A Pruned RNNLM Lattice-Rescoring Algorithm for Automatic Speech Recognition

Hainan Xu[1,2], Tongfei Chen[1], Dongji Gao[1], Yiming Wang[1], Ke Li[1], Nagendra Goel[3], Yishay Carmiel[2], Daniel Povey[1], Sanjeev Khudanpur[1]

[1]Center for Language and Speech Processing, Johns Hopkins University; [2]IntelligentWire, Seattle WA; [3]Go-Vivace Inc., USA

Overview

- Usually lattice-rescoring uses n-gram approximation to limit search space;
- We propose a heuristics that finds more promising arcs to expand, and use it for pruning;
- Complexity of the algorithm grows approximately (empirically) linear with n-gram order, compared with exponential growth of the baseline algorithm;
- 4X and 10X faster for 4-gram and 5-gram;
- The heuristics also consistently improves WER;
- The evaluation is done with TensorFlow RNNLMs. We open source the integration of TensorFlow to Kaldi.

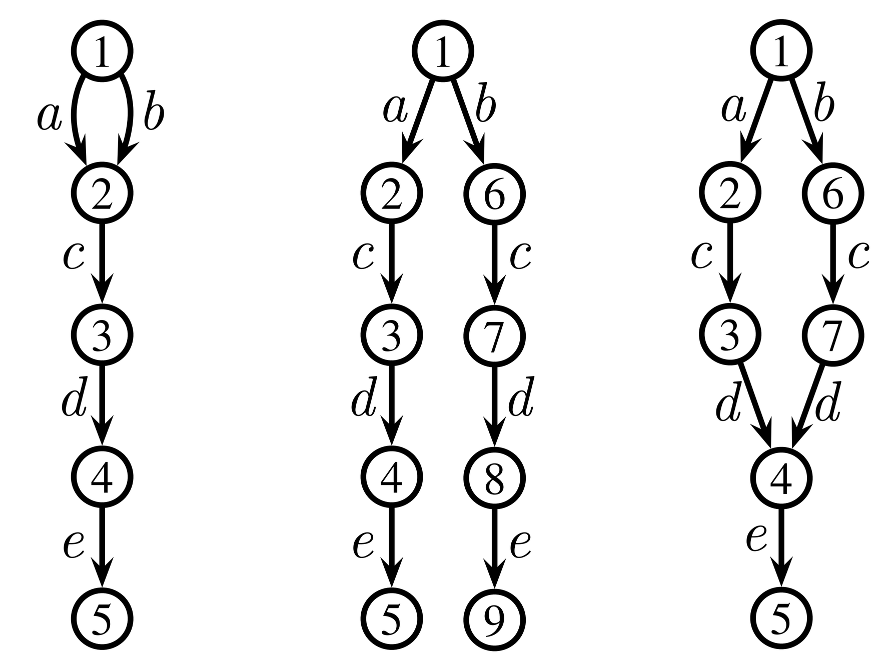
Lattice Rescoring

- In speech recognition, decoding is usually done on a static decoding graph compiled from an n-gram;
- RNNLM rescoring helps further reduce WERs by (partially) replacing LMs weights on a decoded lattice;
- A naive implementation to rescore the lattice is too costly – it runs exponentially w.r.t. lattice-depth;
- An n-gram approximation algorithm is commonly used in order to limit the search space.

Contact Information

- Web: `http://www.hainanxv.com`
- Email: hxu31@jhu.edu

Analysis of Old Algorithm



(a) (b) (c)

- In 3-gram approximation, states 4 and 8 in (b) are merged as state 4 in (c);
- state 4 encodes history of either (a, c, d) or (b, c, d). The choice is arbitrary, and affects the weight computed for p(e | 4).

Pruned Algorithm

- For each arc to be expanded, we compute a score reflecting how likely this arc will become part of the best-path;
- Arcs that are not very promising (out of the beam) are not expanded;
- Arcs that are more promising get expanded first, so that output lattice states encode "better" history.

