Statistical Language Models Based on Neural Networks

Tomáš Mikolov

Speech@FIT, Brno University of Technology, Czech Republic

Google, Mountain View, 2nd April 2012

Overview

- Motivation
- Neural Network Based Language Models
- Training Algorithm
  - Recurrent Neural Network
  - Classes
  - Maximum Entropy Language Model
- Empirical Results:
  - Penn Treebank Corpus
  - Wall Street Journal Speech Recognition
  - NIST RT04 Broadcast News Speech Recognition
- Generating Text with RNNs
- Additional Experiments
- Conclusion: Finally Beyond N-grams?

Motivation

- Statistical language models assign probabilities to word sequences
- For a good model of language, meaningful sentences should be more likely than the ambiguous ones
- Language modeling is an artificial intelligence problem

Motivation - Turing Test

- The famous Turing test can be in principle seen as a language modeling problem
- Given the history of conversation, a good language model should assign high probability to correct responses

- Example:
  $P(Monday|What\ day\ of\ week\ is\ today?) =?$
  $P(red|What\ is\ the\ color\ of\ roses?) =?$
  or more as a language modeling problem:
  $P(red|The\ color\ of\ roses\ is) =?$

Motivation - N-grams

- How to obtain the ”good language model”?
- Simple solution - N-grams: $P(w|h) = \frac{C(h,w)}{C(h)}$
- We count how many times the word w appeared in the context h, and normalize by all observations of h

Motivation - Limitations of N-grams

- Many histories h are similar, but n-grams assume exact match of h
- Practically, n-grams have problems with representing patterns over more than a few words
- With increasing order of the n-gram model, the number of possible parameters increases **exponentially**
- There will be never enough of the training data to estimate parameters of high-order N-gram models

Neural Network Based Language Models

- The sparse history h is projected into some continuous low-dimensional space, where similar histories get clustered
- Thanks to parameter sharing among similar histories, the model is more robust: less parameters have to be estimated from the training data

Model Description - Feedforward NNLM

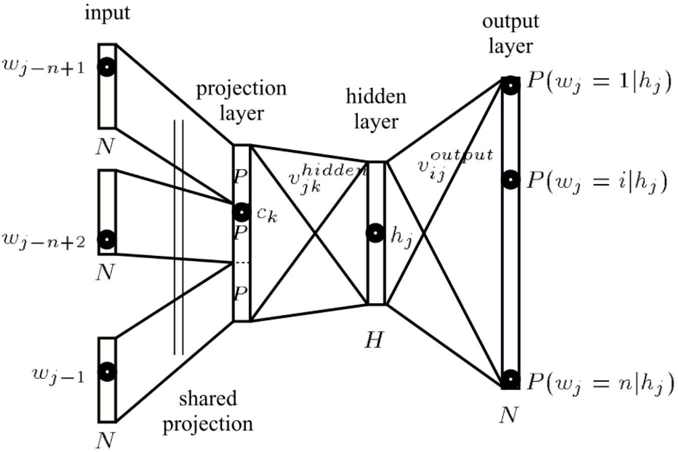
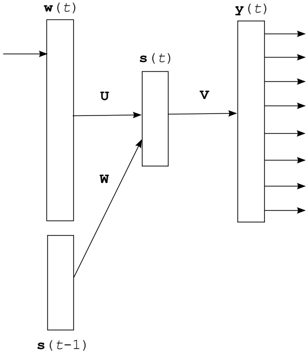


Figure: *Feedforward neural network based LM used by Y. Bengio and H. Schwenk*

Model description - recurrent NNLM



- Input layer $\mathbf{w}$ and output layer $\mathbf{y}$ have the same dimensionality as the vocabulary (10K - 200K)
- Hidden layer $\mathbf{s}$ is orders of magnitude smaller (50 - 1000 neurons)
- $\mathbf{U}$ is the matrix of weights between input and hidden layer, $\mathbf{V}$ is the matrix of weights between hidden and output layer
- Without the recurrent weights $\mathbf{W}$, this model would be a bigram neural network language model

The output values from neurons in the hidden and output layers are computed as follows:

$$\mathbf{s}(t) = f\left(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1)\right) \tag{1}$$

$$\mathbf{y}(t) = g\left(\mathbf{V}\mathbf{s}(t)\right), \tag{2}$$

where $f(z)$ and $g(z)$ are sigmoid and softmax activation functions (the softmax function in the output layer is used to ensure that the outputs form a valid probability distribution, i.e. all outputs are greater than 0 and their sum is 1):

$$f(z) = \frac{1}{1+e^{-z}}, \quad g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{3}$$

Training of RNNLM

- The training is performed using Stochastic Gradient Descent (SGD)
- We go through all the training data iteratively, and update the weight matrices $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$ online (after processing every word)
- Training is performed in several epochs (usually 5-10)

Training of RNNLM

Gradient of the error vector in the output layer $\mathbf{e_o}(t)$ is computed using a cross entropy criterion:

$$\mathbf{e}_o(t) = \mathbf{d}(t) - \mathbf{y}(t) \tag{4}$$

where $\mathbf{d}(t)$ is a target vector that represents the word $\mathbf{w}(t+1)$ (encoded as 1-of-V vector).

