Efficient Training And Decoding Methods For Recurrent Neural Network Language Models

Xunying Liu
CUED NST Team

May 21th 2014

Cambridge University Engineering Department

Outline

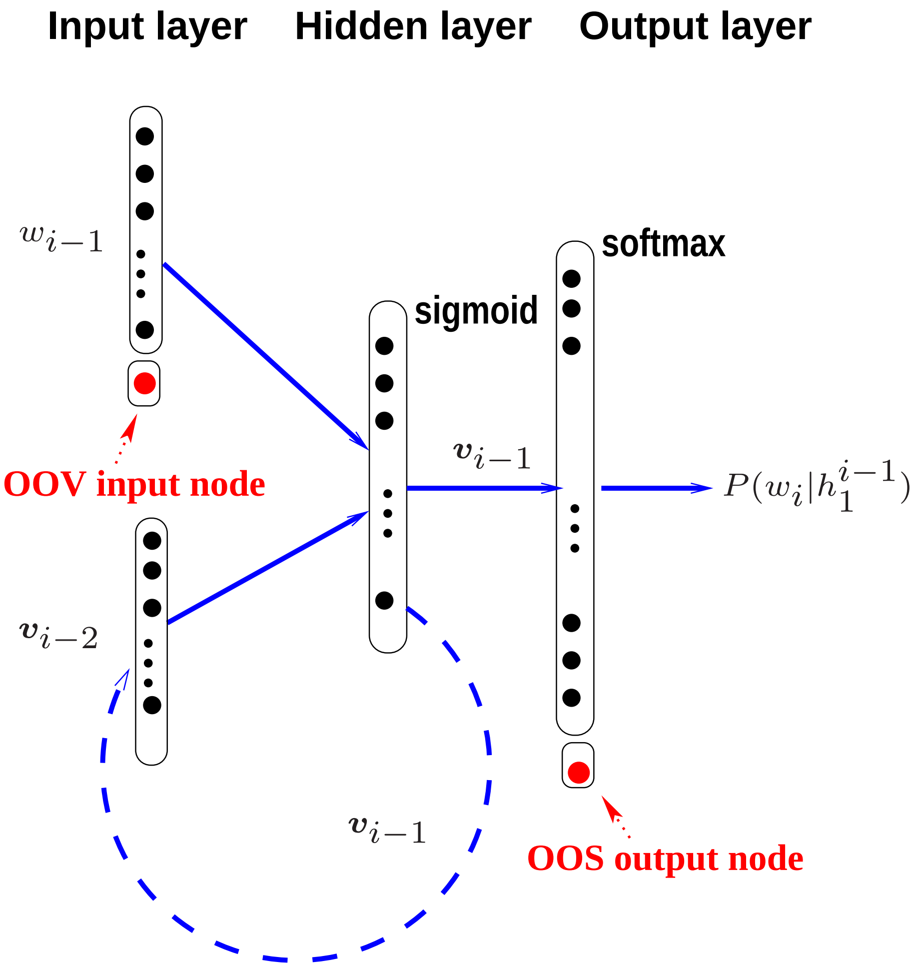
- **This talk summarizes recent research on improving efficiency of using recurrent neural network language models for speech recognition.**
- **Main objectives to achieve under NST:**
  - improving basic learning techniques for language models;
  - improving coverage and generalization of language models;
  - improving efficiency, rapidly deployable for a new domain or situation.
- **Experimental results on state-of-the-art LVCSR task:**
  - conversational telephone speech transcription.

Introduction

- **RNNLMs increasingly popular for ASR language modelling:**
  - strong generalization using continuous vector encoding of full histories;
  - internally clustering longer and potentially variable length histories;
  - more powerful modelling of history contexts than feedforward NNLMs;
  - improvements over back-off LMs and feedforward NNLMs widely reported.
- **Two important issues limiting their application:**
  - training is expensive, problematic when trained on large amounts of data;
  - difficult to use in full search/lattice rescoring, normally rescoring N-best.
- **This work aims at:**
  - **improving RNNLMs training speed and scalability on large data sets;**
  - **deriving efficient lattice rescoring/generation approaches.**

Recurrent Neural Network Language Models

Traditional RNNLMs with an unclustered, full output layer (F-RNNLMs)

