climbing examines a neighborhood of an initial point, finds the direction in which the function is increasing most steeply, moves a suitable step in this direction to reach a new point, and iterates. For a broad class of problems, hill climbing is guaranteed to reach a local maximum of the function. To apply hill climbing ideas to lattice rescoring with complex models, we therefore need to view the alternatives in the lattice as points in a domain-space equipped with a notion of a neighborhood. A natural domain is to consider each word sequence in the lattice as a point, and the edit distance between two word sequences as the metric for defining neighborhoods. Starting with a word sequence in the lattice, we evaluate hypotheses in a small neighborhood with the complex model, step to the best one, and iterate, until a hypothesis scores higher than all the alternatives in its neighborhood.

Hill climbing for ASR has been investigated previously in [4] for rescoring confusion networks [5], which provide posterior probabilities for individual words in the hypotheses based on all the lattice-generating models. Scores from the complex models may be combined with these (total) probabilities during rescoring. The technique presented here evaluates explicit paths in the original lattice, permitting greater flexibility, e.g. for discriminatively combining the individual models used in lattice generation and rescoring [6].

Section 2 describes our hill climbing technique, and efficient construction of the neighborhoods using finite-state automata (FSA) techniques. Section 3 addresses the issue of non-convex optimization, i.e. avoiding local maxima. Sections 4 and 5 describe our experimental setup and results.

2. HILL CLIMBING ON SPEECH LATTICES

The ASR output from the first decoding pass is typically encoded as a lattice: a directed acyclic graph (DAG) with unique start and end nodes, nodes time-stamped w.r.t. the speech signal X, and edges labeled with words w. Each path in the DAG from start to end corresponds to a candidate time-aligned transcripts $W = w_1\, w_2\, \ldots\, w_n$ of X. Each edge in the DAG is also labeled with component model scores, e.g. log-probabilities from the acoustic model (Λ) and language model (Γ). The DAG structure respects the conditional independence assumptions of the component models, so that the score $g(X, W; \Lambda, \Gamma)$ of an entire path is the sum of the scores of the edges along the path.

We will investigate the replacement of the lattice-generating language model Γ with a long-span language model Γ_{new}. We will use hill climbing to find the path W in the lattice that maximizes

$$g(X, W; \Lambda, \Gamma_{\text{new}}) = \alpha \log P(X|W, \Lambda) + \log P(W|\Gamma_{\text{new}}), \quad (1)$$

where α is the inverse of the language model (LM) scaling factor.

In order to apply hill climbing to our problem, we need to define a *neighborhood* for each word sequence W in the lattice. Since our search space consists of a set of word sequences, it is natural to define the neighborhood function using the *edit distance* metric.

Specifically, we define the neighborhood of W at position i to be all the paths in the lattice whose word sequence may be obtained by editing — deleting, substituting or inserting a word to the left of — w_i. We will use $\mathcal{N}(W, i)$ to denote this "distance 1 at position i" neighborhood of W. We explain in Section 2.1 how the set of paths in $\mathcal{N}(W, i)$ can be efficiently generated from the representation of the lattice as a finite-state automaton (FSA) using the intersection operation.

We propose to undertake hill climbing as follows. At each step of the algorithm, a position i in the current word sequence W is selected. All paths in the lattice corresponding to word sequences $W' \in \mathcal{N}(W, i)$ are then extracted, along with the original acoustic scores on each edge. The new scores $g(X, W'; \Lambda, \Gamma_{\text{new}})$ are computed and the hypothesis $\hat{W}'(i)$ with the highest score becomes the new W. A new position i in the new W is then selected[1] and the process continues, until $W = \hat{W}'(1) = \ldots = \hat{W}'(n)$. In other words, the search terminates when W itself is the highest scoring hypothesis in its 1-neighborhood at all positions in i.

Section 2.2 described the hill climbing algorithm formally, and Section 2.3 discusses alternative ways of selecting the position i.

2.1. Efficient Generation of Neighborhoods

As mentioned above, we need to efficiently generate the neighborhood set of a given word sequence W at a specific position i.

