NEURAL NETWORK LANGUAGE MODELING WITH LETTER-BASED FEATURES AND IMPORTANCE SAMPLING

Hainan Xu[1], *Ke Li*[1], *Yiming Wang*[1], *Jian Wang*[2], *Shiyin Kang*[3], *Xie Chen*[4], *Daniel Povey*[1], *Sanjeev Khudanpur*[1]

[1]Center for Language and Speech Processing, Johns Hopkins University;
[2]Toutiao AI Lab, Beijing, China; [3]Tencent AI Lab, Shenzhen, China;
[4]Machine Intelligence Laboratory, Cambridge University
{hxu31,kli26,yiming.wang,khudanpur}@jhu.edu,
{wantee.wang,kangshiyin,dpovey}@gmail.com, xc257@eng.cam.ac.uk

ABSTRACT

In this paper we describe an extension of the Kaldi software toolkit to support neural-based language modeling, intended for use in automatic speech recognition (ASR) and related tasks. We combine the use of subword features (letter n-grams) and one-hot encoding of frequent words so that the models can handle large vocabularies containing infrequent words. We propose a new objective function that allows for training of unnormalized probabilities. An importance sampling based method is supported to speed up training when the vocabulary is large. Experimental results on five corpora show that Kaldi-RNNLM rivals other recurrent neural network language model toolkits both on performance and training speed.

Index Terms— language modeling, recurrent neural networks, importance sampling, automatic speech recognition

1. INTRODUCTION

The language model is a vital component of the speech recognition pipeline. For many years, back-off n-gram models were the dominant approach [1]. However they are limited in their ability to model long-range dependencies and rare combinations of words. In [2], a neural network based language model is proposed. By modeling the language in continuous space, it alleviates the data sparsity issue. Its effectiveness has been shown in its successful application in large vocabulary continuous speech recognition tasks [3].

Recurrent neural network language models (RNNLMs) were proposed in [4]. The recurrent connections enable the modeling of long-range dependencies, and models of this type can significantly improve over n-gram models. More recent work has moved on to other topologies, such as LSTMs (e.g. see [5] for a recent example). We use the term RNNLMs here as a general term for such recurrent topologies, including LSTMs, GRUs etc.

However, the better performances brought by RNNLMs come at a cost - compared with regular n-gram language models, RNNLMs take several orders of magnitude more time to run, both for training and decoding. A key reason for its inefficiency is the softmax normalization at the output layer, which mean that all words' probabilities need to be computed in order to compute any single word's probability. To achieve good performance with RNNLMs, it is common to have a vocabulary of hundreds of thousands of words, and this inevitably leads to poor efficiency of the computation, both in training time and at test time (e.g. when rescoring lattices).

In this paper, we describe Kaldi-RNNLM, an extension of the Kaldi [6] speech recognition toolkit to support neural network language models. We use a novel training criterion that acts like the cross-entropy objective but allows training of unnormalized probabilities, and we use importance sampling during training to avoid having to access the representations of all the words on each minibatch. This design allows for fast run-time for both training and testing. Kaldi-RNNLM incorporates subword-level features for representing words and makes predictions at word-level. The subword-level features can better represent rarely-seen or even *out-of-vocabulary* (OOV) words.

2. PRIOR WORK

There has been prior work on neural-based language models, in general or specifically for ASR tasks. Mikolov's toolkit RNNLM [4] combines a simple vanilla RNNLM with direct-connection layers and a hashing mechanism to cluster histories to achieve then state-of-the-art performances on language modeling. In [7], RNNLM performance is improved by providing a contextual real-valued input vector, which conveys contextual information about the sentences being modeled, in association with each word. CUED-RNNLM [8] supports 2 additional training criteria, i.e. *noise contrastive estima-*

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.

tion [9] and *variance regularization* [10], which allows training unnormalized probabilities for fast testing. *Hierarchical RNNLMs* [11] uses a hierarchical structure for the last layer, clustering words into groups which allows the computation of softmax normalization term to be on a smaller number of word-groups instead of the vocabulary size.

Recently subword-level language models that make word-level prediction gain growing interest because of their superior performance compared with word-level models. In [12], subword features e.g. syllables and characters is combined in the feature to improve performance evaluated by *perplexity* and *word error rate* (WER) in ASR tasks. In [13], a *convolutional neural network* based character-level language model is proposed which achieves the state-of-the-art perplexities on Penn Treebank dataset with fewer parameters. In [14], a weighted combination of character-level and word-level features is used, obtaining better perplexities on several English corpora.

