CS388: Natural Language Processing
Lecture 9: CNNs, Neural CRFs

Greg Durrett
Administrivia

- Project 1 due today at 5pm
- Mini 2 out tonight, due in two weeks
This Lecture

- CNNs
- CNNs for Sentiment
- Neural CRFs
CNNs
Convolutional Layer

- Applies a *filter* over patches of the input and returns that filter’s activations
- Convolution: take dot product of filter with a patch of the input

image: $n \times n \times k$  
filter: $m \times m \times k$

Each of these cells is a vector with multiple values
Images: RGB values (3 dim); text: word vector (50+ dim)

$$activation_{ij} = \sum_{i_o=0}^{k-1} \sum_{j_o=0}^{k-1} image(i + i_o, j + j_o) \cdot filter(i_o, j_o)$$

sum over dot products offsets
Convolutional Layer

- Applies a *filter* over patches of the input and returns that filter’s activations.
- Convolution: take dot product of filter with a patch of the input.

**Image:** \( n \times n \times k \)  
**Filter:** \( m \times m \times k \)  
**Activations:** \( (n - m + 1) \times (n - m + 1) \times 1 \)
Convolutions for NLP

- Input and filter are 2-dimensional instead of 3-dimensional

Sentence: $n \text{ words} \times k \text{ vec dim}$

Filter: $m \times k$

Activations: $(n - m + 1) \times 1$

The movie was good

Vector for each word

- Combines evidence locally in a sentence and produces a new (but still variable-length) representation
Compare: CNNs vs. LSTMs

- Both LSTMs and convolutional layers transform the input using context.
- LSTM: “globally” looks at the entire sentence (but local for many problems).
- CNN: local depending on filter width + number of layers.

The movie was good.

The diagram illustrates the computational complexity and structure of CNNs and LSTMs, showing how each type processes input differently.
CNNs for Sentiment
CNNs for Sentiment Analysis

- Projection + softmax
- c-dimensional vector
- Max pooling over the sentence

- n x c
- c filters, m x k each
- n x k

- The movie was good

$P(y|x)$

Max pooling: return the max activation of a given filter over the entire sentence; like a logical OR (sum pooling is like logical AND)
Filter “looks like” the things that will cause it to have high activation.
the 0.03
movie 0.02
was 0.1

good 1.1
.

“bad“ 0.1

“okay“ 0.3

“terrible“ 0.1

max = 1.1

Understanding CNNs for Sentiment
Understanding CNNs for Sentiment

- Takes variable-length input and turns it into fixed-length output
- Filters are initialized randomly and then learned
Understanding CNNs for Sentiment

- Word vectors for similar words are similar, so convolutional filters will have similar outputs.
the  
movie  
was  
not  
good  

“not good”

max = 1.5

- Analogous to bigram features in bag-of-words models
- Indicator feature of text containing bigram <-> max pooling of a filter that matches that bigram
What can CNNs learn?

the movie was not good

the movie was not really all that good

the cinematography was good, the music great, but the movie was bad

I entered the theater in the bloom of youth and left as an old man
Deep Convolutional Networks

- Low-level filters: extract low-level features from the data

Zeiler and Fergus (2014)
High-level filters: match larger and more “semantic patterns”
Input is batch_size x n x k matrix, filters are c x m x k matrix (c filters)

Typically use filters with m ranging from 1 to 5 or so (multiple filter widths in a single convnet)

All computation graph libraries support efficient convolution operations
CNNs for Sentence Classification

- Question classification, sentiment, etc.
- Conv+pool, then use feedforward layers to classify
- Can use multiple types of input vectors (fixed initializer and learned)

Kim (2014)
Sentence Classification

- movie review sentiment
- subjectivity/objectivity detection
- product reviews
- question type classification

Also effective at document-level text classification

<table>
<thead>
<tr>
<th>Model</th>
<th>MR</th>
<th>SST-1</th>
<th>SST-2</th>
<th>Subj</th>
<th>TREC</th>
<th>CR</th>
<th>MPQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-multichannel</td>
<td>81.1</td>
<td>47.4</td>
<td>88.1</td>
<td>93.2</td>
<td>92.2</td>
<td>85.0</td>
<td>89.4</td>
</tr>
<tr>
<td>NBSVM (Wang and Manning, 2012)</td>
<td>79.4</td>
<td>–</td>
<td>–</td>
<td>93.2</td>
<td>–</td>
<td>81.8</td>
<td>86.3</td>
</tr>
</tbody>
</table>