To this end, note that the set of all word sequences that can be generated from W with one deletion, insertion or substitution at position i can be represented by a FSA. We will call this machine $LC(W, i)$. Figure 1 illustrates the construction of such a FSA. A word sequence $W = w_1\, w_2\, \ldots\, w_5$ is represented as a FSA in Figure 1(a) and Figure 1(b) represents $LC(W, 2)$, the distance 1 neighbors of W at position 2 . The ϵ arc $(1 \rightarrow 2)$ accounts for the neighbor with w_2 deleted, the arc with σ followed by ϵ $(1 \rightarrow 6 \rightarrow 2)$ corresponds to substituting w_2 with any word, including w_2 itself, i.e. W always belongs to $\mathcal{N}(W, i)$, and the path with σ followed by w_2 corresponds to one insertion to the left of w_2. Since the decision to insert only to the left of a position i is arbitrary, we also define the FSA $LC(W, n+1)$, which permits insertions to the right of the last word in W. $LC(W, 6)$ for the example above is shown in Figure 1(c).

Next, to restrict the neighboring $\mathcal{N}(W, i)$ of W to word sequences present in the lattice (our search space), $LC(W, i)$ is intersected with a weighted FSA representation of the lattice, L_{acoustic} whose weights are scaled acoustic scores $\alpha \log P(X_i|w_i, \Lambda)$ from the initial lattice:

$$LN(W, i) \leftarrow LC(W, i) \circ L_{\text{acoustic}}. \tag{2}$$

$LN(W, i)$ therefore is a weighted FSA representation of the subset of word sequences W' in $\mathcal{N}(W, i)$ that are also present in the initial lattice. The weights associated with the words in $LN(W, i)$ are the acoustic scores in the original lattice, which we will need for combining with the new language model scores $\log P(W'|\Gamma_{\text{new}})$ according to (1). Note also that although L_{acoustic} could be a huge WFSA, the intersection is fast and efficient because $LC(W, i)$ is deterministic, and has a very small number of states.

[1] We have experimented with selecting i sequentially from left to right at successive steps, returning to the beginning (i=1) from the end of W.

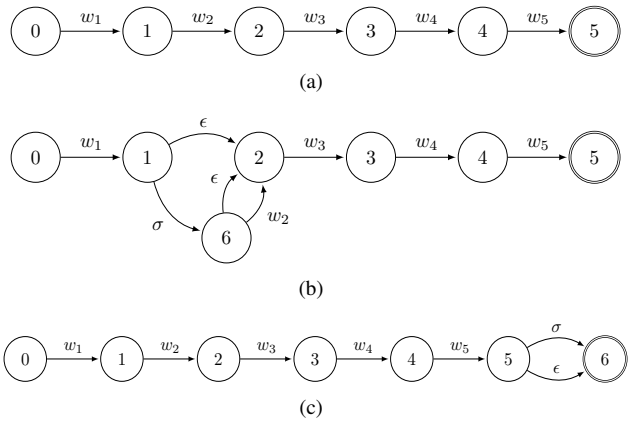


Fig. 1. The FSA representation of the neighborhood set of a given path: (a) Original path. (b) Neighborhood for a specific position. (c) Neighborhood for the last position.

2.2. The Hill Climbing Algorithm

We now introduce the algorithm formally using the notation developed previously. The three major steps are shown in Algorithm 1.

- **Initialization** where the highest scoring word sequence (the Viterbi path) from the initial lattice is selected;
- **Neighborhood Generation** discussed in Section 2.1;
- **Neighborhood Rescoring** which involves evaluating all the word sequences in the neighborhood set using (1), and selecting the word sequence with maximum score for the next step of the algorithm.

It is worth mentioning that our algorithm is based on a variation of hill climbing, called *steepest ascent* hill climbing, in which the best possible move is made at each step, among all possible moves (in our case the best path after rescoring by the new model among paths in the neighborhood set). In the basic hill climbing algorithm, any move in the neighborhood which improves the objective function may be made, not necessarily the move with the most improvement.