Heuristic

- The heuristic is computed as

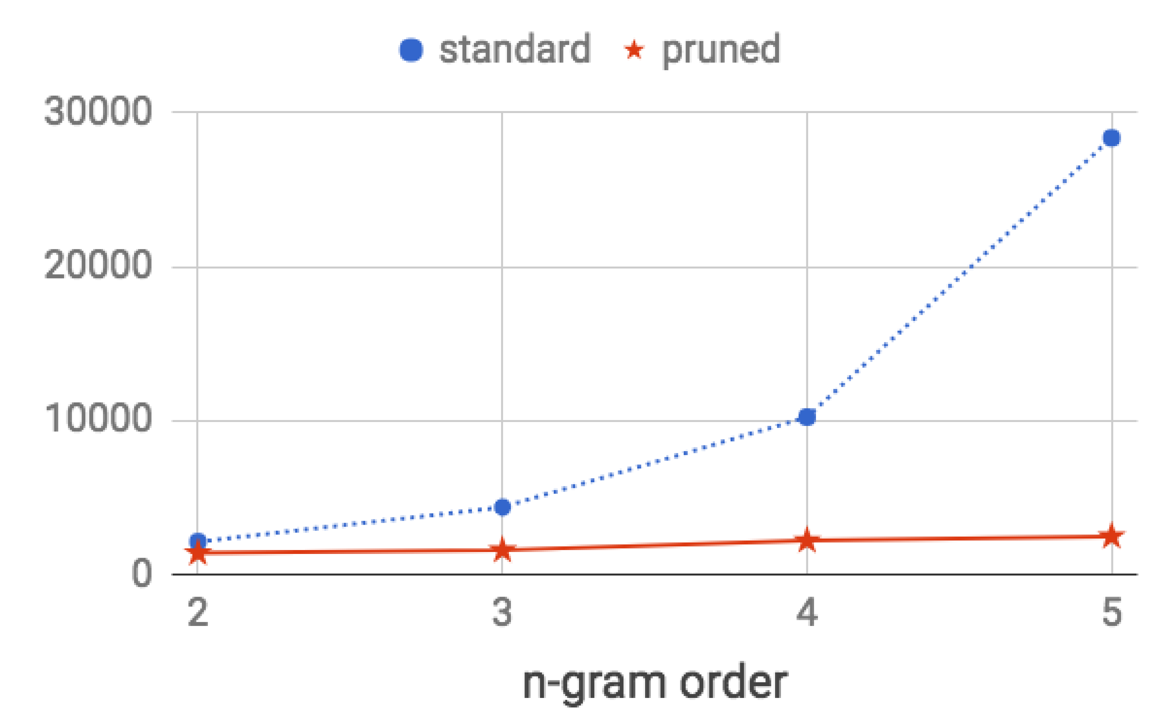
$$H(c) = \alpha(c) + \beta(a) + \delta(c) \quad (1)$$

- c: a state in the output lattice;
- a: the corresponding state in the input lattice;
- $\alpha(c)$ is the forward-cost for c in the output lattice
- $\beta(a)$ is the backward-cost for a in the input lattice
- $\delta(c)$ is an "expectation" of $\beta(c) - \beta(a)$

