CUED-RNNLM – An Open-Source Toolkit for Efficient Training and Evaluation of Recurrent Neural Network Language Models

Xie Chen, Xunying Liu, Yanmin Qian, Mark Gales and Phil Woodland

April 1, 2016

Cambridge University Engineering Department

Overview

- RNNLM Overview
- Introduction of CUED-RNNLM
- Experiments on AMI corpus

Overview of Statistical Language Models

- Language Model (LM): Estimate probability of word sequence

$$P(W) = P(w_1, w_2, ...w_K) = \prod_{k=1}^{K} P(w_k|w_{k-1}, ...w_1)$$

- Three widely used language models
  - N-Gram Language Models (from 1980s)
  - Feed Forward Neural Network Language Models (from 2001)
  - Recurrent Neural Network Language Models (from 2010)

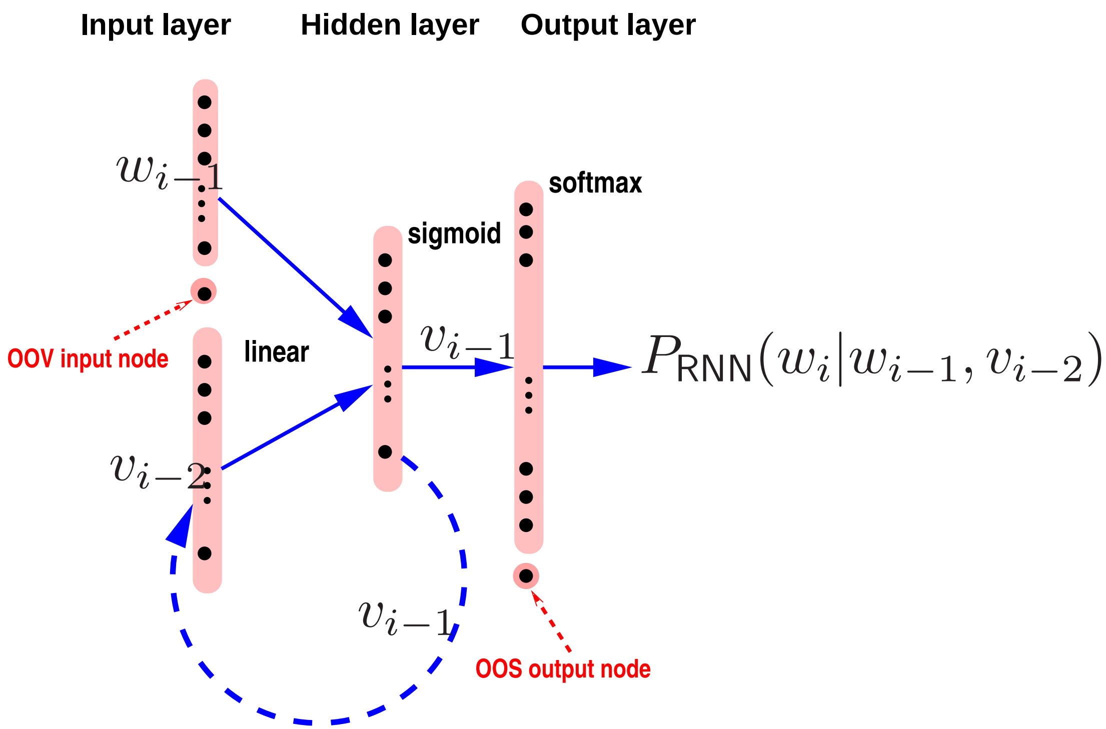
N-Gram Language Models

- Only related to previous $N-1$ words, ML used to estimate parameter

$$P(w_k|w_{k-1},...w_1) \approx P(w_k|w_{k-1},...w_{k-N-1})$$

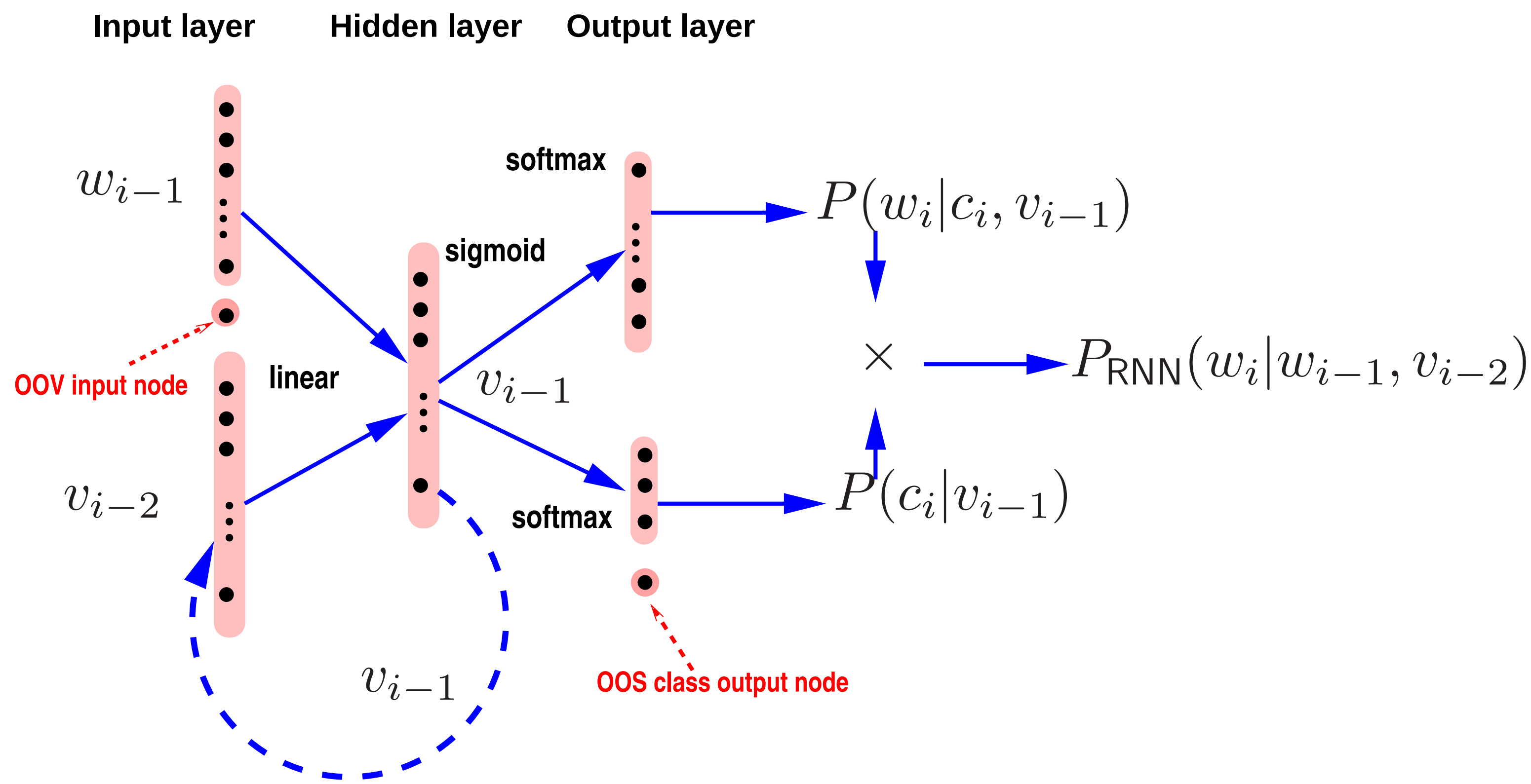
- Most popular LM over two decades
- Easy to implement
- Drawbacks
  - Data sparsity, e.g. $|V| = 1000$, a 4-gram LM needs $1000^4 = 10^{12}$ parameter – smoothing is necessary
  - Cannot model long term history, only consider last $N-1$ words

Recurrent Neural Network LMs



- 1-of-K coding for word in input layer
- Each word projected to a low and continuous space – solve data sparsity
- Long term history to be modeled

Class based Recurrent Neural Network LMs



- Use factorized output layer
- Computation reduced significantly

Existing toolkits for RNNLM

- Toolkits for RNNLM training
  - RNNLM toolkit – by Tomas Mikolov
  - RWTHLM – by RWTH Aachen University
    * Trained on CPU
    * Class based output layer used to reduce computation
    * Lack of parallel implementation
- Popular Toolkits for deep learning
  - Theano – by University of Montreal
  - Tensorflow – by Google
  - CNTK – by Microsoft
    * Support RNN implementation using GPU
    * Designed for general deep learning, not optimized for language model
- Issue: slow to train on large data and model size for RNNLM

