Machine Translation
03: Language Models

Rico Sennrich
University of Edinburgh
Overview

Last lecture
- simple neural networks
- real-valued vectors as input and output

Today’s lecture
- how do we represent language in neural networks?
- how do we treat language probabilistically (with neural networks)?
A probabilistic model of translation

- Suppose that we have:
  - a source sentence $S$ of length $m (x_1, \ldots, x_m)$
  - a target sentence $T$ of length $n (y_1, \ldots, y_n)$

- We can express translation as a probabilistic model

$$T^* = \arg \max_T P(T|S)$$

- Expanding using the chain rule gives

$$P(T|S) = P(y_1, \ldots, y_n|x_1, \ldots, x_m)$$

$$= \prod_{i=1}^{n} P(y_i|y_1, \ldots, y_{i-1}, x_1, \ldots, x_m)$$
A probabilistic model of language

- simpler: instead of $P(T|S)$, what is $P(T)$?
- why do we care?
  - language model is integral part of statistical machine translation:

\[
T^* = \arg \max_T P(S|T)P(T)
\]

Bayes' theorem

\[
T^* \approx \arg \max_T \sum_{m=1}^{M} \lambda_m h_m(S, T)
\]  

[Och, 2003]

- in neural machine translation, separate language model is untypical, but architectures are similar
- language models have many other applications in NLP:
  - language identification
  - predictive typing
  - as a component in various NLP tasks
N-gram language model

chain rule and Markov assumption

- A sentence $T$ of length $n$ is a sequence $x_1, \ldots, x_n$

$$P(T) = P(x_1, \ldots, x_n)$$

$$= \prod_{i=1}^{n} P(x_i|x_0, \ldots, x_{i-1})$$

$(\text{chain rule})$

$$\approx \prod_{i=1}^{n} P(x_i|x_{i-k}, \ldots, x_{i-1})$$

$(\text{Markov assumption: n-gram model})$
Discrete n-gram models

**count-based models**

- estimate probability of n-grams via counting
- main challenge: how to estimate probability of unseen n-grams?

**n-grams**

- **zerogram** uniform distribution
- **unigram** probability estimated from word frequency
- **bigram** $x_i$ depends only on $x_{i-1}$
- **trigram** $x_i$ depends only on $x_{i-2}, x_{i-1}$
German police have recovered more than 100 items stolen from John Lennon’s estate, including three diaries. The diaries were put on display at Berlin police headquarters with other items including a tape recording of a Beatles concert, two pairs of glasses, sheet music and a cigarette case. Police said a 58-year-old man had been arrested on suspicion of handling stolen goods. The items were stolen in New York in 2006 from Lennon’s widow, Yoko Ono. Detectives said much of the haul was confiscated from an auction house in Berlin in July, sparking an investigation to find the rest of the stolen items. Ono identified the objects from photos she was shown at the German consulate in New York, German media reported.
German police have recovered more than 100 items stolen from John Lennon’s estate, including three diaries. The diaries were put on display at Berlin police headquarters with other items including a tape recording of a Beatles concert, two pairs of glasses, sheet music and a cigarette case. Police said a 58-year-old man had been arrested on suspicion of handling stolen goods. The items were stolen in New York in 2006 from Lennon’s widow, Yoko Ono. Detectives said much of the haul was confiscated from an auction house in Berlin in July, sparking an investigation to find the rest of the stolen items. Ono identified the objects from photos she was shown at the German consulate in New York, German media reported.

$$P(\text{of}) = \frac{5}{101} \approx 0.041$$
German police have recovered more than 100 items stolen from John Lennon’s estate, including three diaries. The diaries were put on display at Berlin police headquarters with other items including a tape recording of a Beatles concert, two pairs of glasses, sheet music and a cigarette case. Police said a 58-year-old man had been arrested on suspicion of handling stolen goods. The items were stolen in New York in 2006 from Lennon’s widow, Yoko Ono. Detectives said much of the haul was confiscated from an auction house in Berlin in July, sparking an investigation to find the rest of the stolen items. Ono identified the objects from photos she was shown at the German consulate in New York, German media reported.