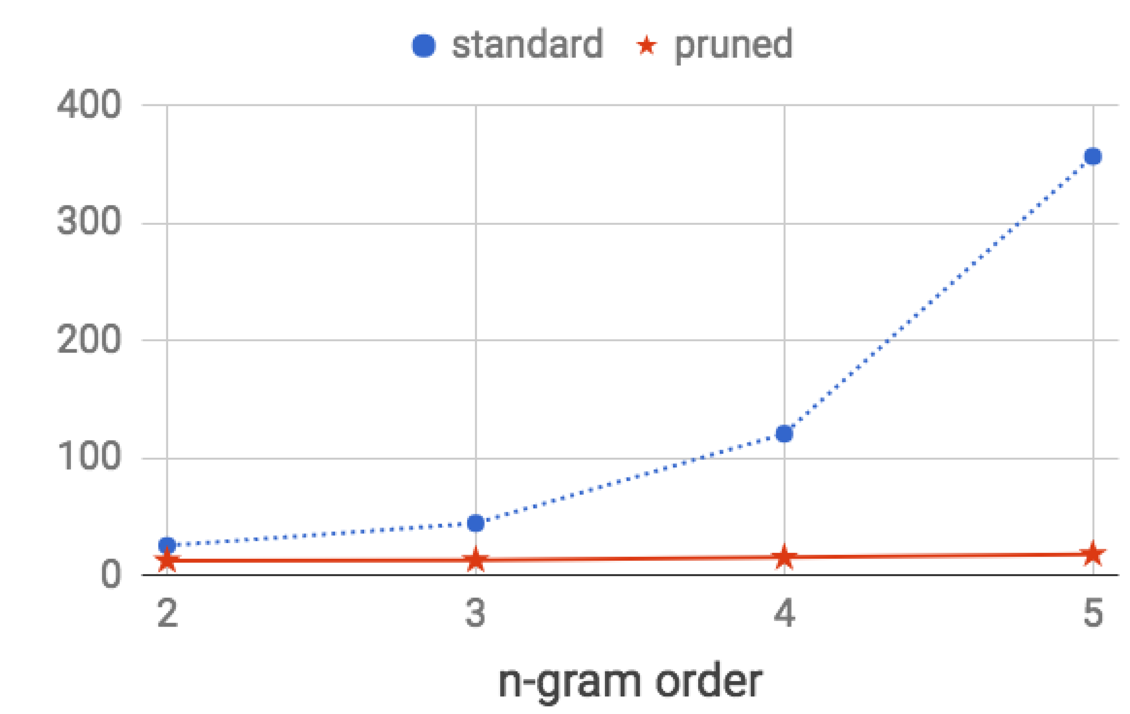
$$\delta(c) = \begin{cases} \beta(c) - \beta(a), & \beta(c) < +\infty \\ \delta(\text{prev}(c)), & \beta(c) = +\infty \end{cases} \quad (2)$$

prev(c) is the previous state of c on the best path from start to c.

Lattice-rescoring Speed



Output Lattice Size (arcs per frame)



Acknowledgements

This work was partially supported by DARPA LORELEI award number HR0011-15-2-0024, NSF Grant No CRI-1513128 and IARPA MATERIAL award number FA8650-17-C-9115 and by IntelligentWire. The authors would also like to thank the TensorFlow team at Google for their help during the project.

Word-error-rate

Corpus	Test set	ARPA baseline	RNNLM rescoring with n-gram approximation: 2-gram standard	2-gram pruned	3-gram standard	3-gram pruned	4-gram standard	4-gram pruned
AMI-IHM (0.5)	dev	24.2	24.5	**24.0**	23.7	**23.4**	23.4	**23.3**
	eval	25.4	25.8	**25.0**	24.6	**24.4**	24.3	**24.2**
SWBD (0.8)	swbd	8.1	8.6	**8.2**	7.4	**7.2**	7.2	**7.1**
	eval2000	12.4	12.9	**12.3**	11.7	**11.5**	11.5	**11.3**
WSJ (0.8)	dev93	7.6	7.2	**6.9**	6.4	**6.2**	6.4	**6.2**
	eval92	5.1	4.6	**4.2**	4.1	**3.9**	3.9	**3.8**
LIB (0.5)	test-clean	6.0	5.5	**5.1**	4.9	**4.8**	4.8	**4.7**
	test-other	15.0	14.0	**13.2**	12.7	**12.4**	12.4	**12.3**
	dev-clean	5.7	5.0	**4.8**	4.4	**4.3**	**4.3**	**4.3**
	dev-other	14.5	13.7	**12.9**	12.3	**12.0**	11.9	**11.7**

Table 1: WER of Lattice-rescoring of Different RNNLMs