Neural Network Language Modelling

\[ \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) \]
Presentation Outline

- Why language models are important

- N-Gram modelling

- Neural Network Language Models:
  - Feedforward Networks (Bengio et al., 2003)
  - Recurrent Networks (Mikolov et al., 2010)
How are language models useful?

• Evaluate probability of a sequence of words occurring naturally in a text

• Predict the next word given the preceding words

speech recognition:

"wreck a nice beach" vs. "recognize speech"
N-Gram Modelling

"it's not rocket science"

\[ P(w_1, \ldots, w_m) \approx \prod_{i=1}^{m} P(w_i | w_{i-(n-1)}, \ldots, w_{i-1}) \]

when using trigrams (n = 3):

\[ P(\text{it's not rocket science}) \approx P(\text{it's} | <s> <s>) \cdot P(\text{not} | \text{it's}, <s>) \cdot P(\text{rocket} | \text{it's}, \text{not}) \cdot P(\text{science} | \text{rocket}, \text{not}) \]
N-Gram Modelling

"it's not rocket science"

Using MLE to collect n-gram statistics:

History (H) = "not rocket"
Word (A) = "science"

\[ P(A | H) = \frac{\text{count}(H + A)}{\text{count}(H)} \]

\[ P(\text{science} | \text{it's not rocket}) = \frac{\text{count}(\text{not rocket science})}{\text{count}(\text{not rocket})} \]
Problem with N-Grams

- They don't make use of longer contexts

"The sky above our heads is blue."

"The sky on a sunny day is blue."
Maps input words to a feature space. Closer words are more similar with regard to their features.

Input: '1-of-V' coding

Training sentence: "I go to class on Monday" generalizes to...
Feedforward Neural Network

Input layer: 1-of-V encoding

Projection layer: Lower dimensional representation of input

Hidden Layer: where probability calculations occur

Output layer: normalizes probabilities
Feedforward Neural Network

\[ \text{Parameters: weights and feature vectors} \]
\[ \theta = (C, \omega) \]
The network can be defined by:

\[ P(w_t = k|w_{t-n+1}, \ldots w_{t-1}) = \frac{e^{a_k}}{\sum_{l=1}^{N} e^{a_l}} \]

"softmax"

where,

\[ a_k = b_k + \sum_{i=1}^{h} W_{ki} \tanh(c_i + \sum_{j=1}^{(n-1)d} V_{ij}x_j) \]

Calculates unnormalized log probabilities for each output word

output layer normalizes to make probability distribution = 1
Training the network

Find the parameters

$$\theta = (C, \omega)$$

that maximize log probability of the training corpus

$$L = \frac{1}{T} \sum_T \log f(w_t, w_{t-1}, \cdots, w_{t-n+1}; \theta) + R(\theta)$$

using gradient ascent/descent with backpropagation

$$\frac{\partial L}{\partial \theta}$$
Experiment

- Trained on 800,000 words in Brown corpus

- 24% lower perplexity than modified Kneser-Kney

- Context length of 5 worked best for the feedforward model

- Trigrams worked best for the Kneser-Kney

Do feedforward networks really make use of longer contexts?
Recurrent Neural Networks

The hidden layer of RNN represents all previous history and not just \( (n-1) \) previous words.

No projection layer.
More RNN

Input into the RNN is a 1-to-V word encoding + the previous state:

\[ x(t) = w(t) + s(t - 1) \]

Hidden layer uses a sigmoid function \( f(z) = \frac{1}{1 + e^{-z}} \) to calculate unnormalized probabilities:

\[ s_j(t) = f \left( \sum_i x_i(t) u_{ji} \right) \]

Output layer normalizes with softmax function:

\[ g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \quad y_k(t) = g \left( \sum_j s_j(t) v_{kj} \right) \]
Training RNN

\[
\text{error}(t) = \text{desired}(t) - y(t)
\]

desired = word that should have been predicted in a particular context

\[ y = \text{actual word that was predicted} \]

Use Back Propagation Through Time:
Experiments

• speech recognition task

• RNN trained on 6.4M words from NYT section of English Gigaword
  (300k sentences - takes several weeks)

• Other models trained on 37M words

Table 2: Comparison of various configurations of RNN LMs and combinations with backoff models while using 6.4M words in training data (WSJ DEV).

<table>
<thead>
<tr>
<th>Model</th>
<th>PPL</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RNN</td>
<td>RNN+KN</td>
</tr>
<tr>
<td>KN5 - baseline</td>
<td>-</td>
<td>221</td>
</tr>
<tr>
<td>RNN 60/20</td>
<td>229</td>
<td>186</td>
</tr>
<tr>
<td>RNN 90/10</td>
<td>202</td>
<td>173</td>
</tr>
<tr>
<td>RNN 250/5</td>
<td>173</td>
<td>155</td>
</tr>
<tr>
<td>RNN 250/2</td>
<td>176</td>
<td>156</td>
</tr>
<tr>
<td>RNN 400/10</td>
<td>171</td>
<td>152</td>
</tr>
<tr>
<td>3xRNN static</td>
<td>151</td>
<td>143</td>
</tr>
<tr>
<td>3xRNN dynamic</td>
<td>128</td>
<td>121</td>
</tr>
</tbody>
</table>

Nearly 50% PPL reduction!
Experiment 2

NIST RT05 data set

- RNN 5.4M words
- Other models trained over 100x more data

Table 4: Comparison of very large back-off LMs and RNN LMs trained only on limited in-domain data (5.4M words).

<table>
<thead>
<tr>
<th>Model</th>
<th>WER static</th>
<th>WER dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT05 LM</td>
<td>24.5</td>
<td>-</td>
</tr>
<tr>
<td>RT09 LM - baseline</td>
<td>24.1</td>
<td>-</td>
</tr>
<tr>
<td>KN5 in-domain</td>
<td>25.7</td>
<td>-</td>
</tr>
<tr>
<td>RNN 500/10 in-domain</td>
<td>24.2</td>
<td>24.1</td>
</tr>
<tr>
<td>RNN 500/10 + RT09 LM</td>
<td><strong>23.3</strong></td>
<td>23.2</td>
</tr>
<tr>
<td>RNN 800/10 in-domain</td>
<td>24.3</td>
<td>23.8</td>
</tr>
<tr>
<td>RNN 800/10 + RT09 LM</td>
<td>23.4</td>
<td>23.1</td>
</tr>
<tr>
<td>RNN 1000/5 in-domain</td>
<td>24.2</td>
<td>23.7</td>
</tr>
<tr>
<td>RNN 1000/5 + RT09 LM</td>
<td>23.4</td>
<td>22.9</td>
</tr>
<tr>
<td>3xRNN + RT09 LM</td>
<td><strong>23.3</strong></td>
<td><strong>22.8</strong></td>
</tr>
</tbody>
</table>
questions...?