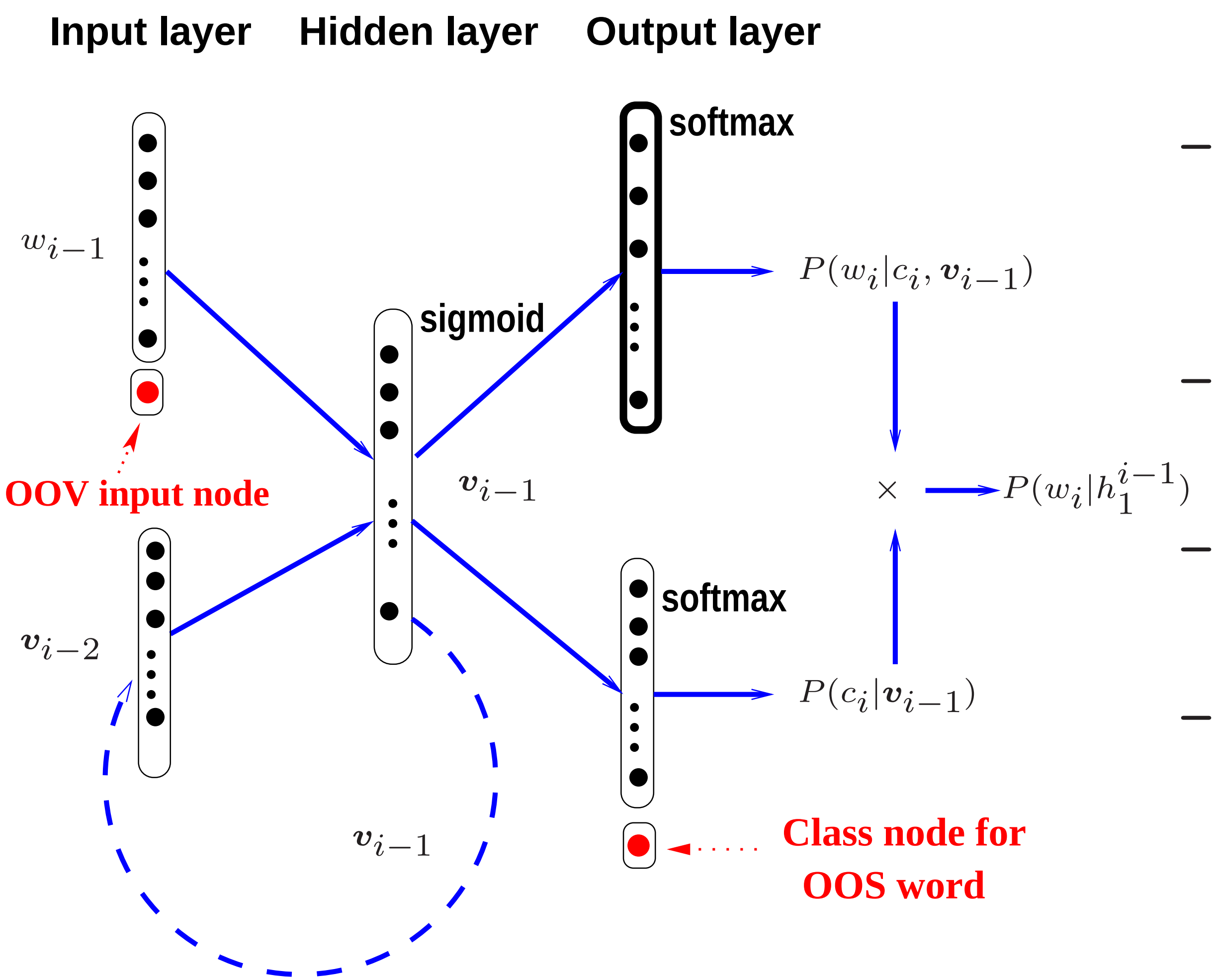


- 1-of-k coding of most recent word;
- recurrent vector represents remaining context;
- sigmoid activation used for hidden layer and softmax activation for output layer;
- hidden layer output fed back into the input layer.
- back propagation through time based training method.

