CS395T: Structured Models for NLP
Lecture 17: CNNs

Project 2 Results

Top 3 scores:
- Su Wang: 90.13 UAS
  Greedy logistic regression with extended feature set, trained for 30 epochs with Adagrad with a weight decay schedule
- Yasumasa Onoe: 89.58 UAS
  Greedy averaged perceptron, features looking at children + grandchildren on the stack, also three-way conjunctive POS features
- Prateek Shrishail Kolhar: 89.42 UAS
  Global model with beam size 5 + averaged perceptron, feature engineering with distance, valency, etc.

Recall: Seq2seq Models

- Generate next word conditioned on previous word as well as hidden state
- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state
- Need to actually evaluate computation graph up to this point to form input for the next state
- Decoder is advanced one state at a time until [STOP] is reached

Project 2 Results

Subham Ghosh
- Model averaging helps a lot
- LR better than SVM for many students

Tanya Goyal
- Other transition systems usually better than arc-standard

- Greedy arc standard
- Greedy arc EAGER
- Greedy arc Hybrid

Le film était bon
the movie was great
Recall: Seq2seq Training

- Objective: maximize $\log P(w^*_i | x, w^*_{i-1})$
- One loss term for each target-sentence word, feed the correct word regardless of model's prediction
- Length of gold sequence is known, can run the whole encoder-decoder in one computation graph and compute losses

Recall: Attention

- For each decoder state, compute a weighted sum of input states reflecting what's most important right now

\[
\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_j \exp(e_{ij})}
\]

\[
c_i = \sum_j \alpha_{ij} h_j
\]

Unnormalized scalar weight

Normalized scalar weight

Weighted sum of input hidden states (vector)

This Lecture

- Other RNN applications (finish up)
- CNNs
- CNNs for Sentiment
- Dilated CNNs for MT

Other RNN Applications
Parsing

- Parsing: input is a sentence, output is a bracketed sentence

  (S (NP (DT the) (NN movie)) ...)
  the movie was good

- Attention is essential: <70 F1 without it, 88.3 F1 / 90.5 F1 (ensemble) with it
- The best parsers still use some structure — we’ll come back to these

Vinyals et al. (2014)

Summarization

- Summarization/compression: input is an article/sentence, output is a summary of the input

  On Friday, the U.S. intends to announce...
  U.S. to lift sanctions Friday

- Long articles, hard to deal with even with attention
- Speech recognition/text-to-speech: neural nets are good at dealing with continuous speech signals!

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 x n x k  
filter: m x m x k

sum over dot products

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)

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

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

\[
\text{image: } n \times n \times k \quad \text{filter: } m \times m \times k \quad \text{activations: } (n - m + 1) \times (n - m + 1) \times 1
\]

- “Narrow convolution” reduces input size, but can also preserve it.

**Convolutions for NLP**

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

\[
\text{sentence: } n \text{ words} \times k \text{ vec dim} \quad \text{filter: } m \times k \quad \text{activations: } (n - m + 1) \times 1
\]

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

- “Narrow convolution” reduces input size, but can also preserve it.

**Compare: LSTMs vs. CNNs**

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

**CNNs for Sentiment**

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

\[
\text{sentence: } n \text{ words} \times k \text{ vec dim} \quad \text{filter: } m \times k \quad \text{activations: } (n - m + 1) \times 1
\]

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

- “Narrow convolution” reduces input size, but can also preserve it.
CNNs for Sentiment Analysis

\[ P(\hat{y}|x) \]

projection + softmax

c-dimensional vector

max pooling over the sentence

\[ n \times c \]

c filters, \n \[ m \times k \text{ each} \n \]

\[ n \times k \]

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

the movie was good

Understanding CNNs for Sentiment

\[ \text{"good" filter output} \]

\[ \max = 1.1 \]

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

Features for classification layer (or more NN layers)

\[ \text{max} = 1.1 \]

Takes variable-length input and turns it into fixed-length output

Filters are initialized randomly and then learned
Word vectors for similar words are similar, so convolutional filters will have similar outputs.

Analogous to bigram features in bag-of-words models.

Indicator feature of text containing bigram $\langle\rightarrow\rangle$ max pooling of a filter that matches that bigram.

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

High-level filters: match larger and more "semantic patterns"
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)
- Filters are initialized randomly, need to learn to pick up on appropriate patterns
- 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

- 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>85.1</td>
<td>93.2</td>
<td>92.2</td>
<td>85.0</td>
<td>80.4</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>

Entity Linking

- CNNs can produce good representations of both sentences and documents like typical bag-of-words features
- Can distill topic representations for use in entity linking

Kim (2014)

- that they had disqualified Armstrong from his seven consecutive
- Armstrong County

- Lance Armstrong

Cycling domain

Geopolitical domain
Entity Linking

Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified Armstrong from his seven consecutive Tour de France wins from 1999–2005.