Highlights of CUED-RNNLM

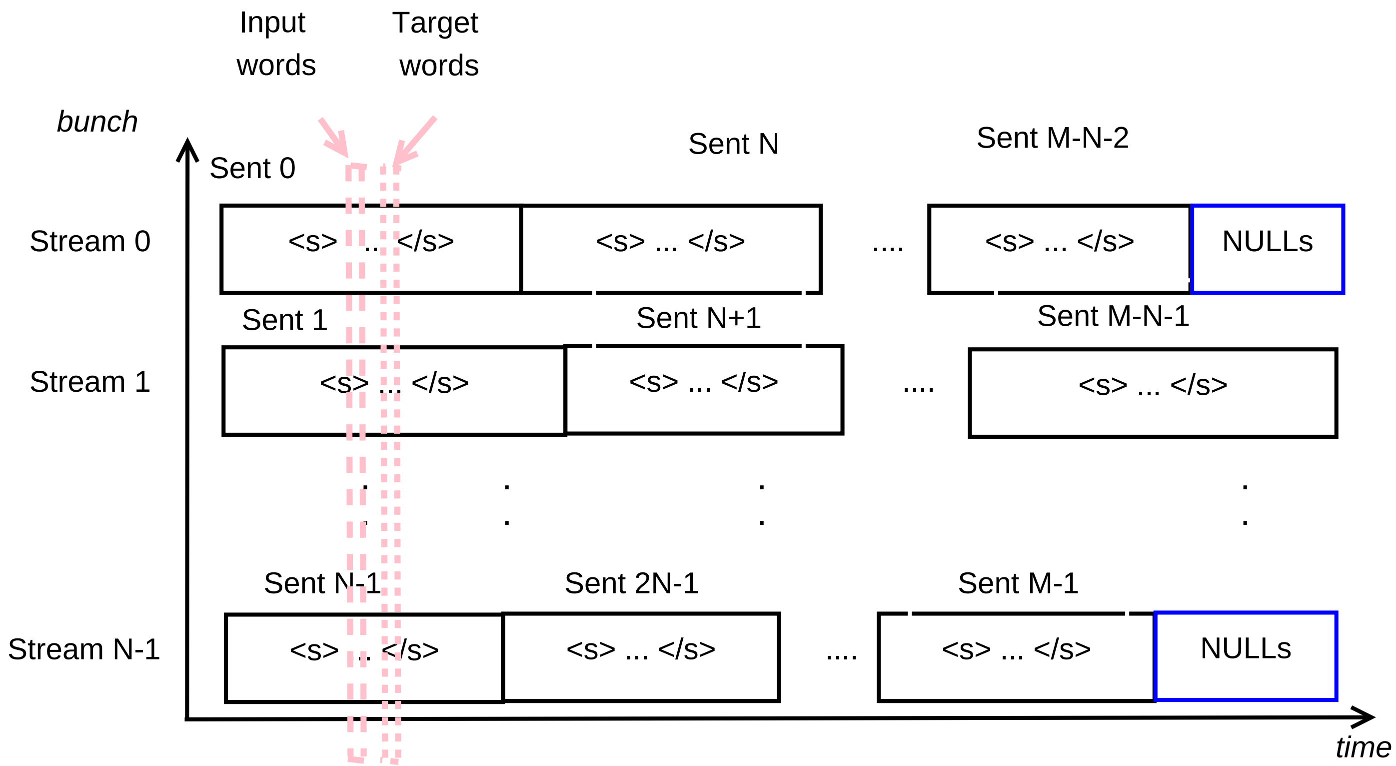
- CUDA
  - class and full output layer
  - minibatch training with GPU implementation

- Efficient training/evaluation criteria
  - standard cross entropy based training
  - variance regularization
  - noise contrastive estimation

- RNNLM Lattice rescoring integration with HTK 3.5
  - n-gram approximation and history vector clustering
  - support HTK lattice directly
  - conversion tools provided to support Kaldi lattice

Spliced Sentence Bunch

- Enable RNNLMs to be trained using bunch (i.e. minibatch) mode
- The number of NULL token is minimized



Network Configuration Support

- Model structure
  - full output layer
  - class based output layer
  - additional feature in the input layer
  - multiple hidden layers
- Specified input and output list
- OOV node in the input layer, OOS node in the output layer

Train Criteria in CUED-RNNLM

- Cross entropy (CE)

$$J^{\mathsf{CE}}(\theta) \;=\; -\frac{1}{N_w}\sum_{i=1}^{N_w} \ln P_{\mathsf{RNN}}(w_i|h_i)$$

- Variance regularization (VR)

$$J^{VR}(\theta) = J^{CE}(\theta) + \frac{\gamma}{2}\frac{1}{N_w}\sum_{i=1}^{N_w}((\ln(Z_i) - (\overline{lnZ}))^2)$$

