Language Modeling with Recurrent Neural Networks

Tomáš Mikolov,
Anoop Deoras, Stefan Kombrink,
Hai Son Le, Ilya Sutskever

Speech@FIT, Brno University of Technology, Czech Republic

5. 12. 2011

Overview

- Motivation
- Architectures of neural net language models
  - Classes in the output layer
  - Maximum entropy language model
- Comparison and combination of techniques on Penn Treebank, WSJ setups
- Broadcast News recognition (RT04)
- Data sampling
- Subword-based language models
- Compression experiments
- Conclusion

Motivation

- Language models assign probabilities to word sequences
- For a good language model, meaningful sentences should be more likely than incorrect ones
- Turing test can be seen as a language modeling problem: given previous conversation, find the most likely response
- Hutter prize: data compression contest - intelligence is about prediction and better prediction leads to better compression

- Language modeling is AI-complete

Introduction - examples

Language modeling is an artificial intelligence problem - examples:

$P(Tokyo|capital\ city\ of\ Japan\ is) =?$
$P(five|two\ and\ three\ are) =?$
$P(Friday|Today\ is\ Thursday.\ What\ is\ the\ day\ tomorrow?) =?$

- Models based on N-gram counts: we need to see correct 'answer' in the training data to assign high probability to it in any of these examples
- There is no intelligence or generalization in N-grams: these are simply efficient databases
- With huge amount of training data, N-grams will perform very well; but nowhere near to AI

- In short-term perspective, we might do well with N-gram models estimated from huge amount of data
- In long-term, sooner or later we will have to explore techniques that can learn more patterns from less training data