2.3. Choosing Positions to Explore Via Hill Climbing

Once the neighborhood $\mathcal{N}(W, i)$ of the current position in W has been explored and W has been updated, the hill climbing algorithm has many choices for the next position (the new i) to explore. We have made the expedient choice of $i \leftarrow i+1$ in Algorithm 1, i.e. the choice to explore the neighborhood(s) of W sequentially from left to right. We emphasize that this is by no means the only way to choose i, and possibly not even the optimal choice. For the local optimality of hill climbing, it suffices that the search terminates only if W is the highest scoring hypothesis in *all* its i neighborhoods.

A possibly smarter choice of i may be based on obtaining the lattice entropy at each w_i. Positions of high entropy indicate higher uncertainty about the choice of w_i, and positions of likely errors in W [7]. Attacking such positions first, with other positions fixed, will likely focus the rescoring effort where it is most needed, and result in even faster convergence to the highest scoring hypothesis.

3. MITIGATING THE EFFECTS OF LOCAL MAXIMA

Our algorithm, as true in general of hill climbing algorithms, may yield a local maximum solution as there is no guarantee that it finds

Algorithm 1 Steepest Ascent Hill Climbing for Rescoring

$L_{\text{acoustic}} \leftarrow$ **WFSA** of the initial lattice with weights representing scaled acoustic scores (scaled by α).
$v \leftarrow$ Viterbi path of the initial lattice //taking into account both initial acoustic and LM scores
$N \leftarrow$ **length**(v)
repeat
  $i \leftarrow 0$
  while $i \leq N + 1$ **do**
    // Local changes for the ith position
    create **FSA** $LC(v, i)$
    // Neighboring paths with their acoustic scores
    $LN(v, i) \leftarrow LC(v, i) \circ L_{\text{acoustic}}$
    //Rescoring and Choosing the best path
    $v \leftarrow Best(Rescore(LN(v, i)))$
    $N \leftarrow$**length**(v) // new length
    if DELETION **then**
      $i \leftarrow i$-1
    end if
    $i \leftarrow i$+1
  end while
until v does not change // Stopping criterion

the global maximum solution when applied to a non-convex space. Two common solutions to overcome this problem are,

1. **Random-restart hill climbing** where hill climbing is carried out using different random starting points (word sequences)
2. **Simulated Annealing** in which unlike hill climbing one chooses a random move from the neighborhood (recall that hill climbing chooses the best move from all those available, at least when using the steepest ascent variant). If the move results in a better word sequence (in terms of the score under the new model) than the current word sequence then simulated annealing will accept it. If the move is worse then it will be accepted according to some probability [8].

This paper considers the random-restart technique. Algorithm 1 is repeated M times, each time with a different initial word sequence. At the end of each iteration, the score (under the new model) of the resulting path is stored. Hence, we will have M different stopping paths , $(v_1, v_2, \cdots, v_M)$, along with their corresponding scores, $(g_1, g_2, \cdots, g_M)$. The path with the maximum score is selected as the final output of the algorithm. The M starting paths (word sequences) are selected by sampling the initial lattices (based on the distribution imposed by the initial model). In addition, we make sure that the sampled paths are not repeated and also for the first iteration we start from the viterbi path in the lattices.

4. EXPERIMENTAL SETUP

4.1. Corpora, Baseline and Rescoring Models

The ASR system used throughout this paper is based on the 2007 IBM Speech transcription system for GALE Distillation Go/No-go Evaluation [9]. The acoustic models used in this system are state-of-the-art discriminatively trained models and are the same ones used for all experiments presented in this paper. As a demonstration of our proposed rescoring framework, we first build a 3-gram language model with Kneser-Ney smoothing on $400M$ *broadcast news* LM training text. This 3-gram LM has about $2.4M$ N-grams and is used to generate initial lattices for all the experiments. For the rescoring experiments, we train two different models (to make sure our algorithm is applicable across different models) on the above LM text: 1) 4-gram LM with about $64M$ N-grams. 2) Model M shrinking based exponential LM [10]. The latter LM has been reported to have the state-of-the-art performance for the broadcast news task [11]. The evaluation set is the 2.5h rt04 evaluation set containing 45k words. The WER of the initial 3-gram LM on this data set is 15.51%. Additionally, we report rescoring results on dev04f using Model M to show the generalization of our algorithm across different data sets. The initial WER on dev04f using 3-gram LM is 17.03%.