\[
P(\text{the}|\text{of}) = \frac{P(\text{of the})}{P(\text{of})} \approx \frac{0.0165}{0.041} \approx 0.4
\]

\[
= \frac{C(\text{of the})}{C(\text{of})} = \frac{2}{5} = 0.4
\]
Sparse data

- what is probability of "of his"?
  → unseen in our training data, so we estimate $P(\text{of his}) \approx 0$

- can we simply use more training data?
  → for higher $n$, most $n$-grams will be unseen

### Google n-grams

<table>
<thead>
<tr>
<th>length</th>
<th>number</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13,588,391</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>314,843,401</td>
<td>1.70e-06</td>
</tr>
<tr>
<td>3</td>
<td>977,069,902</td>
<td>3.89e-13</td>
</tr>
<tr>
<td>4</td>
<td>1,313,818,354</td>
<td>3.85e-20</td>
</tr>
<tr>
<td>5</td>
<td>1,176,470,663</td>
<td>2.54e-27</td>
</tr>
<tr>
<td>Tokens</td>
<td>1,024,908,267,229</td>
<td></td>
</tr>
</tbody>
</table>
Smoothing

- core idea: reserve part of probability mass for unseen events.
- most popular: back-off smoothing: if \( n \)-gram is unseen, make estimate based on smaller \( n \)-grams

example: Jelinek-Mercer smoothing

\[
p_{\text{smooth}}(x|h) = \begin{cases} 
\alpha(x|h) + \gamma(h)\beta(x|h) & C(x, h) > 0 \\
\gamma(h)\beta(x|h) & C(x, h) = 0 
\end{cases}
\]

\[
= \alpha(x|h) + \gamma(h)\beta(x|h)
\]

\[
\alpha(x|h) = \lambda(h)p_{ML}(x|h) = \lambda(h)\frac{C(x, h)}{C(h)}
\]

\[
\gamma(h) = 1 - \lambda(h)
\]

\[
\beta(x_i|x_{i-n+1}^{i-1}) = p_{\text{smooth}}(x_i|x_{i-n+1}^{i-1})
\]
core idea: rather than backing off, rely on similarity in internal representation for estimating unseen events:

\[ P(\text{barks|the Rottweiler}) \approx P(\text{barks|the Terrier}) \]
Continuous n-gram language models

n-gram NNLM [Bengio et al., 2003]

- input: context of n-1 previous words
- output: probability distribution for next word
- linear embedding layer with shared weights
- one or several hidden layers

[Vaswani et al., 2013]
Representing words as vectors

One-hot encoding

- example vocabulary: 'man', 'runs', 'the', '.'
- input/output for $p(\text{runs} | \text{the man})$:

$$x_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad x_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad y_{\text{true}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

- size of input/output vector: vocabulary size
- embedding layer is lower-dimensional and dense
  - smaller weight matrices
  - network learns to group similar words to similar point in vector space
Multi-class logistic regression

\[
\begin{align*}
\mathbf{x} &= [x_1, x_2, \ldots, x_n] \\
\theta^{(0)} &= [\theta_0^{(0)}, \theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_n^{(0)}] \\
\theta^{(2)} &= [\theta_0^{(2)}, \theta_1^{(2)}, \ldots, \theta_n^{(2)}] \\
g(z) &= \text{softmax}(z) \\
P(c = 0|\Theta, X) &\quad P(c = 1|\Theta, X) &\quad P(c = 2|\Theta, X)
\end{align*}
\]

Features

Input layer

Layer 1
Softmax activation function

$$p(y = j | x) = \frac{e^{x_j}}{\sum_k e^{x_k}}$$

- softmax function normalizes output vector to probability distribution
  → computational cost linear to vocabulary size (!)
Cross-entropy Loss

given: predict probability 1 for correct word; 0 for rest:

\[
L(\Theta) = - \sum_{k=1}^{c} \delta(y, k) \log p(k|x, \Theta)
\]

\[
\delta(x, y) = \begin{cases} 
1 & \text{if } x = y \\
0 & \text{otherwise}
\end{cases}
\]

simplified:

\[
L(\Theta) = - \log p(y|x, \Theta)
\]
Feedforward neural language model: math

\[ h_1 = \varphi W_1(E x_1, E x_2) \]
\[ y = \text{softmax}(W_2 h_1) \]
Recurrent neural network language model (RNNLM)