Armstrong County is a county in Pennsylvania...

Lance Edward Armstrong is an American former professional road cyclist

\[ s_{\text{Lance}} = d \cdot a_{\text{Lance}} \]

\[ s_{\text{County}} = d \cdot a_{\text{County}} \]

\[ P(y|x) = \text{softmax}(s) \]

Francis-Landau et al. (2016)

Dilated CNNs for MT

- Dilated CNNs for Machine Translation
  - “ByteNet”: operates over characters (bytes)
  - Encode source sequence w/dilated convolutions
  - Predict nth target character by looking at the nth position in the source and a dilated convolution over the n-1 target tokens so far
  - To deal with divergent lengths, \( t_\alpha \) actually looks at \( s_{\text{neg}} \) where \( \alpha \) is a heuristically-chosen parameter
  - Assumes mostly monotonic translation

- CNNs for Machine Translation
  - “ByteNet”: operates over characters (bytes)
  - Encode source sequence w/dilated convolutions
  - Predict nth target character by looking at the nth position in the source and a dilated convolution over the n-1 target tokens so far
  - To deal with divergent lengths, \( t_\alpha \) actually looks at \( s_{\text{neg}} \) where \( \alpha \) is a heuristically-chosen parameter
  - Assumes mostly monotonic translation

- Dilated Convolutions
  - Standard convolution: looks at every token under the filter
  - Dilated convolution with gap \( d \): looks at every \( d \)th token
  - Can chain successive dilated convolutions together to get a wide receptive field (see a lot of the sentence)
  - Top nodes see lots of the sentence, but with different processing

Strubell et al. (2017)

Kalchbrenner et al. (2016)
Compare: CNNs vs. LSTMs

CNN: source encoding at this position gives us "attention", target encoding gives us decoder context

LSTM: looks at previous word + hidden state, attention over input

Kalchbrenner et al. (2016)

Attention from CNN

- Model is character-level, this visualization shows which words’ characters impact the convolutional encoding the most
- Largely monotonic but does consult other information

Kalchbrenner et al. (2016)

Advantages of CNNs

- LSTM with attention is quadratic: compute attention over the whole input for each decoded token
- CNN is linear!
- CNN is shallower too in principle but the conv layers are very sophisticated (3 layers each)

Kalchbrenner et al. (2016)

English-German MT Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Inputs</th>
<th>Outputs</th>
<th>WMT Test '14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrase Based MT (Freitag et al., 2014; Williams et al., 2015)</td>
<td>phrases</td>
<td>phrases</td>
<td>20.7</td>
</tr>
<tr>
<td>RNN Enc-Dec (Luong et al., 2015)</td>
<td>words</td>
<td>words</td>
<td>11.3</td>
</tr>
<tr>
<td>Reverse RNN Enc-Dec (Luong et al., 2015)</td>
<td>words</td>
<td>words</td>
<td>14.0</td>
</tr>
<tr>
<td>RNN Enc-Dec Att (Zhou et al., 2016)</td>
<td>words</td>
<td>words</td>
<td>20.6</td>
</tr>
<tr>
<td>RNN Enc-Dec Att (Luong et al., 2015)</td>
<td>words</td>
<td>words</td>
<td>20.9</td>
</tr>
<tr>
<td>GNMT (RNN Enc-Dec Att) (Wu et al., 2016a)</td>
<td>word-pieces</td>
<td>word-pieces</td>
<td>24.61</td>
</tr>
<tr>
<td>RNN Enc-Dec Att (Chung et al., 2016b)</td>
<td>BPE</td>
<td>BPE</td>
<td>19.98</td>
</tr>
<tr>
<td>RNN Enc-Dec Att (Chung et al., 2016b)</td>
<td>BPE</td>
<td>char</td>
<td>21.33</td>
</tr>
<tr>
<td>GNMT (RNN Enc-Dec Att) (Wu et al., 2016a)</td>
<td>char</td>
<td>char</td>
<td>22.62</td>
</tr>
<tr>
<td>ByteNet</td>
<td>char</td>
<td>char</td>
<td>23.75</td>
</tr>
</tbody>
</table>

Kalchbrenner et al. (2016)
Up Next

- Next lecture: Ye will talk about using neural networks in lower-resource settings
- After that: advanced neural network structures
  - Tree-structured RNNs
  - Neural CRFs
  - Memory networks, etc.