Training of RNNLM

Weights $\mathbf{V}$ between the hidden layer $\mathbf{s}(t)$ and the output layer $\mathbf{y}(t)$ are updated as

$$\mathbf{V}(t{+}1) = \mathbf{V}(t) + \mathbf{s}(t)\mathbf{e}_o(t)^T \alpha, \tag{5}$$

where α is the learning rate.

Training of RNNLM

Next, gradients of errors are propagated from the output layer to the hidden layer

$$\mathbf{e}_h(t) = d_h\left(\mathbf{e}_o(t)^T\mathbf{V}, t\right), \tag{6}$$

where the error vector is obtained using function $d_h()$ that is applied element-wise

$$d_{hj}(x,t) = x s_j(t)(1 - s_j(t)). \tag{7}$$

Training of RNNLM

Weights $\mathbf{U}$ between the input layer $\mathbf{w}(t)$ and the hidden layer $\mathbf{s}(t)$ are then updated as

$$\mathbf{U}(t{+}1) = \mathbf{U}(t) + \mathbf{w}(t)\mathbf{e}_h(t)^T\alpha. \tag{8}$$

Note that only one neuron is active at a given time in the input vector $\mathbf{w}(t)$. As can be seen from the equation 8, the weight change for neurons with zero activation is none, thus the computation can be speeded up by updating weights that correspond just to the active input neuron.

Training of RNNLM - Backpropagation Through Time

- The recurrent weights $\mathbf{W}$ are updated by unfolding them in time and training the network as a deep feedforward neural network.
- The process of propagating errors back through the recurrent weights is called Backpropagation Through Time (BPTT).

Training of RNNLM - Backpropagation Through Time

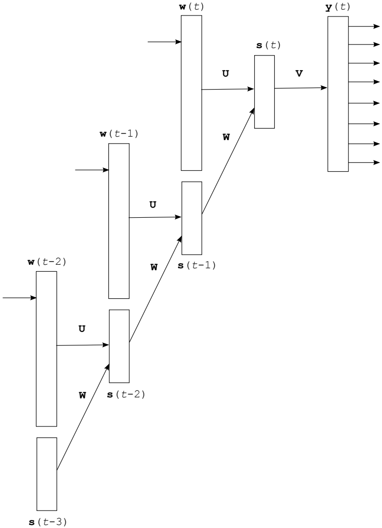


Figure: Recurrent neural network unfolded as a deep feedforward network, here for 3 time steps back in time.

Error propagation is done recursively as follows (note that the algorithm requires the states of the hidden layer from the previous time steps to be stored):

$$\mathbf{e}_h(t-\tau-1) = d_h\left(\mathbf{e}_h(t-\tau)^T\mathbf{W}, t-\tau-1\right). \tag{9}$$

The unfolding can be applied for as many time steps as many training examples were already seen, however the error gradients quickly **vanish** as they get backpropagated in time (in rare cases the errors can **explode**), so several steps of unfolding are sufficient (this is sometimes referred to as *truncated BPTT*).

Training of RNNLM - Backpropagation Through Time

The recurrent weights $\mathbf{W}$ are updated as

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \sum_{z=0}^{T} \mathbf{s}(t-z-1)\mathbf{e}_h(t-z)^T \alpha. \tag{10}$$

Note that the matrix $\mathbf{W}$ is changed in one update at once, and not during backpropagation of errors.

It is more computationally efficient to unfold the network after processing several training examples, so that the training complexity does not increase linearly with the number of time steps T for which the network is unfolded in time.

Training of RNNLM - Backpropagation Through Time

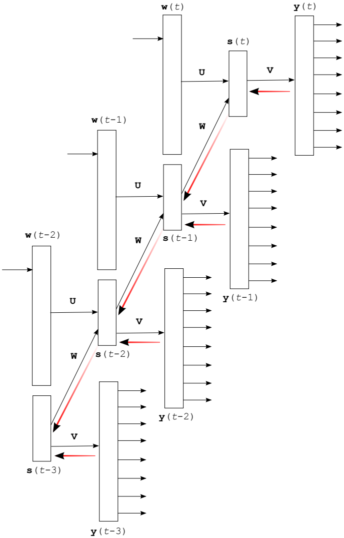


Figure: Example of batch mode training. Red arrows indicate how the gradients are propagated through the unfolded recurrent neural network.

Extensions: Classes

- Computing full probability distribution over all V words can be very complex, as V can easily be more than 100K.
- We can instead do:
  - Assign all words from V to a single class
  - Compute probability distribution over all classes
  - Compute probability distribution over words that belong to the specific class
- Assignment of words to classes can be trivial: we can use frequency binning.