**RNNLM [Mikolov et al., 2010]**

- motivation: condition on arbitrarily long context
  → no Markov assumption
- we read in one word at a time, and update hidden state incrementally
- hidden state is initialized as empty vector at time step 0
- parameters:
  - embedding matrix $E$
  - feedforward matrices $W_1, W_2$
  - recurrent matrix $U$

$$h_i = \begin{cases} 
0, & \text{if } i = 0 \\
\tanh(W_1 E x_i + U h_{i-1}) & \text{if } i > 0
\end{cases}$$

$$y_i = \text{softmax}(W_2 h_{i-1})$$
Cross-entropy Loss in RNNs

- **unrolling** RNN produces acyclic network (with shared weights)
  - backpropagation like with feed-forward network
  - each time step contributes to (shared) weight update

- loss is applied to every word:

\[
L(\Theta) = - \sum_{i=1}^{n} \log p(x_i | x_1, \ldots, x_{i-1}, \Theta)
\]

- **teacher forcing**: for each word, condition prediction on true history
  → efficient, but mismatch to test time (where history is unreliable)

Christopher Olah http://colah.github.io/posts/2015-08-Understanding-LSTMs/
RNNs and long distance dependencies

Christopher Olah
http://colah.github.io/posts/2015-08-Understanding-LSTMs/
RNNs and long distance dependencies

Christopher Olah

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

R. Sennrich

MT – 2018 – 03
Long Short-Term Memory (LSTM)

Christopher Olah
http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Long Short-Term Memory (LSTM)

[Diagram of LSTM architecture with inputs $X_{t-1}$, $X_t$, $X_{t+1}$, and outputs $h_{t-1}$, $h_t$, $h_{t+1}$, showing gates $\sigma$, $\tanh$, and sigmoid functions]
Long Short-Term Memory (LSTM) – Step-by-step

\[ f_t = \sigma \left( W_f \cdot [h_{t-1}, x_t] + b_f \right) \]

Christopher Olah: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
Long Short-Term Memory (LSTM) – Step-by-step

\[
\begin{align*}
    i_t &= \sigma \left( W_i \cdot [h_{t-1}, x_t] + b_i \right) \\
    \tilde{C}_t &= \tanh \left( W_C \cdot [h_{t-1}, x_t] + b_C \right)
\end{align*}
\]
Long Short-Term Memory (LSTM) – Step-by-step

\[
C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t
\]
Long Short-Term Memory (LSTM) – Step-by-step

\[
o_t = \sigma \left( W_o \left[ h_{t-1}, x_t \right] + b_o \right)
\]
\[
h_t = o_t \times \tanh (C_t)
\]

Christopher Olah http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Gated Recurrent Units (GRUs)

\[ z_t = \sigma (W_z \cdot [h_{t-1}, x_t]) \]

\[ r_t = \sigma (W_r \cdot [h_{t-1}, x_t]) \]

\[ \tilde{h}_t = \tanh (W \cdot [r_t \ast h_{t-1}, x_t]) \]

\[ h_t = (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t \]

Christopher Olah http://colah.github.io/posts/2015-08-Understanding-LSTMs/
RNN variants

Gated units

- Sigmoid layers $\sigma$ act as "gates" that control flow of information.
- Reduces vanishing gradient problem.
- Strong empirical results.

[Christopher Olah](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
Further Reading

Required Reading
- Koehn, 13.4

Optional Reading
- Basic probability theory (Sharon Golwater):
  http://homepages.inf.ed.ac.uk/sgwater/math_tutorials.html
- Introduction to LSTMs:
  http://colah.github.io/posts/2015-08-Understanding-LSTMs/