Model description - feedforward NN

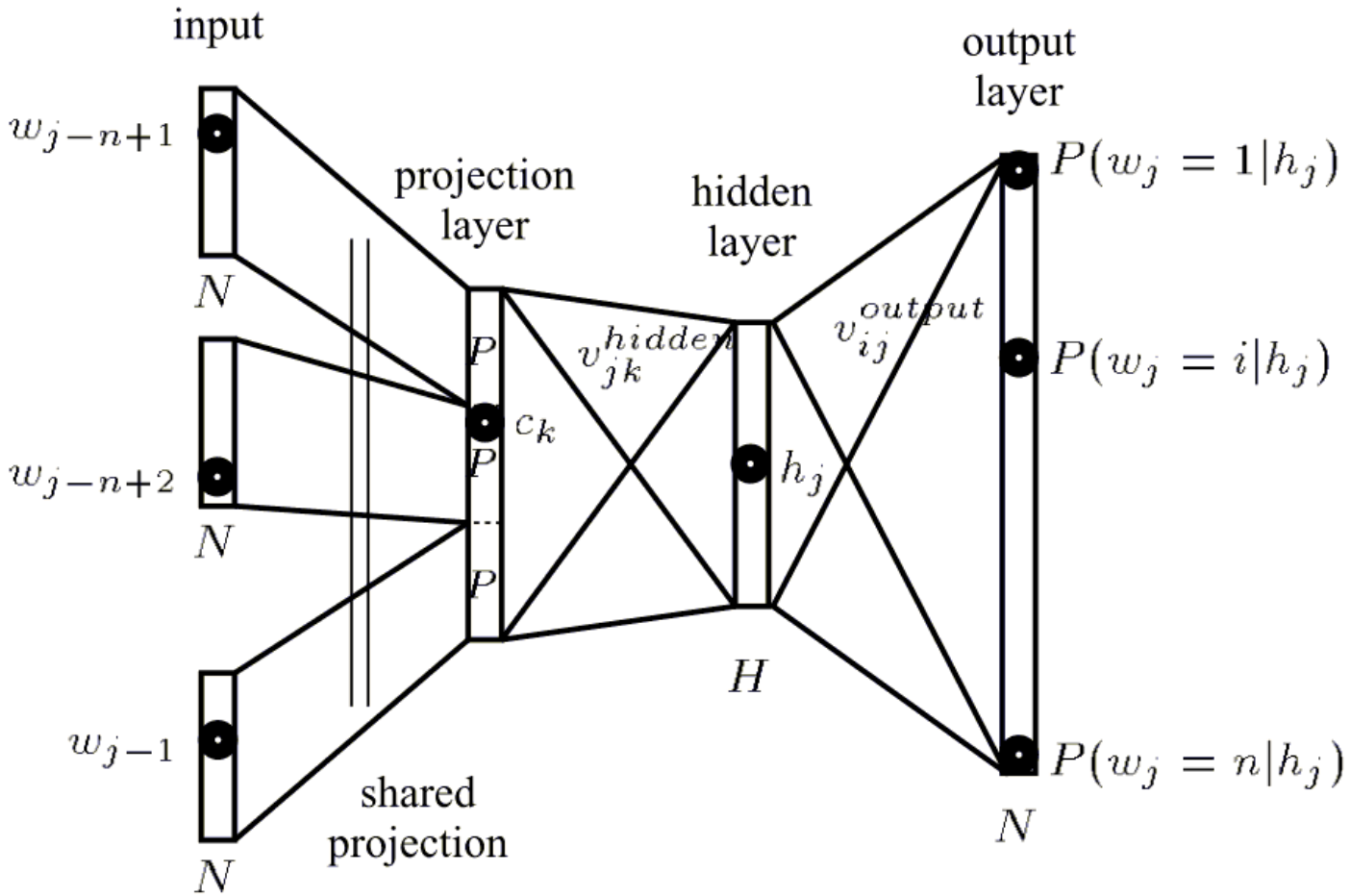
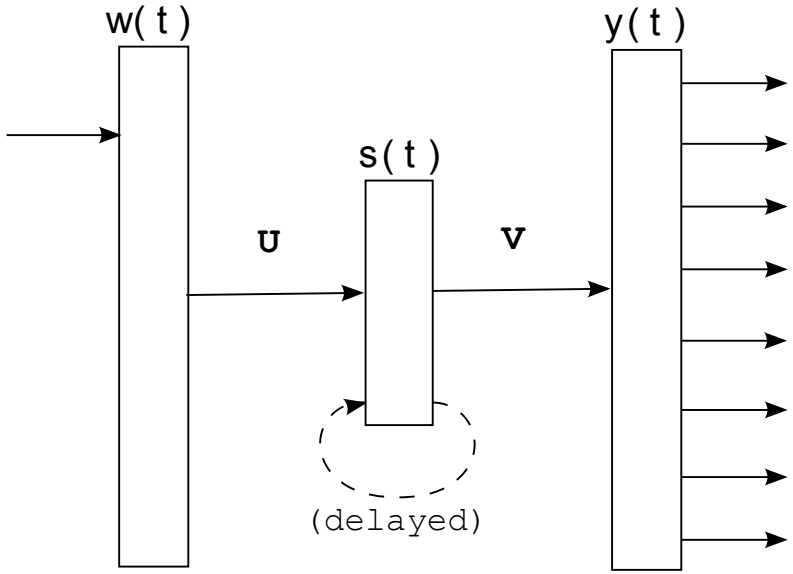


Figure: *Feedforward neural network based LM used by Y. Bengio and H. Schwenk*

Model description - recurrent NN

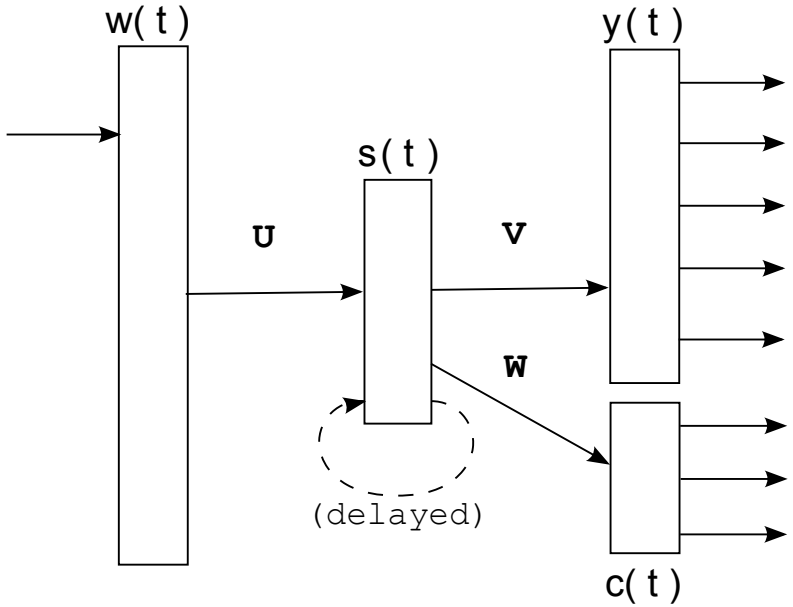


- Input layer $\mathbf{w}$ and output layer $\mathbf{y}$ have the same dimensionality as the vocabulary (10K - 200K)
- Hidden layer $\mathbf{s}$ is orders of magnitude smaller (50 - 1000 neurons)
- $\mathbf{U}$ is the matrix of weights between input and hidden layer, $\mathbf{V}$ is the matrix of weights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