Extensions: Classes

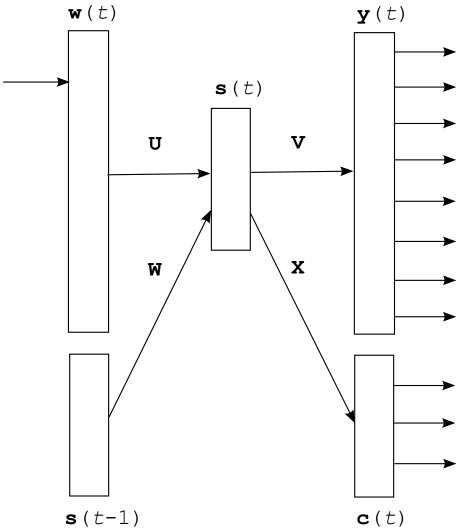


Figure: Factorization of the output layer, $c(t)$ is the class layer.

Extensions: Classes

- By using simple classes, we can achieve speedups on large data sets more than 100 times.
- We lose a bit of accuracy of the model (usually 5-10% in perplexity).

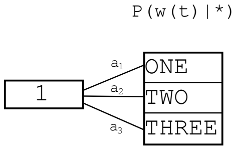
Joint Training With Maximum Entropy Model

- With increasing amount of the training data, we need to increase size of the hidden layer to obtain good performance (will be shown later).
- By joint training of RNNLM with a maximum entropy model, we can afford to have much smaller hidden layers.
- The ME model can be seen as a direct weight matrix between the input and the output layers (for the bigram case).

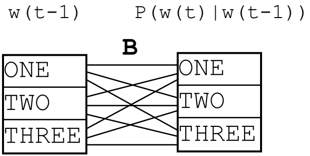
- The jointly trained RNN and ME models is further denoted as the RNNME architecture.

Hash Based Maximum Entropy Model

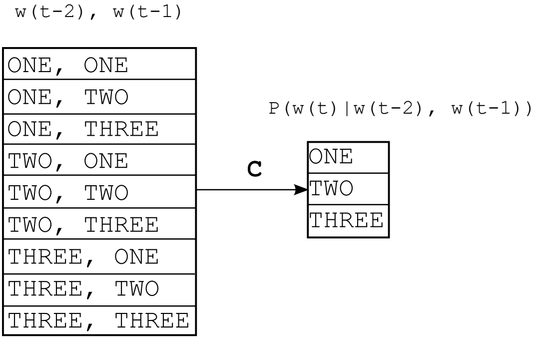
Assume a vocabulary V with three words, V=(ONE, TWO, THREE); maximum entropy model with unigram features:



Maximum entropy model with bigram features $\mathbf{B}$:



Maximum entropy model with trigram features $\mathbf{C}$:



Hash Based Maximum Entropy Model

- A maximum entropy model with full n-gram features has V^n parameters.
- To reduce memory complexity, we can map the $n-1$ dimensional histories to a single dimensional array using a hash function.
- The frequent features will dominate the values in the hash array.
- With small hash, the model will behave as pruned n-gram model.

- Such model can be easily trained by SGD, thus we can train it as a part of the neural net LM.

Empirical Results

- Penn Treebank
  - Comparison of advanced language modeling techniques
  - Combination
- Wall Street Journal
  - JHU setup
  - Kaldi setup
- NIST RT04 Broadcast News speech recognition
- Additional experiments: machine translation, text compression

Penn Treebank

- We have used the Penn Treebank Corpus, with the same vocabulary and data division as other researchers:
  - Sections 0-20: training data, 930K tokens
  - Sections 21-22: validation data, 74K tokens
  - Sections 23-24: test data, 82K tokens
  - Vocabulary size: 10K

Penn Treebank - Comparison

Model	Perplexity			Entropy reduction over baseline	
	individual	+KN5	+KN5+cache	KN5	KN5+cache
3-gram, Good-Turing smoothing (GT3)	165.2	-	-	-	-
5-gram, Good-Turing smoothing (GT5)	162.3	-	-	-	-
3-gram, Kneser-Ney smoothing (KN3)	148.3	-	-	-	-
5-gram, Kneser-Ney smoothing (KN5)	**141.2**	-	-	-	-
5-gram, Kneser-Ney smoothing + cache	**125.7**	-	-	-	-
PAQ8o10t	131.1	-	-	-	-
Maximum entropy 5-gram model	142.1	138.7	124.5	0.4%	0.2%
Random clusterings LM	170.1	126.3	115.6	2.3%	1.7%
Random forest LM	131.9	131.3	117.5	1.5%	1.4%
Structured LM	146.1	125.5	114.4	2.4%	1.9%
Within and across sentence boundary LM	116.6	110.0	108.7	5.0%	3.0%
Log-bilinear LM	144.5	115.2	105.8	4.1%	3.6%
Feedforward neural network LM	140.2	116.7	106.6	3.8%	3.4%
Syntactical neural network LM	131.3	110.0	101.5	5.0%	4.4%
Recurrent neural network LM	124.7	105.7	97.5	5.8%	5.3%
Dynamically evaluated RNNLM	123.2	102.7	98.0	6.4%	5.1%
Combination of static RNNLMs	102.1	95.5	89.4	7.9%	7.0%
Combination of dynamic RNNLMs	101.0	92.9	90.0	8.5%	6.9%