Training F-RNNLMs is very expensive !

Class-based Recurrent Neural Network LMs

- **RNNLM architecture with a class based output layer (C-RNNLMs).**



- inspired by earlier work on class-based output layer for feedforward NNLMs;
- words in the output layer vocabulary assigned to classes;
- LM probabilities factorized into two individual terms;
- back propagation through time based training also used.

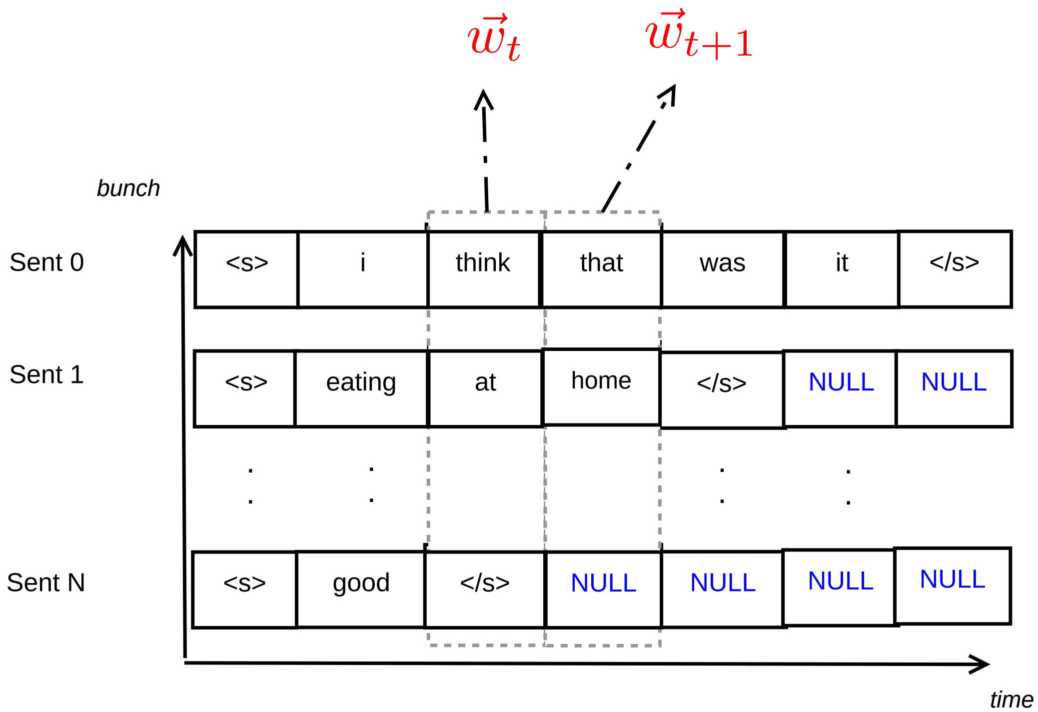
Class-based Recurrent Neural Network LMs (cont)

- **#classes and #words in classes smaller than full output vocab size:**
  - using class-based output layer significantly improves training speed;
  - training speed up to 15 time reported;
- **But C-RNNLMs also introduce performance sensitivity to word classing:**
  - can lead to performance degradation against F-RNNLMs.
- **CPU-based training, still slow on large data sets.**
- **Using irregular sized class specific output layer submatrices:**
  - complicates parallelization algorithms and limits potential speed up.
- **Our approach:**