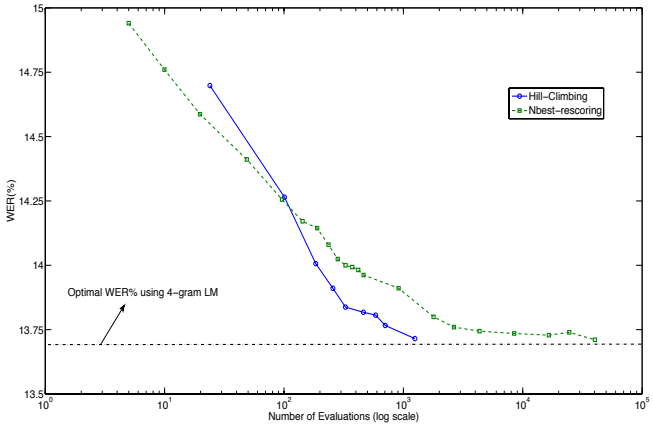
4.2. Evaluation of the Efficacy of Hill Climbing

We evaluate two different aspects of our proposed algorithm. First, comparison of the proposed algorithm and N-best rescoring. For this, we calculate the average number of word sequence evaluations needed for both methods to get to a particular WER (after rescoring under the new models). In our algorithm during the rescoring phase (of the neighborhood set), we need to query the LM score of each word sequence in the neighborhood set. In order to get the score, we use a look-up table of previously computed scores for word sequences, computing scores using the evaluation method for the rescoring model only for new word sequences which are not included in the look-up table. For each utterance, the effective number of performed evaluations by the rescoring model can be obtained from the size of the look-up table at the end of the hill climbing procedure. Also, for N-best rescoring we generate a list of word sequences in which all the paths are unique and hence the size of the generated list is essentially the number of effective evaluations. We observed in our experiments that for the same number of evaluations, the two algorithms took the same amount of time. Therefore, a reduction in the number of evaluations directly corresponds to a speedup in runtime. Second, the algorithms are analyzed based on how close they can get to the WER of the optimal solution (global maximum) of the rescoring model on the lattices.

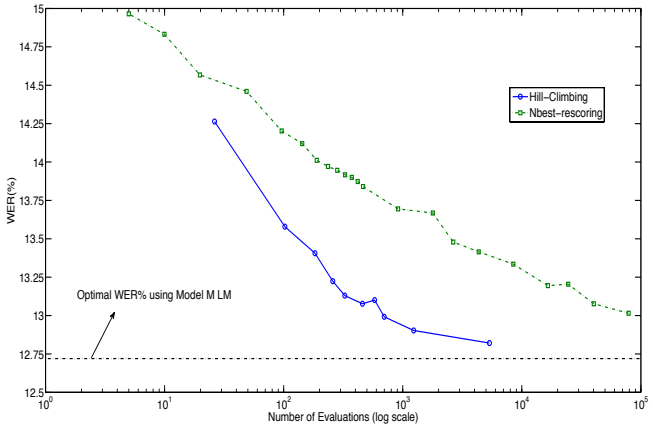
5. RESULTS AND DISCUSSION

The results for comparing our hill climbing method to N-best rescoring using 4-gram LM and Model M can be seen in Figure 2(a) and (b), respectively. Figure 3 also illustrates rescoring results using Model M LM on dev04f. The x-axis corresponds to the average number of effective evaluations (in $\log$ scale) calculated as described in Section 4.2 for each method. The y-axis shows the WER of the output of the algorithms for each experiment. For hill climbing experiments, we use $M = \{1, 5, 10, 15, 20, 30, 40, 50, 100\}$ starting paths (which corresponds to the different points of solid lines in all figures) and also we evaluate N-best rescoring with N ranging from 5 to 50000 (different points in dashed lines). In addition, the horizontal dashed line in the figures shows the WER of the optimal solution in the lattices using 4-gram and Model M LMs respectively. These optimal WERs can be calculated due to the fact that the rescoring models, which we used for our experiments, can be represented as **WFSA** and hence be composed with the lattices to extract the best path. However, we emphasize that this *can not* be done for models with longer dependencies and syntactic information and therefore, the need for a rescoring framework is inevitable with those models.