Penn Treebank - Comparison

Model	Perplexity			Entropy reduction over baseline	
	individual	+KN5	+KN5+cache	KN5	KN5+cache
3-gram, Good-Turing smoothing (GT3)	165.2	-	-	-	-
5-gram, Good-Turing smoothing (GT5)	162.3	-	-	-	-
3-gram, Kneser-Ney smoothing (KN3)	148.3	-	-	-	-
5-gram, Kneser-Ney smoothing (KN5)	**141.2**	-	-	-	-
5-gram, Kneser-Ney smoothing + cache	**125.7**	-	-	-	-
PAQ8o10t	131.1	-	-	-	-
Maximum entropy 5-gram model	142.1	138.7	124.5	0.4%	0.2%
Random clusterings LM	170.1	126.3	115.6	2.3%	1.7%
Random forest LM	131.9	131.3	117.5	1.5%	1.4%
Structured LM	146.1	125.5	114.4	2.4%	1.9%
Within and across sentence boundary LM	116.6	110.0	108.7	5.0%	3.0%
Log-bilinear LM	144.5	115.2	105.8	4.1%	3.6%
Feedforward neural network LM	140.2	116.7	106.6	3.8%	3.4%
Syntactical neural network LM	131.3	110.0	101.5	5.0%	4.4%
Recurrent neural network LM	124.7	105.7	97.5	5.8%	5.3%
Dynamically evaluated RNNLM	123.2	102.7	98.0	6.4%	5.1%
Combination of static RNNLMs	102.1	95.5	89.4	7.9%	7.0%
Combination of dynamic RNNLMs	101.0	92.9	90.0	8.5%	6.9%

Penn Treebank - Comparison

Model	Perplexity			Entropy reduction over baseline	
	individual	+KN5	+KN5+cache	KN5	KN5+cache
3-gram, Good-Turing smoothing (GT3)	165.2	-	-	-	-
5-gram, Good-Turing smoothing (GT5)	162.3	-	-	-	-
3-gram, Kneser-Ney smoothing (KN3)	148.3	-	-	-	-
5-gram, Kneser-Ney smoothing (KN5)	**141.2**	-	-	-	-
5-gram, Kneser-Ney smoothing + cache	**125.7**	-	-	-	-
PAQ8o10t	131.1	-	-	-	-
Maximum entropy 5-gram model	142.1	138.7	124.5	0.4%	0.2%
Random clusterings LM	170.1	126.3	115.6	2.3%	1.7%
Random forest LM	131.9	131.3	117.5	1.5%	1.4%
Structured LM	146.1	125.5	114.4	2.4%	1.9%
Within and across sentence boundary LM	116.6	110.0	108.7	5.0%	3.0%
Log-bilinear LM	144.5	115.2	105.8	4.1%	3.6%
Feedforward neural network LM	140.2	116.7	106.6	3.8%	3.4%
Syntactical neural network LM	131.3	110.0	101.5	5.0%	4.4%
Recurrent neural network LM	124.7	105.7	97.5	5.8%	5.3%
Dynamically evaluated RNNLM	123.2	102.7	98.0	6.4%	5.1%
Combination of static RNNLMs	102.1	95.5	89.4	7.9%	7.0%
Combination of dynamic RNNLMs	101.0	92.9	90.0	8.5%	6.9%

Penn Treebank - Comparison

Model	Perplexity			Entropy reduction over baseline	
	individual	+KN5	+KN5+cache	KN5	KN5+cache
3-gram, Good-Turing smoothing (GT3)	165.2	-	-	-	-
5-gram, Good-Turing smoothing (GT5)	162.3	-	-	-	-
3-gram, Kneser-Ney smoothing (KN3)	148.3	-	-	-	-
5-gram, Kneser-Ney smoothing (KN5)	**141.2**	-	-	-	-
5-gram, Kneser-Ney smoothing + cache	**125.7**	-	-	-	-
PAQ8o10t	131.1	-	-	-	-
Maximum entropy 5-gram model	142.1	138.7	124.5	0.4%	0.2%
Random clusterings LM	170.1	126.3	115.6	2.3%	1.7%
Random forest LM	131.9	131.3	117.5	1.5%	1.4%
Structured LM	146.1	125.5	114.4	2.4%	1.9%
Within and across sentence boundary LM	116.6	110.0	108.7	5.0%	3.0%
Log-bilinear LM	144.5	115.2	105.8	4.1%	3.6%
Feedforward neural network LM	140.2	116.7	106.6	3.8%	3.4%
Syntactical neural network LM	131.3	110.0	101.5	5.0%	4.4%
Recurrent neural network LM	124.7	105.7	97.5	5.8%	5.3%
Dynamically evaluated RNNLM	123.2	102.7	98.0	6.4%	5.1%
Combination of static RNNLMs	102.1	95.5	**89.4**	7.9%	7.0%
Combination of dynamic RNNLMs	101.0	**92.9**	90.0	8.5%	6.9%