GPU-based parallelized training for non-class based F-RNNLMs

Standard Bunch/Minibatch Mode RNNLM Training

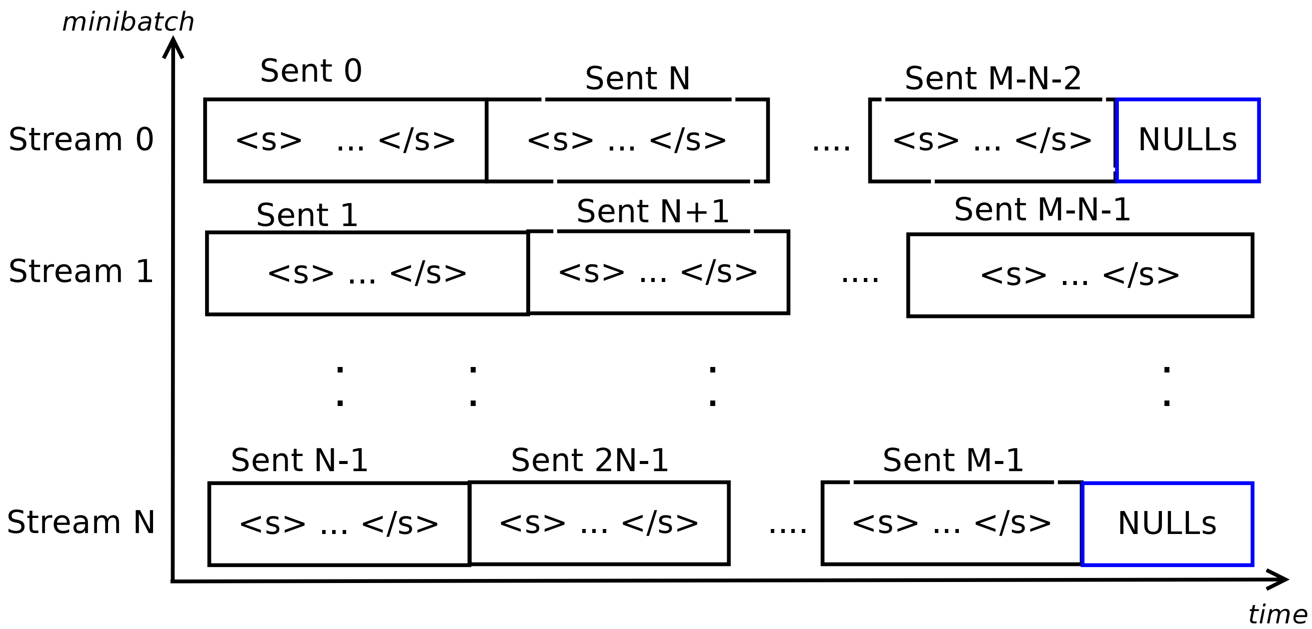
- **Previously used for parallelized training of feedforward NNLMs:**
  - multiple n-grams propagated through network without updating eights;
  - intermediate gradient statistics independently generated;
  - before being accumulated and used for updating the weight parameters.
- **RNNLM probabilities are inter-dependent within each sentence:**



- **sentence level bunch used;**
- formed by aligning sentences and inserting NULL tokens at end of shorter sentences;
- sentence length variation leads to synchronization overhead, limits speed improvements from parallelization;

Improved Bunch Mode RNNLM Training: Sentence Splicing

- **Bunch (minibatch) now filled with spliced/joined sentences:**
  - sentence boundaries marked to reset recurrent history vector as required;
  - more comparable in length, insertion of NULL token minimized;
  - reduces synchronization overhead and improves efficiency in parallelization.



- **Efficient Implementation for Nvidia GeForce GTX TITAN GPU.**

Experiments on Conversational Telephone Speech

- **Adapted 2000 hour Fisher data trained PLP MPE acoustic models:**
  - 545M words (20M Fisher+525M UWWeb) trained back-off 4-gram LM;
  - 20M words Fisher trained RNNLMs, intplt with baseline 4-gram LM;
  - 58k vocab, 20k output shortlist; 200 classes, 500 hidden nodes used;
  - Intel Xeon E5 8 core CPU (C-RNN), Nvidia GeForce TITAN GPU (F-RNN).

model type	bunch size	#parm	speed (w/s)	time (hours)	PPL	WER
w4g	-	-	-	-	51.8	16.7
C-RNN	-	26.9M	0.37k	202.1	46.5	15.3
F-RNN	**128**	26.8M	**9.9k**	**7.5**	**46.3**	**15.2**

- **27 time speed up over CPU C-RNNLM baseline using bunch size 128:**
  - Small reductions in WER also obtained over class-based RNNLMs.
- **Doubled model size, training on 500M words takes 2 weeks.**