3. FEATURE REPRESENTATION OF WORDS

Inspired by [15], we propose a simple but effective representation of words that can incorporate subword information and thus deal with OOV words. Given a word (e.g., `nice`), we break it into a combination of letter n-grams (e.g `^n`, `^ni`, `nic`, `ice`, `ce$`, `e$`). We then optionally append two augmented features: the unigram probability in log space and the word-length of each word to its corresponding letter n-gram vector. The former is for better generalization when a neural language model trained on in-domain dataset is applied to out-of-domain data. In sum, besides a one-hot representation for the most frequent words, each word has additional representation as a vector of counts of its letter n-grams and two extra augmented features, and the word-level embedding vector is a sum over terms corresponding to each feature. This can be represented as a subword level embedding matrix, which is trained jointly with the neural language model.

Kaldi-RNNLM shares the input and output embeddings for the neural-network, following [16]. Combined with using subword features, this allows using very large vocabularies in Kaldi-RNNLM without having to use a shortlist to avoid the data-sparsity issues.

4. A NEW OBJECTIVE FUNCTION FOR TRAINING UNNORMALIZED PROBABILITIES

In the standard cross-entropy training, if we write $\mathbf{z}$ as the layer of the neural network before the final softmax operation, and j as the index for the correct word, then the objective for one data point is

$$z_j - \log \sum_i \exp(z_i) \tag{1}$$

We note that $\log x \le x - 1$, and define the following objective function,

$$z_j + 1 - \sum_i \exp(z_i) \tag{2}$$

Note that (2) $\le$ (1), with equality iff $\sum_i \exp(z_i) = 1$, i.e. our objective is a lower bound on the cross-entropy objective, with equality when the output is a well-normalized probability distribution.

Therefore, when the new objective is maximized, it is similar to cross-entropy training plus a penalty term that makes the output of the network sum to a value close to 1 ($\sum_i \exp(z_i) \simeq 1$).

During test time, instead of computing quantities (1) or (2), we simply use z_j as the computed "probability" as an approximation since we know the expectation of $1 - \sum_i \exp(z_i)$ is 0. This can greatly speed up the computation in tasks where the label for which we want to compute a probability is known, including lattice and n-best list rescoring for speech recognition and machine translation.

4.1. Stability of Training

One potential problem of using (2) as the objective function is potential instabilities during training, especially in the beginning stages. Instabilities happen here because of the exp terms, which could result in very high derivatives computed. We propose two methods to deal with such problems and ensure convergence in training.

The 1st method is careful weight initialization. Instead of initializing the weights of the output embedding layer with 0 mean, empirically, having a mean value of around $-\log(\text{vocab-size})$ would prevent instabilities from happening.

The 2nd method is to transform z's with the following function, and compute the objective on $f(z)$'s instead of z's[1].

$$f(z) = \begin{cases} z & \text{if } z \le 0 \\ \log(z+1) & \text{if } z > 0 \end{cases} \tag{3}$$

Function f keeps the output of the network from getting too large, ensuring the derivatives computed to have reasonable values, thus keeping the training stable.

Kaldi-RNNLM uses method 2 to prevent training instabilities. Method 1 is used in Kaldi's TensorFlow RNNLM setup described in [17].

5. SAMPLING ALGORITHMS

To compute the new objective function, the term $\sum_i \exp(z_i)$ still requires looping over all vocabulary words. In Kaldi-RNNLM, we use a sampling-based method to compute an unbiased estimate of this quantity.

[1]This is equivalent to having an "element-wise f function" layer in the neural network before the output but we implemented this in the function where it computes the objective function

Note that it is always possible to compute such sum terms using sampling-based methods, with the guarantee that the sum is an unbiased estimator. However, such guarantee does not carry through a nonlinear operation like $\log$, and thus in the standard cross-entropy systems, using an importance-sampling based method will inevitably add bias in the estimate. In Kaldi-RNNLM, using the new objective function (2), taking the sum out of the log operation makes an unbiased importance-sampling based training possible.

The interaction between the training and sampling is as follows: say the set of vocabulary words is W, we have a minibatch of k sequences, and we use a sample size of m. In each minibatch, we have k data-points to train on, including k (or fewer) histories (a set H) and k or fewer "correct words" (a set Y). We first generate for each word w in our vocabulary an inclusion probability $p(w)$, which is the probability we will include word w in our sample of size m. For words that must be included (words in Y), this is 1, and in any case $p(w) \leq 1$. The inclusion probabilities must sum to the number of samples m (we do not allow duplicates).

On each minibatch we randomly sample a set of words S, of size m (e.g. 512). The summations in the objective function are limited to those for words $w \in S$, but weighted by the inverse of the inclusion probabilities $p(w)$. This is the standard approach used in importance sampling, and it ensures that the objective function and its derivatives are distributions with mean equal to that they would be in the unsampled case.