Kim (2014)
Neural CRF Basics
Features in CRFs: I[tag=B-LOC & curr_word=Hangzhou], I[tag=B-LOC & prev_word=to], I[tag=B-LOC & curr_prefix=Han]

Linear model over features

Downsides:
- Lexical features mean that words need to be seen in the training data
- Linear model can’t capture feature conjunctions as effectively (can’t look at more than 2 words with a single feature)
Barack Obama will travel to Hangzhou today for the G20 meeting.

- Transducer (LM-like model)
- What are the strengths and weaknesses of this model compared to CRFs?
LSTMs for NER

Barack Obama will travel to Hangzhou today for the G20 meeting.

- Bidirectional transducer model
- What are the strengths and weaknesses of this model compared to CRFs?
Neural CRFs

Barack Obama will travel to Hangzhou today for the G20 meeting.

- Neural CRFs: bidirectional LSTMs (or some NN) compute emission potentials, capture structural constraints in transition potentials
Neural CRFs

\[
P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x))
\]

- Conventional: \( \phi_e(y_i, i, x) = w^\top f_e(y_i, i, x) \)
- Neural: \( \phi_e(y_i, i, x) = W_{y_i}^\top f(i, x) \)  \( W \) is a num_tags x len(f) matrix
- \( f(i, x) \) could be the output of a feedforward neural network looking at the words around position \( i \), or the \( i \)th output of an LSTM, ...
- Neural network computes unnormalized potentials that are consumed and “normalized” by a structured model
- Inference: compute \( f \), use Viterbi
Computing Gradients

\[ P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x)) \]

- **Conventional:** \( \phi_e(y_i, i, x) = w^\top f_e(y_i, i, x) \)

- **Neural:** \( \phi_e(y_i, i, x) = W_{y_i}^\top f(i, x) \)

\[ \frac{\partial L}{\partial \phi_{e,i}} = -P(y_i = s|x) + I[s \text{ is gold}] \] “error signal”, compute with F-B chain rule say to multiply together, gives our update

- For linear model: \( \frac{\partial \phi_{e,i}}{w_i} = f_{e,i}(y_i, i, x) \)

- For neural model: compute gradient of \( \phi \) w.r.t. parameters of neural net
Neural CRFs

1) Compute $f(x)$
2) Run forward-backward
3) Compute error signal
4) Backprop (no knowledge of sequential structure required)

Barack Obama will travel to Hangzhou today for the G20 meeting.
**FFNN Neural CRF for NER**

<table>
<thead>
<tr>
<th>B-PER</th>
<th>I-PER</th>
<th>O</th>
<th>O</th>
<th>O</th>
<th>B-LOC</th>
<th>O</th>
<th>O</th>
<th>O</th>
<th>B-ORG</th>
<th>O</th>
<th>O</th>
</tr>
</thead>
</table>

*Barack Obama* will travel to *Hangzhou* today for the *G20* meeting.

PERSON | LOC | ORG

\[
\phi_e = W g(V f(x, i))
\]

\[
f(x, i) = [\operatorname{emb}(x_{i-1}), \operatorname{emb}(x_i), \operatorname{emb}(x_{i+1})]
\]

**FFNN**

<table>
<thead>
<tr>
<th>previous word</th>
<th>curr word</th>
<th>next word</th>
</tr>
</thead>
<tbody>
<tr>
<td>e(to)</td>
<td>e(Hangzhou)</td>
<td>e(today)</td>
</tr>
</tbody>
</table>

*to Hangzhou* today
Barack Obama will travel to Hangzhou today for the G20 meeting.

Bidirectional LSTMs compute emission (or transition) potentials
Barack Obama will travel to Hangzhou today for the G20 meeting.