Backpropagation through time

- Training of RNNs by normal backpropagation is not optimal (but for small data sets, even this works!)
- Backpropagation through time (BPTT) is an efficient algorithm for training recurrent neural networks
- BPTT works by unfolding the recurrent part of the network in time to obtain usual feedforward representation of the network; such deep network is then trained by backprop
- For on-line learning, ”truncated BPTT” is used (network is unfolded in time just for several time steps back)

Factorization of the output layer



$$P(w_i|history) = P(c_i|\mathbf{s}(t))P(w_i|c_i, \mathbf{s}(t)) \quad (1)$$

- Words are assigned to ”classes” based on their unigram frequency (very simple)
- First, class layer is evaluated; then, only words belonging to the predicted class are evaluated, instead of the whole output layer $\mathbf{y}$ [Goodman2001]
- Provides speedup in some cases more than $100\times$

Maximum entropy language model

Maximum entropy (ME) model has the following form:

$$P(w|h) = \frac{e^{\sum_{i=1}^{N} \lambda_i f_i(h,w)}}{\sum_w e^{\sum_{i=1}^{N} \lambda_i f_i(h,w)}} \tag{2}$$

where $\mathbf{f}$ is a set of features, $\boldsymbol{\lambda}$ is a set of weights and h is a history.

- Features are usually binary
- Most usual features are N-grams - for example, $f_1(h, w) = 1$ if $h = IT$ and $w = IS$, otherwise it is 0
- We learn the set of weights λ
- Maxent LM can be seen as neural net language model without the hidden layer
- We can train maxent LM by normal stochastic gradient descent

Maximum entropy language model

We can view the NNLM as

$$P(w|h) = \frac{e^{\sum_{i=1}^{N} \lambda_i f_i(s,w)}}{\sum_w e^{\sum_{i=1}^{N} \lambda_i f_i(s,w)}} \tag{3}$$

where s is a state of hidden layer. Thus, the main difference between ME model and NN model is that the features of NNLM are learned from the history, and are distributed (while features for ME language model are usually binary and local, like N-gram features).

Hash-based implementation of ME LM

- The full set of N-gram features for ME model has size V^N
- For $V = 100K$ and $N = 5$, this is too much
- The frequency of access to different features varies a lot: many features are never used
- We can thus map the large V^N space into much smaller using a hash function
- Using hash, the whole implementation of ME LM becomes trivial
- Drawback is high memory complexity

