Part of Speech Tagging with LSTM Networks
Project Presentation

Zeming Lin

Department of Computer Science at University of Virginia

04/30/2015
Table of Contents

Background
  POS Tagging
  Recurrent Neural Networks

Methods and Results
  LSTM Networks
  Network Structure

Results and Discussion
  Results
  Discussion
Part of Speech

Noun

Pronoun

Verb

Adjective

Parts of Speech

Adverb

Preposition

Conjunction

Interjection
Penn Treebank Dataset

- We use 93915 words, from NLTK. Only consider sentences with length > 4.
- Already tokenized.
- Example:
  - Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.
  - NNP NNP, CD NNS JJ, MD VB DT NN IN DT JJ NN NNP CD.
State of the art

<table>
<thead>
<tr>
<th>Author</th>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brants (2000)</td>
<td>Hidden Markov Model</td>
<td>96.46%</td>
</tr>
<tr>
<td>Giménez and Márquez (2004)</td>
<td>SVM</td>
<td>97.16%</td>
</tr>
<tr>
<td>Spoustová et al. (2009)</td>
<td>Averaged Perceptron</td>
<td>97.44%</td>
</tr>
<tr>
<td>Manning (2011)</td>
<td>Dependency Network</td>
<td>97.32%</td>
</tr>
<tr>
<td>Søgaard (2011)</td>
<td>Condensed Nearest Neighbors</td>
<td>97.50%</td>
</tr>
</tbody>
</table>
State of the art

- Human disagreement is \( \sim 3.5\% \)
- Why is this interesting?
  - Machines often make very obvious mistakes
  - Single error tends to cascade to downstream modules for NLP
Table of Contents

Background
   POS Tagging
   Recurrent Neural Networks

Methods and Results
   LSTM Networks
   Network Structure

Results and Discussion
   Results
   Discussion
Neural Networks
Recurrent Networks
Recurrent networks

- Hard to train!
- Backpropagation through time is used to approximate training
Recurrent Networks

- BPTT algorithm not guaranteed to converge to a *local* minimum
  - Very sensitive to learning rate changes
- Exploding / vanishing gradients
Table of Contents

Background
  POS Tagging
  Recurrent Neural Networks

Methods and Results
  LSTM Networks
  Network Structure

Results and Discussion
  Results
  Discussion
Long-term memory

- Fixes the gradients problem, so we can train on longer time steps!
- LSTM Cell:
LSTM Cell

\[ i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \]
\[ \tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \]
\[ f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \]
\[ C_t = i_t \odot \tilde{C}_t + f_t \odot C_{t-1} \]
\[ o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_f) \]
\[ h_t = o_t \odot \tanh C_t \]
LSTM Cell

\[ i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \]
\[ \mathbf{\tilde{C}}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \]
\[ f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \]
\[ C_t = i_t \odot \tilde{C}_t + f_t \odot C_{t-1} \]
LSTM Cell

\[ o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_f) \]
\[ h_t = o_t \odot \tanh C_t \]
LSTM Network

Error gradients no longer vanish / explode!
Table of Contents

Background
   POS Tagging
   Recurrent Neural Networks

Methods and Results
   LSTM Networks
   Network Structure

Results and Discussion
   Results
   Discussion
Embedding is a $E = 50$ dim vector, trained from wikipedia, lookup table of 130k by 50.
Layers

- **Embedding** is a $E = 50$ dim vector, trained from wikipedia, lookuptable of 130k by 50.

- $R$ is the size of output
Embedding is a $E = 50$ dim vector, trained from wikipedia, lookuptable of 130k by 50.

- $R$ is the size of output

- $C$ is the memory of the network, the “error carousel”
- Embedding is a $E = 50$ dim vector, trained from wikipedia, lookuptable of 130k by 50.

- $R$ is the size of output

- $C$ is the memory of the network, the “error carousel”

- $V$ is number of tags to label, or 46.
Running scheme

- Run sequence through twice: Only consider the second run through
  - “Read entire sequence” before considering POS labels.
  - 2-time slowdown, but ~1-2% extra accuracy
Table of Contents

Background
  POS Tagging
  Recurrent Neural Networks

Methods and Results
  LSTM Networks
  Network Structure

Results and Discussion
  Results
  Discussion
Results

<table>
<thead>
<tr>
<th>$L$</th>
<th>$R$</th>
<th>$T$</th>
<th>Accuracy</th>
<th>Speed (wps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100</td>
<td>40</td>
<td>.942</td>
<td>319</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
<td>40</td>
<td>.952</td>
<td>88</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>40</td>
<td>.953</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>400</td>
<td>.942</td>
<td>363</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>10</td>
<td>.932</td>
<td>394</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>40</td>
<td>.936</td>
<td>239</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>40</td>
<td>.924</td>
<td>171</td>
</tr>
</tbody>
</table>

- Each network has $L$ layers
- Consider $T$-length sequences
- Cells memory of $R$ units.
Table of Contents

Background
   POS Tagging
   Recurrent Neural Networks

Methods and Results
   LSTM Networks
   Network Structure

Results and Discussion
   Results
   Discussion
Discussion

<table>
<thead>
<tr>
<th>$L$</th>
<th>$R$</th>
<th>$T$</th>
<th>Accuracy</th>
<th>Speed (wps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100</td>
<td>40</td>
<td>.942</td>
<td>319</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
<td>40</td>
<td>.952</td>
<td>88</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>40</td>
<td>.953</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>400</td>
<td>.942</td>
<td>363</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>10</td>
<td>.932</td>
<td>394</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>40</td>
<td>.936</td>
<td>239</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>40</td>
<td>.924</td>
<td>171</td>
</tr>
</tbody>
</table>

- More layers == worse performance?
- Increase number of training iterations?
<table>
<thead>
<tr>
<th>$L$</th>
<th>$R$</th>
<th>$T$</th>
<th>Accuracy</th>
<th>Speed (wps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100</td>
<td>40</td>
<td>.942</td>
<td>319</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
<td>40</td>
<td>.952</td>
<td>88</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>40</td>
<td>.953</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>400</td>
<td>.942</td>
<td>363</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>10</td>
<td>.932</td>
<td>394</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>40</td>
<td>.936</td>
<td>239</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>40</td>
<td>.924</td>
<td>171</td>
</tr>
</tbody>
</table>

- High $T$ doesn’t impact, but low $T$ does
- Memory units $R$ had large impact, 100 → 250 gave 1% accuracy!
Future Work

- Find the full Treebank dataset, see if we get state of the art 97.5% results
- Test larger models, use GPU to parallelize matrix computation
- Batch gradient descent to parallelize training, can use Mapreduce
Thank you!