- Noise contrastive estimation (NCE)

$$J^{\mathsf{NCE}}(\theta) \;=\; -\frac{1}{N_w}\sum_{i=1}^{N_w}(\ln P(C_{w_i}^{\mathsf{RNN}} = 1|w_i, h_i) + \sum_{j=1}^{k} \ln P(C_{\check{w}_{i,j}}^{n} = 1|\check{w}_{i,j}, h_i))$$

Additional Feature in CUED-RNNLM

- Perplexity calculation
- N-best rescoring
  - unnormalized probability to be applied (for VR and NCE trained model)
- Sampling sentences from well-trained RNNLMs
- Appended feature in input layer, e.g. LDA based topic representation
- ReLU for hidden node
  - faster convergence and slightly better performance

Experiments Setup

- Acoustic Model
  - AMI Kaldi recipe used
  - 78 hours data
  - sequence training for DNN
  - DNN: 6 hidden layers, each layer with 2048 hidden nodes, 4000 targets
  - Lattice generated from Kaldi, and converted to HTK format

- Language Model
  - 1M AMI transcription + 13M fisher data
  - 49k word decoding vocabulary
  - 33k RNNLM input vocabulary, 22k RNNLM output vocabulary
  - 512 hidden nodes
  - Full output layer RNNLMs (F-RNNLMs) trained by CUED-RNNLM
  - Class based RNNLM (C-RNNLMs) trained by Mikolov's RNNLM Toolkit
  - RNNLMs are interpolated with n-gram LM using weight 0.5

Experiments on 1M AMI transcription

LM Type	Train Crit	Re score	PPL dev	PPL eval	WER dev	WER eval
3g	-	-	93.6	82.8	25.2	25.4
+CRNN	CE	lattice	83.3	75.2	24.0	24.1
		50 best			23.9	24.1
+FRNN	CE	lattice	81.0	71.7	24.0	23.9
		50 best			23.9	24.0
	VR	lattice	80.4	71.6	23.9	24.0
		50 best			23.9	23.9
	NCE	lattice	81.1	72.8	24.1	24.1
		50 best			24.0	24.1

- RNNLMs give significant improvement over 3-gram LM
- F-RNNLMs are slightly better than C-RNNLMs
- F-RNNLMs trained by CE, VR and NCE give comparable performance

Experiments on 14M (AMI+Fisher) data

LM Type	Re score	PPL dev	PPL eval	WER dev	WER eval
3g	-	84.5	79.6	24.2	24.7
4g	lattice	80.3	76.3	23.7	24.1
+CRNN	lattice	70.5	67.5	22.4	22.5
	50 best			22.4	22.6
+FRNN	lattice	69.8	67.0	22.0	22.3
	50 best			22.2	22.5

- Similar trend observed on 14M data
- RNNLMs give significant performance improvement over n-gram LM
- F-RNNLMs are slightly better than C-RNNLMs
- Lattice (6g approximation) and N-best rescoring give comparable performance

Experiments on 14M data using various criteria

- Data shuffled for training of RNNLMs
  - give slight performance gain

- N-Best results reported

Train Crit	PPL		WER	
	dev	eval	dev	eval
CE	67.5	63.9	22.1	22.4
VR	68.0	64.4	22.1	22.4
NCE	68.5	65.1	22.1	22.4

- F-RNNLMs trained with CE, VR, NCE give comparable performance

Training and testing speed of RNNLMs

Toolkit	Train Crit	Train Speed(kw/s)	Test (CPU) Speed(kw/s)
RNNLM	CE	0.45	6.0
CUED-RNNLM	CE	11.5	0.32
	VR	11.5	15.3
	NCE	20.5	15.3

- CUED-RNNLM is much faster than RNNLM Toolkit from Mikolov
- NCE almost double train speed compared with VR and CE
- VR and NCE are much faster than CE due to unnormalized probability in test

Train Speed (kw/s) against number of hidden nodes

Toolkit	# Hidden node				
	128	256	512	1024	2048
RNNLM	4.1	1.7	0.45	0.095	0.012
CUED-RNNLM	19.8	14.2	11.5	6.6	3.7

- RNNLM Toolkit slow down quickly with the increase of hidden layer

- CUED-RNNLM is more suitable for training of RNNLM with large model size

Toolkit Download and Future Work

- Available at http://mi.eng.cam.ac.uk/projects/cued-rnnlm/
  - Source code (implemented by C++)
  - Document
  - Lattice conversion tool
  - AMI recipe
- License: BSD license
- Future work
  - CTS recipe
  - LSTM based RNNLM
  - Bidirectional RNNLM

Thanks !
Q & A