Decoding Using Recurrent Neural Network Language Models

- **RNNLMs' model full histories - has practical implications in decoding:**
  - RNNLMs provide no explicit sharing of different histories like n-grams LMs;
  - difficult to use in full decoding/lattice rescoring, normally rescoring N-best;
  - limits gains from downstream applications favoring lattices, e.g. CN;
- **What has been done so far for efficient RNNLM lattice generation ?**
  - WFST based approximation and sampling approaches previously proposed;
  - but unable to produce performance comparable to std. N-best rescoring.
- **Our Approach: deriving alternative RNNLMs decoding methods**
  - by exploiting their intrinsic modelling characteristics to
  - **produce 1-best performance comparable to RNNLM N-best rescoring;**
  - **produce lattice representation suitable for ConfScore/CN decoding.**

History Context Clustering For RNNLMs

- **Efficient use of LMs in decoding requires history clustering:**
  - shared histories represented by a finite number of context dependent states;
  - truncated history of $N-1$ words for back-off LMs and feedforward NNLMs.

$$\Psi_{\text{NG}}(h_1^{i-1}) \;=\; h_{i-N+1}^{i-1} \;=\; <w_{i-1},\ldots,w_{i-N+1}>$$

- **RNNLMs encode full history $h_1^{i-1} = <w_{i-1},\ldots,w_1>$ as an ordered pair:**

$$\Psi_{\text{RNN}}(h_1^{i-1}) \;=\; h_1^{i-1} \;=\; <w_{i-1},\boldsymbol{v}_{i-2}>$$

  - number of context states grow exponentially as search space is widened;
  - **history clustering methods required to share RNNLM context states.**

History Context Clustering For RNNLMs (cont)

- **RNNLMs exploit two modelling characteristics to acquire generalization:**
- **Diminishing effect of most distant contexts on RNNLM probs:**
  - full histories overlapped in recent contexts share similar distribution;
  - possible to approx. RNNLMs using truncated histories of sufficient length.
- **Internally cluster full histories via vector space similarity:**
  - possible to share RNNLM probs using history vector distance.
- **Motivated by these characteristics:**
  - two history clustering schemes proposed;
  - to derive suitable finite state approximations for RNNLMs.

History Context Clustering For RNNLMs (cont)

- **n-gram history clustering: considers fixed number of most recent words**
  - intuitive clustering, same context states as comparable feedforward NNLMs in decoding;
  - shared RNNLM probabilities computed on-the-fly by request and cached;
- **History vector clustering: considers previous word + history vectors' Euclidean distance**
  - generic approach applicable for both feedforward/recurrent NNLMs;
  - distance beam adjusts trade-off between precision and compactness of RNNLM state representation;
- **Both implemented for CU-HTK lattice processing tools:**
  - generic on-the-fly lattice expansion algorithm suitable for back-off n-gram LMs, feedforward NNLMs, recurrent NNLMs and their interpolated forms.

Experiments on Conversational Telephone Speech

LM	dev04 1-best	dev04 CN	LatDensity (Arcs/Sec)
w4g+rnn.50best	15.4	15.4	188(97†)
w4g+rnn.10000best	15.3	15.0	32277(10212†)
w4g+rnn.sample1G.4g	16.2	15.9	462
w4g+rnn.approx6g	**15.4**	**15.0**	**3025**
w4g+rnn.hvd (γ=0.00050)	**15.4**	**15.0**	**2818**

- **1-best/CN performance comparable to 10k-best rescoring obtained by:**
  - using 6-gram (5 words truncated history) approximation;
  - or setting history vector beam $\gamma = 0.0005$.
- **Over 70% more compact than prefix tree structured† 10k-best;**
- **WER reductions of 0.9% abs. (5.6% rel.) over sampling approximation.**

Conclusion and Future Work

- **Efficient GPU-based bunch mode full RNNLM training investigated:**
  - 27 time training speed up against standard CPU-based RNNLM toolkit;
  - improvements in perplexity and recognition performance also obtained.
- **Two efficient lattice rescoring methods for RNNLMs proposed:**
  - 1-best and CN performance comparable with a 10k-best RNNLM rescoring;
  - consistent gains from CN decoding on RNNLM rescored lattices;
  - compact lattice representation produced, over 70% compression in size.
- **Future research will focus on:**
  - improving word classing and training parallelization for class-based RNNLMs.
  - improving history clustering and efficiency in lattice rescoring using NNLMs.