These figures show that our proposed hill climbing algorithm results in far fewer evaluations to reach competitive WERs, including the optimal WER, under both models. In fact, our algorithm produces similar WER results with speedups close to *two orders of magnitude.* This improvement is due to the characteristics of the hill climbing method where at each step the moves are selected (from



(a)



(b)

Fig. 2. Hill-Climbing vs. N-best rescoring on rt04: (a) 4-gram LM (b) Model M LM

neighborhood set) based on their quality under the new model (in contrast to N-best rescoring where the evaluating points are selected based on the initial model). An important observation, seen by comparing Figure 2(a) with Figure 2(b), is that the problem with N-best rescoring (non-efficiency in terms of effective evaluations) is more severe when the rescoring model is more different/orthogonal to the initial model. In our case, Model M LM is capturing knowledge which significantly differs from the initial 3-gram model, compared to the 4-gram vs. 3-gram LM. Hence, one can see in Figure 2(b) a large gap between the two rescoring methods in terms of number of needed evaluations.

6. CONCLUSIONS

We have introduced a novel rescoring framework on speech lattices based on hill climbing via edit-distance based neighborhoods. We have shown that the proposed method results in far fewer full-sentence evaluations by the complex model than conventional N-best rescoring. Although we have used the proposed algorithm to carry out only LM rescoring, we emphasize that the method is applicable for rescoring using a new acoustic model, with multiple knowledge sources and in a discriminative model combination scenario.

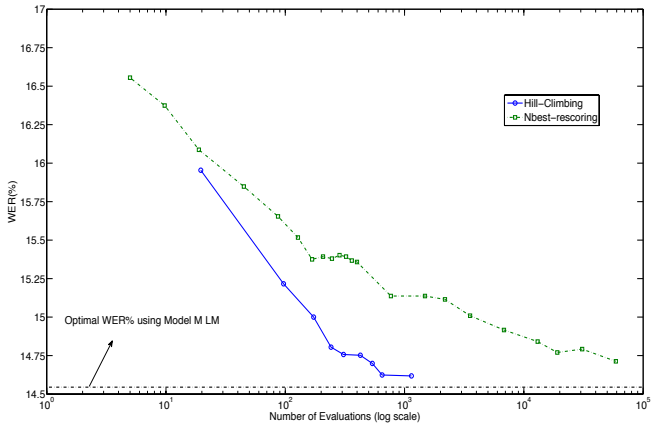


Fig. 3. Hill-Climbing vs. N-best rescoring on dev04f using Model M LM

7. REFERENCES

[1] S. Khudanpur and J. Wu, "Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling," *Computer Speech and Language*, pp. 355–372, 2000.

[2] C. Chelba and J. Frederick, "Structured language modeling," *Computer Speech and Language*, vol. 14, no. 4, pp. 283–332, Oct. 2000.

[3] H. K. J. Kuo, L. Mangu, A. Emami, I. Zitouni, and L. Young-Suk, "Syntactic features for Arabic speech recognition," in *Proc. ASRU*, 2009.

[4] A. Deoras and F. Jelinek, "Iterative decoding: A novel rescoring framework for confusion networks," in *Proc. ASRU*, 2009, pp. 282–286.

[5] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus among words: Lattice-based word error minimization," in *Sixth European Conference on Speech Communication and Technology*, 1999.

[6] P. Beyerlein, "Discriminative Model Combination," *Proc. ICASSP*, 1998.

[7] L. Burget and et al., "Combination of strongly and weakly constrained recognizers for reliable detection of OOVs," in *Proc. ICASSP*, 2008.

[8] S Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 671–680, 1983.

[9] S. Chen, B. Kingsbury, L. Mangu, D. Povey, G. Saon, H. Soltau, and G. Zweig, "Advances in speech transcription at IBM under the DARPA EARS program," *IEEE Transactions on Audio, Speech and Language Processing*, pp. 1596–1608, 2006.

[10] S. F. Chen, "Shrinking exponential language models," in *Proc. NAACL-HLT*, 2009.

[11] S. F. Chen, L. Mangu, B. Ramabhadran, and et. al, "Scaling shrinkage-based language models," *Proc. ASRU*, 2009.