Penn Treebank - Combination

Model	Weight	PPL
3-gram with Good-Turing smoothing (GT3)	0	165.2
5-gram with Kneser-Ney smoothing (KN5)	0	141.2
5-gram with Kneser-Ney smoothing + cache	0.0792	125.7
Maximum entropy model	0	142.1
Random clusterings LM	0	170.1
Random forest LM	0.1057	131.9
Structured LM	0.0196	146.1
Within and across sentence boundary LM	0.0838	116.6
Log-bilinear LM	0	144.5
Feedforward NNLM	0	140.2
Syntactical NNLM	0.0828	131.3
Combination of static RNNLMs	0.3231	102.1
Combination of adaptive RNNLMs	0.3058	101.0
ALL	1	**83.5**

Combination of Techniques (Joshua Goodman, 2001)

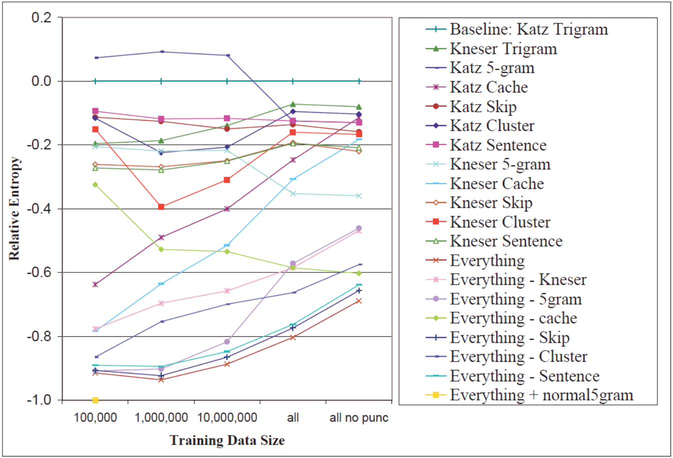
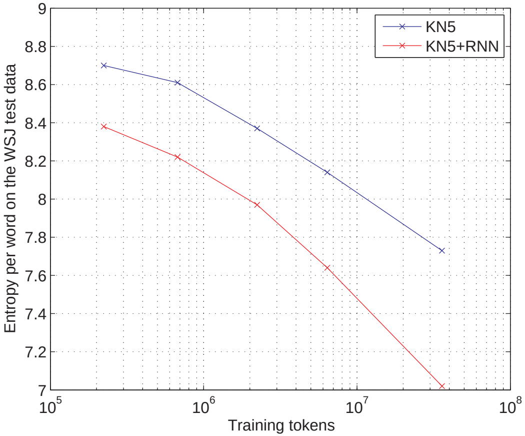


Figure from *"A bit of progress in language modeling, extended version"* (Goodman, 2001)

Empirical Evaluation - JHU WSJ Setup Description

- Setup from Johns Hopkins University (results are comparable to other techniques)
- Wall Street Journal: read speech, very clean (easy task for language modeling experiments)
- Simple decoder (not state of the art)
- 36M training tokens, 200K vocabulary
- WER results obtained by 100-best list rescoring

Improvements with Increasing Amount of Data



- The improvement obtained from a single RNN model over the best backoff model **increases** with more data!
- However, it is also needed to increase size of the hidden layer with more training data.

Improvements with Increasing Amount of Data

# words	PPL		WER		Improvement[%]	
	KN5	+RNN	KN5	+RNN	Entropy	WER
223K	415	333	-	-	3.7	-
675K	390	298	15.6	13.9	4.5	10.9
2233K	331	251	14.9	12.9	4.8	13.4
6.4M	283	200	13.6	11.7	6.1	14.0
36M	212	133	12.2	10.2	8.7	16.4

Comparison of Techniques - WSJ, JHU Setup

Model	Dev WER[%]	Eval WER[%]
Baseline - KN5	12.2	17.2
Discriminative LM	11.5	16.9
Joint structured LM	-	16.7
Static RNN	10.3	14.5
Static RNN + KN	10.2	14.5
Adapted RNN	9.7	14.2
Adapted RNN + KN	9.7	14.2
3 combined RNN LMs	**9.5**	**13.9**

Empirical evaluation - Kaldi WSJ setup description

- The same test sets as JHU setup, but lattices obtained with Kaldi speech recognition toolkit
- N-best lists were produced by Stefan Kombrink last summer, currently the best Kaldi baseline is much better
- 37M training tokens, 20K vocabulary
- WER results obtained by 1000-best list rescoring
- results obtained with RNNME models, with up to 4-gram features and size of hash 2G parameters