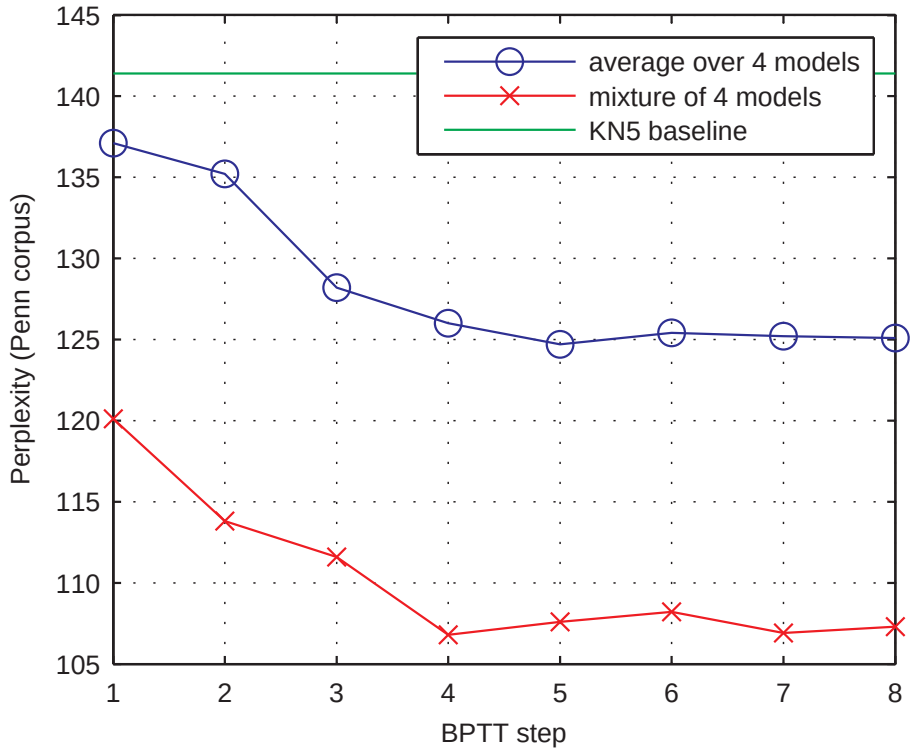
RNNME language model

- Maximum entropy models can be trained much faster - only N feature functions for N-gram ME model are active at any given time (while the hidden layer in NNLM can have size over 500 neurons)
- As both models have very similar form and both can be trained by on-line gradient descent, we can train them jointly
- We hope to reduce training time of the original NNLM, as the distributed part of the model can directly focus on discovering complementary information to N-gram features
- Thus, we can get very good performance with tiny hidden layers

Empirical evaluation - PTB setup description

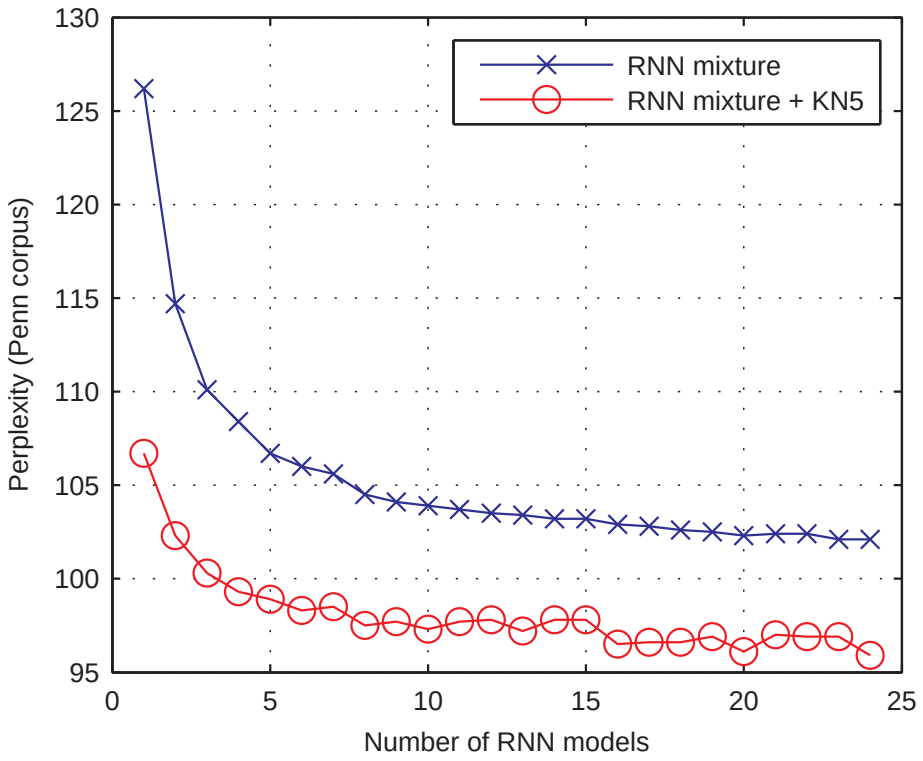
- We have used the Penn Treebank Corpus, with the same vocabulary and data division as other researchers:
  - Sections 0-20: training data, 930K tokens
  - Sections 21-22: validation data, 74K tokens
  - Sections 23-24: test data, 82K tokens
  - Vocabulary size: 10K

Importance of BPTT training



- Importance of BPTT training on Penn Corpus. BPTT=1 corresponds to standard backpropagation.

