EXTENSIONS OF RECURRENT NEURAL NETWORK LANGUAGE MODEL

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan “Honza” Černocký, Sanjeev Khudanpur

Speech@FIT, Brno University of Technology,
Johns Hopkins University

25. 5. 2011
Overview

- Introduction
- Model description
- Extensions
- Empirical evaluation
- Current work
Neural network based LMs outperform standard backoff n-gram models
- Words are projected into low dimensional space, similar words are automatically clustered together
- Smoothing is solved implicitly
- Standard backpropagation algorithm (BP) is used for training
- In [Mikolov2010], we have shown that recurrent neural network (RNN) architecture is competitive with the standard feedforward architecture
In this presentation, we will show:

- Importance of "backpropagation through time" (BPTT) [Rumelhart et al. 1986] training algorithm for RNN language models
- Simple speed-up technique that reduces computational complexity $10 \times - 100 \times$
- Results after combining randomly initialized RNN models
- Comparison of different advanced LM techniques on the same data set
- Results on large data sets and LVCSR experiments
Input layer $w$ and output layer $y$ have the same dimensionality as the vocabulary.

Hidden layer $s$ is orders of magnitude smaller.

$U$ is the matrix of weights between input and hidden layer, $V$ is the matrix of weights between hidden and output layer.
Training of RNNs by normal backpropagation is not optimal.

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 a usual feedforward representation of the network; such a deep network is then trained by backpropagation.

For on-line learning, "truncated BPTT" is used.
RNN unfolded in time
RNN unfolded in time
RNN unfolded in time
Words are assigned to "classes" based on their unigram frequency

First, class layer is evaluated; then, only words belonging to the predicted class are evaluated, instead of the whole output layer $y$

[Goodman2001]

Provides speedup in some cases more than $100 \times$
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 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 on Penn Corpus. Models are interpolated with the baseline 5-gram backoff model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kneser-Ney 5-gram</td>
<td>141</td>
</tr>
<tr>
<td>Random forest [Xu 2005]</td>
<td>132</td>
</tr>
<tr>
<td>Structured LM [Filimonov 2009]</td>
<td>125</td>
</tr>
<tr>
<td>Feedforward NN LM</td>
<td>116</td>
</tr>
<tr>
<td>Syntactic NN LM [Emami 2004]</td>
<td>110</td>
</tr>
<tr>
<td>RNN trained by BP</td>
<td>113</td>
</tr>
<tr>
<td>RNN trained by BPTT</td>
<td>106</td>
</tr>
<tr>
<td>4x RNN trained by BPTT</td>
<td>98</td>
</tr>
</tbody>
</table>
Speedup with different amount of classes

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

Values around $\sqrt{\text{vocabulary size}}$ lead to the largest speed-ups
The improvement obtained from a single RNN model over the best backoff model \textit{increases} with more data!
Current work

- Dynamic evaluation for model adaptation
- Combination and comparison of RNNs with many other advanced LM techniques
- More than 50% improvement in perplexity on large data set against modified Kneser-Ney smoothed 5-gram
Current work - ASR

- Almost 20% reduction of WER (Wall Street Journal) with simple ASR system, against backoff 5-gram model
  (WER 17.2% → 14.4%)

- Almost 10% reduction of WER (Broadcast News) with state of the art IBM system, against backoff 4-gram model
  (WER 13.1% → 12.0%)
Our experiments can be repeated using toolkit available at
http://www.fit.vutbr.cz/~imikolov/rnnlm/
Thanks for attention!