- Better repeatability of experiments than with the JHU setup

Empirical evaluation - Kaldi WSJ setup

Model	Perplexity		WER [%]	
	heldout	Eval 92	Eval 92	Eval 93
GT2	167	209	14.6	19.7
GT3	105	147	13.0	17.6
KN5	87	131	12.5	16.6
KN5 (no count cutoffs)	80	122	**12.0**	**16.6**
RNNME-0	90	129	12.4	17.3
RNNME-10	81	116	11.9	16.3
RNNME-80	70	100	10.4	14.9
RNNME-160	65	95	10.2	14.5
RNNME-320	62	93	9.8	14.2
RNNME-480	59	90	10.2	13.7
RNNME-640	59	89	9.6	14.4
combination of RNNME models	-	-	9.24	13.23
+ unsupervised adaptation	-	-	**9.15**	**13.11**

Empirical evaluation - Kaldi WSJ setup

Model	Perplexity		WER [%]	
	heldout	Eval 92	Eval 92	Eval 93
GT2	167	209	14.6	19.7
GT3	105	147	13.0	17.6
KN5	87	131	12.5	16.6
KN5 (no count cutoffs)	80	122	**12.0**	**16.6**
RNNME-0	90	129	12.4	17.3
RNNME-10	81	116	11.9	16.3
RNNME-80	70	100	10.4	14.9
RNNME-160	65	95	10.2	14.5
RNNME-320	62	93	9.8	14.2
RNNME-480	59	90	10.2	13.7
RNNME-640	59	89	9.6	14.4
combination of RNNME models	-	-	9.24	13.23
+ unsupervised adaptation	-	-	**9.15**	**13.11**

Results improve with larger hidden layer.

Evaluation - Broadcast News Speech Recognition

- NIST RT04 Broadcast News speech recognition task
- The baseline system is state-of-the-art setup from IBM based on Attila decoder: very well tuned, hard task
- 87K vocabulary size, 400M training tokens (10x more than WSJ setups)
- It has been reported by IBM that state of the art LM on this setup is a regularized class-based maxent model (called ”model M”)
- NNLMs have been reported to perform about the same as model M (about 0.6% absoulte WER reduction), but are computationally complex

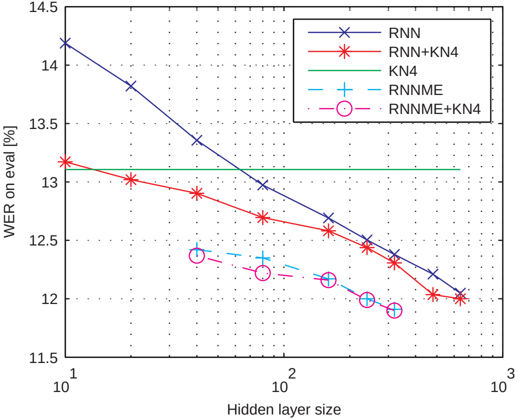
- We tried class based RNN and RNNME models...

Evaluation - Broadcast News Speech Recognition

Model	WER[%]	
	Single	Interpolated
KN4 (baseline)	13.11	13.11
model M	13.1	12.49
RNN-40	13.36	12.90
RNN-80	12.98	12.70
RNN-160	12.69	12.58
RNN-320	12.38	12.31
RNN-480	12.21	12.04
RNN-640	12.05	**12.00**
RNNME-0	13.21	12.99
RNNME-40	12.42	12.37
RNNME-80	12.35	12.22
RNNME-160	12.17	12.16
RNNME-320	11.91	**11.90**
3xRNN	-	**11.70**

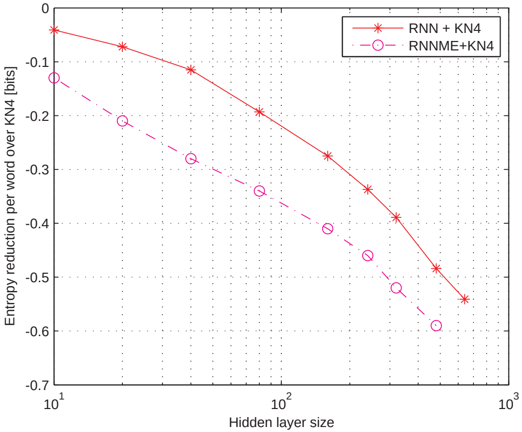
- Word error rate on the NIST RT04 evaluation set
- Still plenty of space for improvements! Adaptation, bigger models, combination of RNN and RNNME, ...
- Another myth broken: maxent model (aka ”model M”) is not more powerful than NNLMs!

