POS tagging

Labeling the tag that’s correct for the context.

Fruit flies like a banana

Time flies like an arrow

(Just tags in evidence within the Penn Treebank — more are possible!)
Named entity recognition

tim cook is the ceo of apple

3 or 4-class:
- person
- location
- organization
- (misc)

7-class:
- person
- location
- organization
- time
- money
- percent
- date
Supersense tagging

The station wagons arrived at noon, a long shining line

that coursed through the west campus.

Noun supersenses (Ciarmita and Altun 2003)
Segmentation

- $B =$ character is the start of new word
- $I =$ character is inside existing word

<table>
<thead>
<tr>
<th>#</th>
<th>b</th>
<th>l</th>
<th>a</th>
<th>c</th>
<th>k</th>
<th>l</th>
<th>i</th>
<th>v</th>
<th>e</th>
<th>s</th>
<th>m</th>
<th>a</th>
<th>t</th>
<th>t</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>B</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
</tbody>
</table>

# black lives matter

# black live smatter
Sequence labeling

For a set of inputs $x$ with $n$ sequential time steps, one corresponding label $y_i$ for each $x_i$.

$x = \{x_1, \ldots, x_n\}$

$y = \{y_1, \ldots, y_n\}$
# Sequence labeling models

<table>
<thead>
<tr>
<th>Model</th>
<th>Form</th>
<th>Label Dependency</th>
<th>Rich Features?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden Markov Models</td>
<td>$\prod_{i=1}^{N} P(x_i</td>
<td>y_i) P(y_i</td>
<td>y_{i-1})$</td>
</tr>
<tr>
<td>MEMM</td>
<td>$\prod_{i=1}^{N} P(y_i</td>
<td>y_{i-1}, x, \beta)$</td>
<td>Markov assumption</td>
</tr>
<tr>
<td>CRF</td>
<td>$P(y</td>
<td>x, \beta)$</td>
<td>pairwise through entire sequence</td>
</tr>
<tr>
<td>RNN</td>
<td>$\prod_{i=1}^{N} P(y_i</td>
<td>x_{1:i}, \beta)$</td>
<td>none</td>
</tr>
</tbody>
</table>
Back to RNNs

• RNN allow arbitrarily-sized conditioning contexts; condition on the entire sequence history.

• We used RNNs for document classification to generate a representation of a sequence that we can then use for prediction.
I

loved

the

movie

!
I loved the movie!
I loved the movie!

Iyyer et al. (2015), “Deep Unordered Composition Rivals Syntactic Methods for Text Classification” (ACL)
I loved the movie!

\[ y = x_1a_1 + x_2a_2 + x_3a_3 + x_4a_4 + x_5a_5 \]

weighted sum
Each time step has two inputs:

- $x_i$ (the observation at time step $i$); one-hot vector, feature vector or word embedding.
- $s_{i-1}$ (the output of the previous state); base case: $s_0 = 0$ vector
Recurrent neural network

\[ s_i = R(x_i, s_{i-1}) \]

\[ y_i = O(s_i) \]

*R* computes the output state as a function of the current input and previous state.

*O* computes the output as a function of the current output state.
“Simple” RNN

\[ s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b) \]

Different weight vectors \( W \) transform the previous state and current input before combining:

\[ W^s \in \mathbb{R}^{H \times H} \]
\[ W^x \in \mathbb{R}^{D \times H} \]
\[ b \in \mathbb{R}^{H} \]

\[ y_i = O(s_i) = s_i \]

Elman 1990, Mikolov 2012
Recurrent neural network

- Often used for sequential prediction tasks:
  - Language models—predicting the next symbol (word, character) in a sequence
  - Machine translation—predicting a sequence of words (sentence) in language $f$ conditioned on sentence in language $e$
  - Sequence labeling (POS tagging, NER)
RNNs for sequence labeling

- The output state $s_i$ is an $H$-dimensional real vector; we can transfer that into a probability by passing it through an additional linear transformation followed by a softmax

$$y_i = O(s_i) = \text{softmax}(s_i W^o + b^o)$$
Training RNNs

• Given this definition of an RNN:

\[ s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b) \]

\[ y_i = O(s_i) = \text{softmax}(s_iW^o + b^o) \]

• We have five sets of parameters to learn:

\[ W^s, W^x, W^o, b, b^o \]
For POS tagging, predict the tag from $y$ conditioned on the context.
RNNs for POS

- To make a prediction for $y_t$, RNNs condition on all input seen through time $t$ ($x_1, \ldots, x_t$)

- But knowing something about the future can help ($x_{t+1}, \ldots, x_n$)

The horse raced past the barn fell

- DT
- NN
- VBD
- IN
- DT
- NN
- ???
Bidirectional RNN

• A powerful alternative is make predictions conditioning both on the past and the future.

• Two RNNs
  • One running left-to-right
  • One right-to-left

• Each produces an output vector at each time step, which we concatenate
Bidirectional RNN

I loved the movie!

backward RNN
Bidirectional RNN

I

loved

the

movie

!

I

loved

the

movie

!
I loved the movie!
Training BiRNNs

• Given this definition of an BiRNN:

\[ s_b^i = R_b(x^i, s_b^{i+1}) = g(s_b^{i+1} W_b^s + x^i W_b^x + b_b) \]

\[ s_f^i = R_f(x^i, s_f^{i-1}) = g(s_f^{i-1} W_f^s + x^i W_f^x + b_f) \]

\[ y_i = \text{softmax} \left( [s_f^i; s_b^i] W^o + b^o \right) \]

• We have 8 sets of parameters to learn (3 for each RNN + 2 for the final layer)
How do we fix this?
BiLSTM for each word; concatenate final state of forward LSTM, backward LSTM, and word embedding as representation for a word.

Lample et al. (2016), “Neural Architectures for Named Entity Recognition”
BiLSTM for each word; concatenate final state of forward LSTM, backward LSTM, and word embedding as representation for a word.

Lample et al. (2016), “Neural Architectures for Named Entity Recognition”
Character CNN for each word; concatenate character CNN output and word embedding as representation for a word.

Chu et al. (2016), “Named Entity Recognition with Bidirectional LSTM-CNNs”
LSTM/RNN

• An RNN doesn’t use the dependencies between nearby labels in making predictions.
The information that's passed between states is not the categorical choice (VBZ) but a hidden state that generated the distribution.
If we knew the categorical choice of VBZ at t₂, P(VB) at t₃ would be much lower.
TimeDistributed

- In Keras, the `TimeDistributed` wrapper applies the same operation to every time step in a sequence (e.g., the same `Dense` layer with the same parameters)
I loved the movie!
lstm_output = LSTM(lstm_size, return_sequences=True) (embedded_sequences) 

preds = TimeDistributed(Dense(output_dim, activation="softmax"))(lstm_output)
Activity

• 12.ner/SequenceLabelingBiLSTM.TODO