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
### Convolutions for NLP

- **Input and filter are 2-dimensional instead of 3-dimensional**
  - sentence: n words x k vec dim
  - filter: m x k
  - activations: (n - m + 1) x 1

- **Combines evidence locally in a sentence and produces a new (but still variable-length) representation**

- **Images:** RGB values (3 dim); text: word vector (50+ dim)

### 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
**CNNs for Sentiment Analysis**

- **$P(y|x)$**
  - projection + softmax
  - $c$-dimensional vector
  - max pooling over the sentence

- **$n \times c$**
  - $c$ filters,
  - $m \times k$ each

- **Max pooling**: return the max activation of a given filter over the entire sentence; like a logical OR (sum pooling is like logical AND)

---

**Understanding CNNs for Sentiment**

- **“good” filter output**
  - $\text{max} = 1.1$

- **Filter “looks like” the things that will cause it to have high activation**

---

- **“bad”**
  - $0.1$

- **“okay”**
  - $0.3$

- **“terrible”**
  - $0.1$
Understanding CNNs for Sentiment

- The movie was good.
- Features for classification layer (or more NN layers):
  - max = 1.1

- "bad" -> 0.1
- Filters are initialized randomly and then learned
- Takes variable-length input and turns it into fixed-length output

Understanding CNNs for Sentiment

- The movie was good.
- Features for classification layer (or more NN layers):
  - max = 1.8

- "not good" -> 1.5
- Word vectors for similar words are similar, so convolutional filters will have similar outputs

Understanding CNNs for Sentiment

- The movie was not good.
- "not good" -> 1.5
- 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 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

  ![Layer 2](image)

  Zeiler and Fergus (2014)

- High-level filters: match larger and more “semantic patterns”

  ![image](image)

  Zeiler and Fergus (2014)

---

CNNs: Implementation

- 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)

![Diagram](image)

Kim (2014)
Sentence Classification

Also effective at document-level text classification

Kim (2014)

Neural CRF Basics

NER Revisited

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 conjuctions as effectively (can’t look at more than 2 words with a single feature)

LSTMs for NER

Transducer (LM-like model)

What are the strengths and weaknesses of this model compared to CRFs?
LSTMs for NER

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

Neural CRFs

- 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: \(\phi_e(y_i, i, x) = w^T f_e(y_i, i, x)\)
- Neural: \(\phi_e(y_i, i, x) = W_{y_i} f(i, x)\) \(W\) 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: \(\phi_e(y_i, i, x) = w^T f_e(y_i, i, x)\)
- Neural: \(\phi_e(y_i, i, x) = W_{y_i} f(i, x)\)
- \(\frac{\partial L}{\partial \phi_{e,i}} = -P(y_i = s|x) + I[s \text{ is gold}]\) “error signal”, compute with F-B
- Chain rule say to multiply together, gives our update
- For linear model: \(\frac{\partial \phi_e}{\partial w_i} = f_{e,i}(y_i, i, x)\)
- For neural model: compute gradient of \(\phi\) w.r.t. parameters of neural net
Neural CRFs

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

B-PER I-PER O O O B-LOC O O O B-ORG O O

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

FFNN Neural CRF for NER

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

B-PER I-PER O O O B-LOC O O O B-ORG O O

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

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

LSTM Neural CRFs

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

B-PER I-PER O O O B-LOC O O O B-ORG O O

Bidirectional LSTMs compute emission (or transition) potentials

LSTMs for NER

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

B-PER I-PER O O O B-LOC O O O B-ORG O O

How does this compare to neural CRF?
“NLP (Almost) From Scratch”

<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>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

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>97.20</td>
<td>93.63</td>
<td>88.67</td>
<td>–</td>
</tr>
<tr>
<td>Sentence Approach</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>F₁</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><strong>90.94</strong></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