Empirical Evaluation - Broadcast News Speech Recognition



- The improvements increase with more neurons in the hidden layer

Empirical Evaluation - Broadcast News Speech Recognition



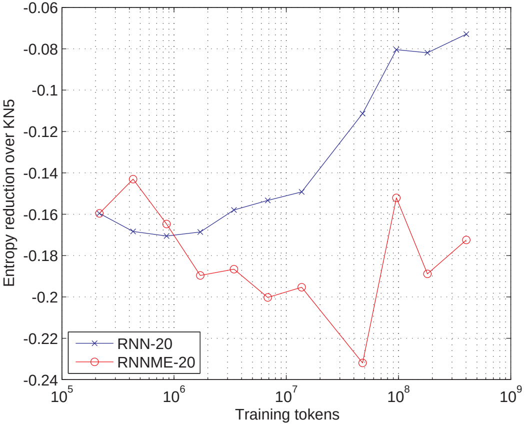
- Comparison of entropy improvements obtained from RNN and RNNME models over KN4 model

Empirical Evaluation - Entropy

Additional experiments to compare RNN and RNNME:

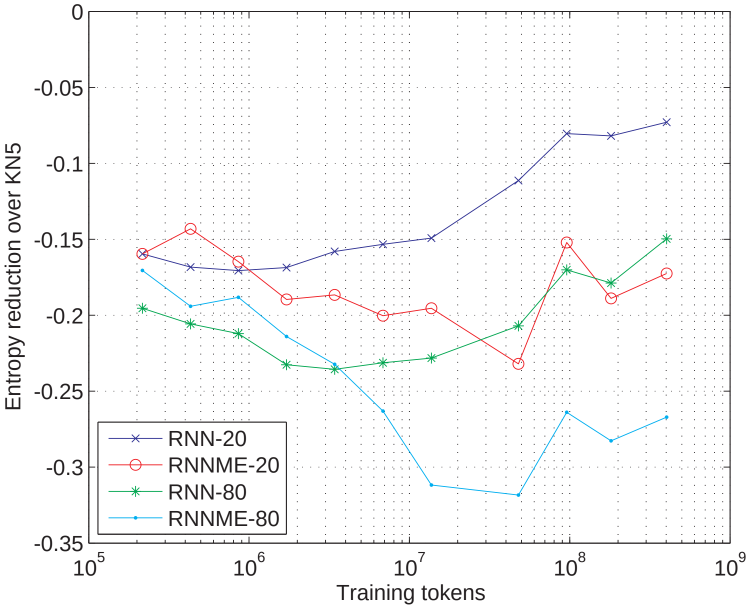
- Randomized order of sentences in the training data (to prevent adaptation)
- Comparison of entropy reductions over KN 5-gram model with no count cutoffs and no pruning

Empirical Evaluation - Entropy



- If hidden layer size is kept constant in the RNN model, the improvements seem to vanish with more data
- RNNME seems to be useful at large data sets

Empirical Evaluation - Entropy



Additional Experiments: Machine Translation

- Machine translation: very similar task as speech recognition (from the language modeling point of view)
- I performed the following experiments when visiting JHU at 2010
- Basic RNN models were used (no classes, no BPTT, no ME)
- Baseline systems were trained by Zhifei Li and Ziyuan Wang

Additional Experiments: Machine Translation

Table: *BLEU on IWSLT 2005 Machine Translation task, Chinese to English.*

Model	BLEU
baseline (n-gram)	48.7
300-best rescoring with RNNs	51.2

- About 400K training tokens, small task

Additional Experiments: Machine Translation

Table: *BLEU and NIST score on NIST MT 05 Machine Translation task, Chinese to English.*

Model	BLEU	NIST
baseline (n-gram)	33.0	9.03
1000-best rescoring with RNNs	34.7	9.19

- RNNs were trained on subset of the training data (about 17.5M training tokens), with limited vocabulary

Additional Experiments: Text Compression

Compressor	Size [MB]	Bits per character
original text file	1696.7	8.0
gzip -9	576.2	2.72
RNNME-0	273.0	1.29
PAQ8o10t -8	272.1	1.28
RNNME-40	263.5	1.24
RNNME-80	258.7	1.22
RNNME-200	256.5	1.21

- PAQ8o10t is state of the art compression program
- Data compressor = Predictor + Arithmetic coding

- Task: compression of normalized text data that were used in the NIST RT04 experiments
- Achieved entropy of English text 1.21 bpc is already lower than the upper bound 1.3 bpc estimated by Shannon
- Several tricks were used to obtain the results: multiple models with different learning rate, skip-gram features

Conclusion: Finally Beyond N-grams?

- Extensive experiments confirm that n-grams can be significantly beaten at many interesting tasks:
  - Penn Treebank: perplexity reduced from 141 to 79
  - WSJ: 21% - 23% relative reduction of WER
  - Broadcast News Speech Recognition: 11% relative reduction of WER
  - MT: 1.7 - 2.5 BLEU points
  - Text compression
- Experiments can be easily repeated using freely available RNNLM tool!
- But are we any closer to ”intelligent language models”?