Combination of randomly initialized RNNs



- By linearly interpolating outputs from randomly initialized RNNs, we obtain better results

Comparison of different language modeling techniques

Model	Perplexity			Entropy reduction	
	individual	+KN5	+KN5+cache	over KN5	over KN5+cache
3-gram with Good-Turing smoothing (GT3)	165.2	-	-	-	-
5-gram with Good-Turing smoothing (GT5)	162.3	-	-	-	-
3-gram with Kneser-Ney smoothing (KN3)	148.3	-	-	-	-
5-gram with Kneser-Ney smoothing (KN5)	**141.2**	-	-	-	-
5-gram with Kneser-Ney smoothing + cache	**125.7**	-	-	-	-
Maximum entropy model with 5-gram features	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%
Adaptive 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 adaptive RNNLMs	101.0	92.9	90.0	8.5%	6.9%

Combination of different language modeling techniques

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

Speedup with different amount of classes

Classes	RNN	RNN+KN5	Min/epoch	Sec/test
30	134	112	12.8	8.8
100	136	114	9.1	5.6
1000	131	111	16.1	15.7
4000	127	108	44.4	57.8
Full	123	106	154	212

- Values around `sqrt(vocabulary size)` lead to the largest speed-ups

Combination of techniques (Joshua Goodman, 2001)