5.1. Sampling distribution

For importance-sampling methods the only hard constraint on the distribution that we sample from is that the probabilities should be bounded away from zero; it is best to have $p(w)$ be larger for words that are expected to have a higher predicted probability, as this will minimize the variance of the computed derivatives[2].

In our work, in each minibatch in training, we sample the words from a distribution that is computed by averging the n-gram distributions of all histories in the minibatch. These distributions come from an n-gram backoff model trained on our training corpus, estimated and pruned in a way that makes it efficient to sample from.

5.2. 2-stage sampling

We follow the *unequal probability systematic sampling* algorithm described in [18] to sample the subset of words. One issue arises because the original algorithm needs to loop over all words once, and this could be quite costly for large vocabularies. Because we do not require independence in the selected samples, we follow a "2-stage" sampling procedure, where in order to sample m words from all words, we first divide the words into groups in a way that, the sum of the inclusion probabilities of words in each group is less than 1. In the 1st stage, we sample m groups from all the groups; in the 2nd stage, from each group, we sample exactly one word according to their probabilities. This allows that only a small subset of words needs to be considered and greatly speed up the sampling procedure.

[2]This is a complicated topic, and a proper analysis is beyond the scope of this paper

6. EVALUATIONS

We report the performance on 4 datasets in English, i.e. AMI-IHM, SWBD, WSJ and TED-LIUM, as well as Hub4 in Spanish. We compare Kaldi-RNNLM with CUED-RNNLM (CUED) and TensorFlow[19] based RNNLMs (TF) in terms of perplexities, as well as performance in ASR tasks, where we will compare the WERs after performing lattice-rescoring with different RNNLMs. We will also include stats for training speed for different toolkits.

6.1. Perplexities

Table 1 compares the perplexity (PPL) results of different RNNLM toolkits. For Kaldi-RNNLM, we train models on the full vocabulary, and normalize the output of the neural network in order to compute valid perplexities. For CUED-RNNLM and TensorFlow, we train language models on a shortlist of most frequent unigrams, and report the perplexities after adding a correction term (by distributing the probability predicted for the "[oos]" symbol according to the unigram count of words, with smoothing). We see that overall, Kaldi-RNNLM achieves better perplexities than other toolkits, and we hypothesize it is the result of having subword features making the model more robust to rarely seen words in corpora.

PPL	dataset	CUED	TF	KALDI
AMI	train	52.2	49.1	47.0
	dev	76.5	82.1	**72.2**
HUB4	train	99.7	131.2	73.7
	dev	197.5	192.0	**180.7**
SWBD	train	37.7	46.7	43.4
	dev	52.0	54.3	**47.5**
WSJ	train	39.1	37.4	43.4
	dev	67.2	64.5	**50.9**
TED-LIUM	train	123.3	133.4	96.3
	dev	169.6	154.7	**137.0**

Table 1: Perplexities of Different RNNLMs

6.2. Lattice-rescoring

Table 2 reports the WER performance in different recipes, where we give the un-rescored baseline numbers, as well as WERs after lattice-rescoring with different RNNLMs. The acoustic models are trained with the nnet3 setup of Kaldi, and we use its latest developments including *pronunciation and silence probability modeling* [20], *TDNN models* [21], *lattice-free maximum mutual information models* [22] and *backstitch* optimization method [23]. For the SWBD acoustic model, *speed perturbation* [24] is used. For lattice-rescoring, we use the *pruned lattice-rescoring* algorithm described in [17], and a 4-gram approximation is used in all the experiments.

PPL	dataset	Baseline	CUED	TF	KALDI
AMI	dev	24.2	23.0	23.2	**22.8**
	eval	25.4	**23.9**	24.2	**23.9**
HUB4	test	14.4	12.8	13.1	**12.6**
SWBD	swbd	8.0*	**7.0**	7.1	**7.0**
WSJ	dev93	7.1*	6.1	6.0	**5.8**
	eval92	5.0*	3.9	**3.8**	3.9
TED-LIUM	dev	10.7*	10.3	10.4	**9.9**
	test	9.8*	9.3	9.3	**9.0**

Table 2: WER of Lattice-rescoring of Different RNNLMs; the baseline numbers with * means it is rescored by a (count-based) 4-gram model; all RNNLM numbers used a 4-gram approximation in rescoring.