Data sampled from 4-gram backoff model

OR STUDENT'S IS FROM TEETH PROSECUTORS DO FILLED WITH HER SOME BACKGROUND ON WHAT WAS GOING ON HERE
ALUMINUM CANS OF PEACE
PIPER SWEAT COLONEL SAYING HAVE ALREADY MADE LAW THAT WOULD PREVENT THE BACTERIA
DOWN FOR THE MOST OF IT IN NINETEEN SEVENTY EIGHT WHICH WAS ONE OF A NUMBER OF ISSUES INCLUDING CIVIL SUIT BY THIS TIME NEXT YEAR
CRYSTAL
FIRMLY AS A HERO OF MINE A PREVIEW THAT
THOMAS SEVENTY BODIES AND ASKING QUESTIONS MAYBE
ATTORNEY'S OFFICE THEATERS CUT ACROSS THE ELEVENTH AND SUPPORT THEM WITH ELLEN WISEST PULLING DATA GATHERING IN RESPONSE TO AN UNMITIGATED DISPOSITION CONTRACTORS AND
AND I'M VERY SORRY FOR THE DEATH OF HER SPOKESWOMAN
ONIONS THE FRESH CORN THANKSGIVING CONTROL WHEN I TALKED TO SAID THAT AND THEY THINK WHAT AT LEAST UNTIL AFTER
I'M UPSET SO WE INCORPORATED WITH DROPPING EXTRAORDINARY PHONED

Data sampled from RNN model

THANKS FOR COMING IN NEXT IN A COUPLE OF MINUTES
WHEN WE TAKE A LOOK AT OUR ACCOMPANYING STORY IMAGE
GUIDE WHY ARE ANY OF THOSE DETAILS BEING HEARD IN LONDON
BUT DEFENSE ATTORNEYS SAY THEY THOUGHT THE CONTACT WAS
NOT AIMED DAMAGING AT ANY SUSPECTS
THE UNITED NATIONS SECURITY COUNCIL IS NAMED TO WITHIN
TWO MOST OF IRAQI ELECTION OFFICIALS
IT IS THE MINIMUM TIME A TOTAL OF ONE DETERMINED TO
APPLY LIMITS TO THE FOREIGN MINISTERS WHO HAD MORE POWER
AND NOW THAN ANY MAN WOULD NAME A CABINET ORAL
FIND OUT HOW IMPORTANT HIS DIFFERENT RECOMMENDATION IS
TO MAKE WHAT THIS WHITE HOUSE WILL WILL TO BE ADDRESSED
ELAINE MATHEWS IS A POLITICAL CORRESPONDENT FOR THE
PRESIDENT'S FAMILY WHO FILED A SIMILAR NATIONWIDE
OPERATION THAT CAME IN A DEAL
THE WEIGHT OF THE CLINTON CERTAINLY OUTRAGED ALL
PALESTINIANS IN THE COUNTRY IS DESIGNED TO REVIVE THE
ISRAELI TALKS

WSJ-Kaldi rescoring

`5-gram:` **IN TOKYO FOREIGN EXCHANGE TRADING YESTERDAY THE UNIT INCREASED AGAINST THE DOLLAR**
`RNNLM:` **IN TOKYO FOREIGN EXCHANGE TRADING YESTERDAY THE YEN INCREASED AGAINST THE DOLLAR**

`5-gram:` **SOME CURRENCY TRADERS SAID THE UPWARD REVALUATION OF THE GERMAN MARK WASN'T BIG ENOUGH AND THAT THE MARKET MAY CONTINUE TO RISE**
`RNNLM:` **SOME CURRENCY TRADERS SAID THE UPWARD REVALUATION OF THE GERMAN MARKET WASN'T BIG ENOUGH AND THAT THE MARKET MAY CONTINUE TO RISE**

`5-gram:` **MEANWHILE QUESTIONS REMAIN WITHIN THE E. M. S. WEATHERED YESTERDAY'S REALIGNMENT WAS ONLY A TEMPORARY SOLUTION**
`RNNLM:` **MEANWHILE QUESTIONS REMAIN WITHIN THE E. M. S. WHETHER YESTERDAY'S REALIGNMENT WAS ONLY A TEMPORARY SOLUTION**

`5-gram:` **MR. PARNES FOLEY ALSO FOR THE FIRST TIME THE WIND WITH SUEZ'S PLANS FOR GENERALE DE BELGIQUE'S WAR**
`RNNLM:` **MR. PARNES SO LATE ALSO FOR THE FIRST TIME ALIGNED WITH SUEZ'S PLANS FOR GENERALE DE BELGIQUE'S WAR**

`5-gram:` **HE SAID THE GROUP WAS MARKET IN ITS STRUCTURE AND NO ONE HAD LEADERSHIP**
`RNNLM:` **HE SAID THE GROUP WAS ARCANE IN ITS STRUCTURE AND NO ONE HAD LEADERSHIP**

Conclusion: Finally Beyond N-grams?

- RNN LMs can generate much more meaningful text than n-gram models trained on the same data
- Many novel but meaningful sequences of words were generated

- RNN LMs are clearly better at modeling the language than n-grams
- However, many simple patterns in the language cannot be efficiently described even by RNNs...

Conclusion: Finally Beyond N-grams?

- Thanks to Yoshua Bengio, Lukas Burget, Sanjeev Khudanpur and Daniel Povey for help with the theory
- Thanks to Anoop Deoras, Stefan Kombrink, Hai Son Le and Ilya Sutskever for collaboration with the experiments

- Thanks for attention!