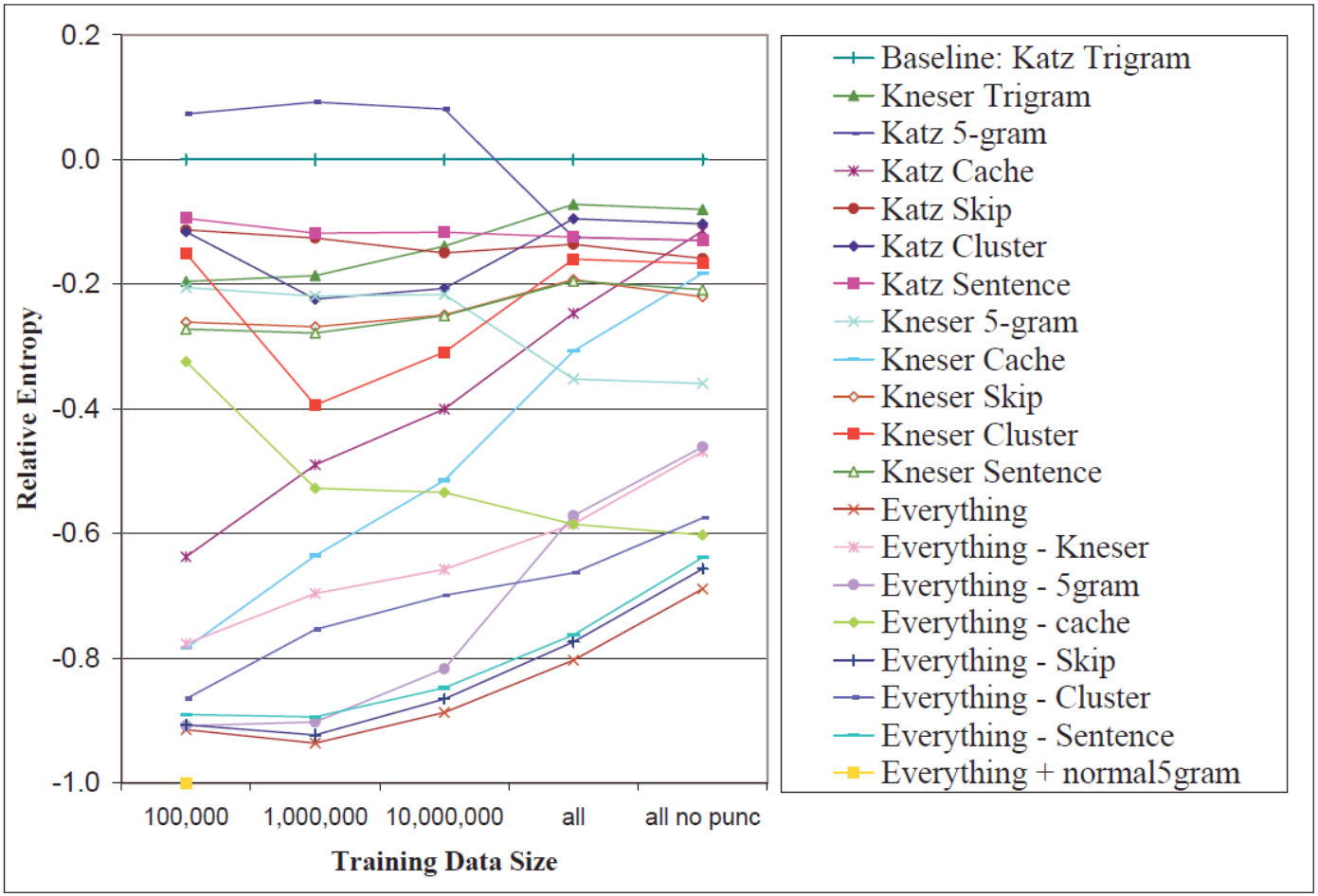
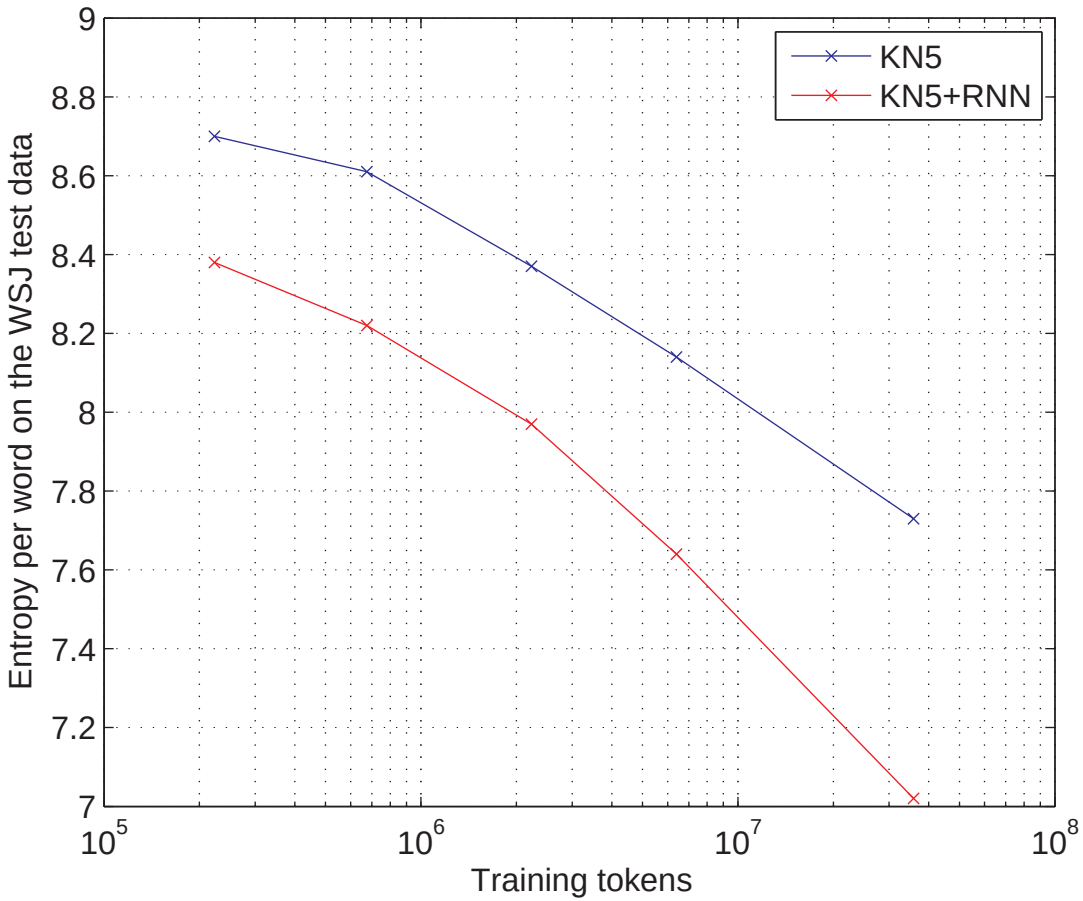


Figure from *"A bit of progress in language modeling, extended version"* (Goodman, 2001)

Empirical evaluation - WSJ setup description

- Wall Street Journal: read speech, very clean (easy task for language modeling experiments)
- old and simple decoder
- 36M words of training data, 200K vocabulary

Improvements with increasing amount of data - WSJ



- The improvement obtained from a single RNN model over the best backoff model **increases** with more data!