How does this compare to neural CRF?
### Benchmark Systems

<table>
<thead>
<tr>
<th>Approach</th>
<th>POS (PWA)</th>
<th>CHUNK (F1)</th>
<th>NER (F1)</th>
<th>SRL (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN+WLL</td>
<td>96.31</td>
<td>89.13</td>
<td>79.53</td>
<td>55.40</td>
</tr>
<tr>
<td>NN+SLL</td>
<td>96.37</td>
<td>90.33</td>
<td>81.47</td>
<td>70.99</td>
</tr>
<tr>
<td>NN+WLL+LM1</td>
<td>97.05</td>
<td>91.91</td>
<td>85.68</td>
<td>58.18</td>
</tr>
<tr>
<td>NN+SLL+LM1</td>
<td>97.10</td>
<td>93.65</td>
<td>87.58</td>
<td>73.84</td>
</tr>
<tr>
<td>NN+WLL+LM2</td>
<td>97.14</td>
<td>92.04</td>
<td>86.96</td>
<td>58.34</td>
</tr>
<tr>
<td>NN+SLL+LM2</td>
<td>97.20</td>
<td>93.63</td>
<td>88.67</td>
<td>74.15</td>
</tr>
</tbody>
</table>

- **WLL**: independent classification; **SLL**: neural CRF
- **LM2**: word vectors learned from a precursor to word2vec/GloVe, trained for 2 weeks (!) on Wikipedia

CNN Neural CRFs

- Append to each word vector an *embedding of the relative position* of that word
- Convolution over the sentence produces a position-dependent representation

Diagram:
- Conv + pool + FFNN
- Words: travel, to, Hangzhou, today, for
- Positions: -2, -1, 0, 1, 2

Text Example:
- travel to *Hangzhou* today for
## CNN NCRFs vs. FFNN NCRFs

<table>
<thead>
<tr>
<th>Approach</th>
<th>POS (PWA)</th>
<th>CHUNK (F1)</th>
<th>NER (F1)</th>
<th>SRL (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark Systems</td>
<td>97.24</td>
<td>94.29</td>
<td>89.31</td>
<td>77.92</td>
</tr>
<tr>
<td>Window Approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN+SLL+LM2</td>
<td>97.20</td>
<td>93.63</td>
<td>88.67</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentence Approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN+SLL+LM2</td>
<td>97.12</td>
<td>93.37</td>
<td>88.78</td>
<td>74.15</td>
</tr>
</tbody>
</table>

- Sentence approach (CNNs) is comparable to window approach (FFNNs) except for SRL where they claim it works much better.

Collobert and Weston 2008, 2011
Neural CRFs with LSTMs

- Neural CRF using character LSTMs to compute word representations

Chiu and Nichols (2015), Lample et al. (2016)
Neural CRFs with LSTMs

- Chiu+Nichols: character CNNs instead of LSTMs
- Lin/Passos/Luo: use external resources like Wikipedia
- LSTM-CRF captures the important aspects of NER: word context (LSTM), sub-word features (character LSTMs), outside knowledge (word embeddings)

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collobert et al. (2011)*</td>
<td>89.59</td>
</tr>
<tr>
<td>Lin and Wu (2009)</td>
<td>83.78</td>
</tr>
<tr>
<td>Lin and Wu (2009)*</td>
<td>90.90</td>
</tr>
<tr>
<td>Huang et al. (2015)*</td>
<td>90.10</td>
</tr>
<tr>
<td>Passos et al. (2014)</td>
<td>90.05</td>
</tr>
<tr>
<td>Passos et al. (2014)*</td>
<td>90.90</td>
</tr>
<tr>
<td>Luo et al. (2015)* + gaz</td>
<td>89.9</td>
</tr>
<tr>
<td>Luo et al. (2015)* + gaz + linking</td>
<td>91.2</td>
</tr>
<tr>
<td>Chiu and Nichols (2015)</td>
<td>90.69</td>
</tr>
<tr>
<td>Chiu and Nichols (2015)*</td>
<td>90.77</td>
</tr>
<tr>
<td>LSTM-CRF (no char)</td>
<td>90.20</td>
</tr>
<tr>
<td>LSTM-CRF</td>
<td>90.94</td>
</tr>
</tbody>
</table>

Chiu and Nichols (2015), Lample et al. (2016)
Takeaways

- CNNs are a flexible way of extracting features analogous to bag of n-grams, can also encode positional information.

- All kinds of NNs can be integrated into CRFs for structured inference. Can be applied to NER, other tagging, parsing, ...

- This concludes the ML/DL-heavy portion of the course. Starting Tuesday: syntax, then semantics.