6.3. Training Speed

We report the training speed of different RNNLMs, in terms of number of words per second, when training a simple 1 hidden-layer LSTM model for the AMI corpus. The hidden dimensions are fixed to be 200 and we use the same shortlist with 10000 words; we fix minibatch-sizes to be 64 and chunksizes to be 20 for all the experiments. The evaluation is done on a Tesla K10.G2.8GB GPU. Table 3 shows the speed of different toolkits in number of thousands-of-words per second. We can see that with sampling, Kaldi-RNNLM achieves the best speed among all the toolkits.

7. CONCLUSION AND FUTURE WORK

We introduce Kaldi-RNNLM, which uses an importance-sampling based method to speed up training, and applies a new method to train unnormalized probabilities. Subword level information such as letter n-gram features and two augmented features are utilized in generating word-embedding for better representing rare or out-of-vocabulary words. Experimental results show that Kaldi-RNNLM achieves better perplexities and comparable performance in ASR tasks when rescoring decoded lattices.

In the future, we plan to investigate more into how to incorporate other types of subword information for word representation to improve language modeling performances. We are also going to investigate more flexible ways to generate embedding matrices.

RNNLM	Speed (words/second)
CUED-RNNLM, CE	8.30K
CUED-RNNLM, NCE	12.8K
TensorFlow-RNNLM	23.3K
Kaldi-RNNLM, no sampling	18.3K
Kaldi-RNNLM, 512 samples	31.0K

Table 3: Training Speed of Different RNNLMs

References

[1] Joshua T Goodman, “A bit of progress in language modeling,” *Computer Speech & Language*, vol. 15, no. 4, pp. 403–434, 2001.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[3] Holger Schwenk and Jean-Luc Gauvain, “Connectionist language modeling for large vocabulary continuous speech recognition,” in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. IEEE, 2002, vol. 1, pp. I–765.

[4] Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky, “Rnnlm-recurrent neural network language modeling toolkit,” in *Proc. of the 2011 ASRU Workshop*, 2011, pp. 196–201.

[5] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu, “Exploring the limits of language modeling,” *arXiv preprint arXiv:1602.02410*, 2016.

[6] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., “The kaldi speech recognition toolkit,” in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, number EPFL-CONF-192584.

[7] Tomas Mikolov and Geoffrey Zweig, "Context dependent recurrent neural network language model," in *Proceedings of SLT*, 2012.

[8] Xie Chen, Xunying Liu, Yanmin Qian, MJF Gales, and Philip C Woodland, "Cued-rnnlman open-source toolkit for efficient training and evaluation of recurrent neural network language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 6000–6004.

[9] Michael Gutmann and Aapo Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 297–304.

[10] Yongzhe Shi, Wei-Qiang Zhang, Meng Cai, and Jia Liu, "Variance regularization of rnnlm for speech recognition.," in *ICASSP*, 2014, pp. 4893–4897.

[11] Hong-Kwang Kuo, Ebru Arısoy, Ahmad Emami, and Paul Vozila, "Large scale hierarchical neural network language models," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[12] Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky, "Subword language modeling with neural networks," *preprint (http://www.fit.vutbr.cz/imikolov/rnnlm/char.pdf)*, 2012.

[13] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush, "Character-aware neural language models," in *AAAI*, 2016, pp. 2741–2749.

[14] Yasumasa Miyomoto and Kyunghyun Cho, "Gated word-character recurrent language model," *preprint (https://arxiv.org/abs/1606.01700)*, 2016.

[15] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proceedings of the 22nd ACM international conference on information & knowledge management*. ACM, 2013, pp. 2333–2338.

[16] Ofir Press and Lior Wolf, "Using the output embedding to improve language models," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Valencia, Spain, April 2017, pp. 157–163, Association for Computational Linguistics.

[17] Hainan Xu, Tongfei Chen, Dongji Gao, Yiming Wang, Ke Li, Nagendra Goel, Yishay Carmiel, Daniel Povey, and Sanjeev Khudanpur, "A pruned rnnlm lattice-rescoring algorithm for aumatic speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018.

[18] Jean-Claude Deville and Yves Tille, "Unequal probability sampling without replacement through a splitting method," *Biometrika*, vol. 85, no. 1, pp. 89–101, 1998.

[19] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[20] Guoguo Chen, Hainan Xu, Minhua Wu, Daniel Povey, and Sanjeev Khudanpur, "Pronunciation and silence probability modeling for asr," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[21] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[22] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi.," in *INTERSPEECH*, 2016, pp. 2751–2755.

[23] Yiming Wang, Vijayaditya Peddinti, Hainan Xu, Xiaohui Zhang, Daniel Povey, and Sanjeev Khudanpur, "Backstitch: Counteracting finite-sample bias via negative steps," *Interspeech*, 2017.

[24] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5220–5224.