Improvements with increasing amount of data - WSJ

# 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

Model	Dev WER[%]	Eval WER[%]
Baseline - KN5	12.2	17.2
Discriminative LM	11.5	16.9
Joint structured LM	-	16.7
Static RNN	10.5	14.9
Static RNN + KN	10.2	14.6
Adapted RNN	9.8	14.5
Adapted RNN + KN	9.8	14.5
Both RNN	9.7	14.4
Both RNN + RNNME	**9.5**	**14.0**

Hai Son Le from LIMSI trained three feedforward NNLMs on this setup with even more parameters (1000 neurons in the hidden layer) and obtained 14.0% WER on the Evaluation set.

Empirical evaluation - Broadcast News 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
- 400M training tokens, 87K vocabulary size
- 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 0.5% WER improvement), but are computationally complex

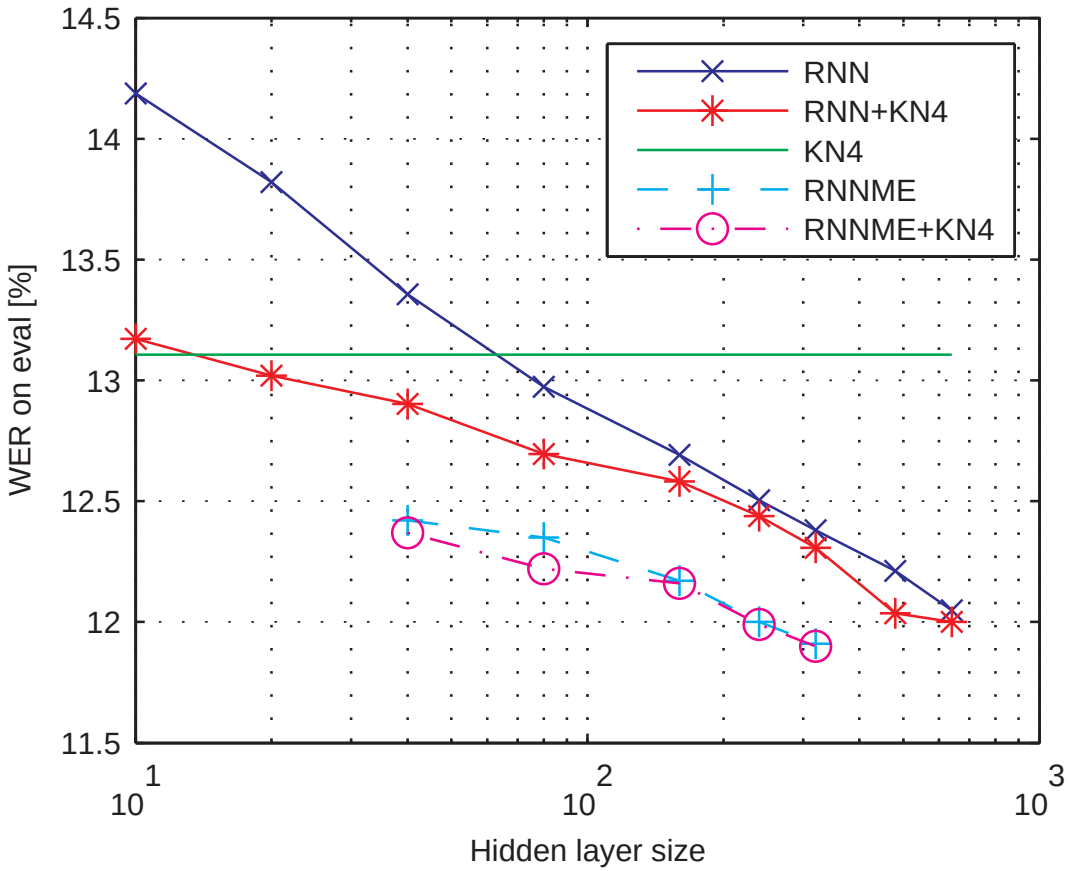
- We tried class based RNN and RNNME models...

Empirical evaluation - Broadcast News 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 recognition



- The improvements increase with more neurons in the hidden layer

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

Data sampling

- Sampling random sentences from RNN model can be actually quite useful: we can approximate RNN models by N-gram models that are built on top of the sampled data
- It is in principle the same as approximation of Turing machine with infinite state machine: the more data we sample, the closer the approximation will be
- In practice, we can get around 30% of improvement that RNN gives us by approximating it with N-gram model (that can be easily used directly by standard decoders)

Subword-based language models

- Using words as atomic units might look meaningful for English, but fails for many other languages (Czech, German, Russian, Arabic, Chinese, ...)
- The rate of novel words (so-called Out of vocabulary words - OOVs) is often several percent even with huge amounts of data
- OOVs are important source of errors in ASR systems
- Even in English, we have unlimited amount of proper names etc., so finite-sized vocabularies are not optimal

Subword-based language models - PTB

Model	Bits/character
NNLM	1.57
N-discounted n-gram	1.48
BPTT-RNN	1.42
HF-MRNN	1.41
Maximum Entropy n-gram (ME)	1.37

- Character-based models, including RNNs trained by BPTT and RNNs trained by Hessian-Free optimizer
- Results are on the Penn Treebank setup, with words divided into characters and space is rewritten as special symbol
- ME model works surprisingly well; usual smoothed N-grams work poor

Subword-based language models - text8, characters

Model	Bits/character
N-discounted n-gram	1.64
ME	1.55
RNNME	1.55
HF-MRNN	1.54
ME interpolated with HF-MRNN	1.47

- Results are on larger data set ('text8'), 90M training characters
- RNNs can learn a good character-level model, but the performance is still worse than of word-level models

Subword-based language models - text8, subwords

Model	Bits/fragment	Bits/character
Witten-Bell n-gram	4.71	1.58
ME	4.61	1.55
HF-MRNN	4.44	1.49
RNNME	4.36	1.46

- Text8, subwords (fragments) instead of characters
- Generally, the results for all models improved when we use fragments instead of characters
- Still, this model has zero OOV rate (all infrequent words are rewritten into sequences of shorter units - syllables or characters)
- This is another way to avoid problems with normalization over huge (potentially infinite) vocabularies with NNLMs

Compression of LMs with word-based RNNLMs - NIST RT04

Model	WER [%]	size (MB)	size(MB) compressed
unpruned 4-gram	-	2792	242*
4.7M 4-gram	14.1	122	12*
54M 4-gram	13.11	1862	162*
RNN-80 (text format)	12.98	130	-
RNN-80 (quantized)	13.00	-	13

- Again Broadcast news NIST RT04 task
- Compressed size of N-gram models is estimated based on the best results reported in research papers
- RNNLMs are order of magnitude smaller than compressed N-gram models
- As large N-gram LMs are used mainly for second pass rescoring of lattices, we can avoid this step and use smaller and better RNNLMs (rescoring speed is much faster than real-time, and can be easily optimized)

Compression of LMs with Subword-based RNNLMs - NIST RT05

Model	WER [%]	size (MB)	size (MB) compressed
Word-based bigram	27.0	93	11*
Word-based 4-gram	25.1	464	43*
Word-based 4-gram (pruned)	25.5	142	15*
Subword RNN-160 (text format)	26.5	6.7	-
Subword RNN-160 (quantized)	26.5	-	0.6
Subword RNN-480 (text format)	25.0	21.3	-
Subword RNN-480 (quantized)	24.9	-	1.7

- Meeting recognition NIST RT05 task
- By using subword RNNLMs, we can reduce size of models even more

Text compression with RNNLMs

- As RNNLMs are very general, we can simply add arithmetic coding and we obtain a data compressor!
- State of the art compression programs are PAQ archivers from Matthew Mahoney
- By using RNNME, we were able to beat the best PAQ program on large data set (1.7 GB of data) by almost 6%
- For the ME part, we added skip-gram features; also, two different models were trained, with different learning rate - otherwise, no more tricks were needed
- The only drawback is processing speed - only several KB/s with large RNN models

- Back to ideas behind Hutter prize: the best compressor is the most intelligent one! ;)

Conclusion

- We have shown amazing performance of neural net models on many tasks, including speech recognition, compression (both of LMs and of text), and also machine translation and general data compression (not in this presentation)
- Most experiments can be repeated as we implemented freely available toolkit for training RNN and RNNME language models (+all things like classes, support for rescoring of n-best lists, data generation, perplexity calculation, compression layer, dynamic evaluation and many other options)
- Several data sets are available on the RNNLM webpage, as well as papers
- RNNLM has also been integrated into novel speech recognition toolkit - KALDI

Conclusion

- We hope to boost the research of LM techniques, and to help to overcome the black magic practice (reporting results on private data sets, not comparing to proper baselines, providing misleading information)
- Language modeling is hard, but progress is needed to get closer to language understanding (and to AI) - at least in my opinion

Conclusion

- Thanks for attention!