165B
Machine Learning
Sequence to Sequence Learning

Lei Li (leili@cs)
UCSB
Acknowledgement: Slides borrowed from Bhiksha Raj’s 11485 and Mu Li & Alex Smola’s 157 courses on Deep Learning, with modification
Recap

- Vocabulary building
  - Subword Tokenization
    - No OOV.
- Language Model
  \[ P(y) = \prod_{t} P(y_{t+1} | y_1 \ldots y_t) \]
- Word Embedding
- CNN-Language model
- Recurrent neural network
  - memory
  - Long-short term memory
Language Modeling

• Given a sentence $y$, estimate the probability

$$P(y) = \prod_{t} P(y_{t+1} | y_1 \ldots y_t)$$

$$P(y_{t+1} | y_1 \ldots y_t) = f_{\theta}(y_1, \ldots, y_t)$$

The cat sits on a __

$Y_1 \ Y_2 \ Y_3 \ Y_4 \ Y_5 \ Y_6$

<table>
<thead>
<tr>
<th>Word</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>mat</td>
<td>0.15</td>
</tr>
<tr>
<td>rug</td>
<td>0.13</td>
</tr>
<tr>
<td>chair</td>
<td>0.08</td>
</tr>
<tr>
<td>hat</td>
<td>0.05</td>
</tr>
<tr>
<td>dog</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Recurrent Neural Network

\[
p(x_t | x_1, \ldots, x_{t-1}) = \text{softmax}(U \cdot h_t)
\]

\[
h_t = \sigma \left( W \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b \right)
\]


Long-Short Term Memory (LSTM)

- Replace cell with more advanced one
- Adaptively memorize short and long term information

\[
i_{t+1} = \sigma(M_{ix}x_{t+1} + M_{ih}h_t + b_i)
\]
\[
f_{t+1} = \sigma(M_{fx}x_{t+1} + M_{fh}h_t + b_f)
\]
\[
o_{t+1} = \sigma(M_{ox}x_{t+1} + M_{oh}h_t + b_o)
\]
\[
a_{t+1} = \tanh(M_{cx}x_{t+1} + M_{ch}h_t + b_a)
\]
\[
c_{t+1} = f_{t+1} \otimes c_t + i_{t+1} \otimes a_{t+1}
\]
\[
h_{t+1} = o_{t+1} \otimes \tanh(c_{t+1})
\]

Hochreiter & Schmidhuber. Long Short-Term Memory, 1997
Gers et al. Learning to Forget: Continual Prediction with LSTM. 2000
LSTM Language Modelling

\[
\text{A cat sits on a mat}. 
\]

Embedding → LSTM → Linear → Softmax
LSTM Generation

A cat sits on a mat.

[Embedding] → LSTM → Linear → Softmax
LSTM: More layers

A cat sits on a mat.

[BOS] A cat sits on a
Expressive Power of RNN-LM

Perplexity:

\[ PPL = P(x_1, \ldots, x_N)^{-\frac{1}{N}} = \exp(-\frac{1}{N} \sum_{n=1}^{N} \log P(x_n | x_1 \ldots x_{n-1})) \]

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Perplexity</th>
<th>Number of Params [billions]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid-RNN-2048 (Ji et al., 2015a)</td>
<td>68.3</td>
<td>4.1</td>
</tr>
<tr>
<td>Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)</td>
<td>67.6</td>
<td>1.76</td>
</tr>
<tr>
<td>Sparse Non-Negative Matrix LM (Shafer et al., 2015)</td>
<td>52.9</td>
<td>33</td>
</tr>
<tr>
<td>RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)</td>
<td>51.3</td>
<td>20</td>
</tr>
<tr>
<td>LSTM-512-512</td>
<td>54.1</td>
<td>0.82</td>
</tr>
<tr>
<td>LSTM-1024-512</td>
<td>48.2</td>
<td>0.82</td>
</tr>
<tr>
<td>LSTM-2048-512</td>
<td>43.7</td>
<td>0.83</td>
</tr>
<tr>
<td>LSTM-8192-2048 (No Dropout)</td>
<td>37.9</td>
<td>3.3</td>
</tr>
<tr>
<td>LSTM-8192-2048 (50% Dropout)</td>
<td>32.2</td>
<td>3.3</td>
</tr>
<tr>
<td>2-Layer LSTM-8192-1024 (BIG LSTM)</td>
<td>30.6</td>
<td>1.8</td>
</tr>
<tr>
<td>BIG LSTM+CNN Inputs</td>
<td>30.0</td>
<td>1.04</td>
</tr>
<tr>
<td>BIG LSTM+CNN Inputs + CNN Softmax</td>
<td>39.8</td>
<td>0.29</td>
</tr>
<tr>
<td>BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction</td>
<td>35.8</td>
<td>0.39</td>
</tr>
<tr>
<td>BIG LSTM+CNN Inputs + Char LSTM predictions</td>
<td>47.9</td>
<td>0.23</td>
</tr>
</tbody>
</table>

Sequence Labelling
Understanding Query Intention

Noodle house near Santa Barbara

How to go from Santa Barbara to Los Angeles?

Sequence Labelling
Named entity recognition

date
In **April 1775** fighting broke out between **Massachusetts** militia units and **British** regulars at **Lexington** and **Concord**. 

Geo-Political
Sequence Labelling

- Named entity recognition
  In April 1775 fighting broke out between Massachusetts militia units and British regulars at Lexington and Concord.

- Semantic role labeling
  The excess supply pushed gasoline prices down in that period.

- Question Answering: subject parsing
  Who created Harry Potter?
BIO scheme

O O O B-GPE I-GPE O B-PER I-PER O

The governor of Santa Barbara is Cathy Murillo.
RNN/LSTM for Sequence Labelling

The governor of Santa Barbara is Cathy Murillo.

1640 897 45 1890 78 943 3521 782 533
The governor of Santa Barbara is Cathy Murillo.
Twisted NN for NER

Chinese NER
OntoNotes Data 4-class:

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-NER-WA*</td>
<td>84.42</td>
<td>76.34</td>
<td>80.18</td>
</tr>
<tr>
<td>Wang et al.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNN-2b with WS ours</td>
<td>84.75</td>
<td>77.85</td>
<td>81.15</td>
</tr>
</tbody>
</table>

* Wang et al used bilingual data

OntoNotes Data 18-class:

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sameer Pradhan et al.</td>
<td>78.20</td>
<td>66.45</td>
<td>71.85</td>
</tr>
<tr>
<td>RNN-2b with WS ours</td>
<td>78.69</td>
<td>70.54</td>
<td>74.39</td>
</tr>
</tbody>
</table>

Twisted NN [Zefu Lu, Lei Li, Wei Xu, 2015]
Twisted NN

[Lu, Li, Xu, 2015]

State-of-the-art result: F1 74.39% on ontonotes-5.0 18-class data. (Chinese)
Sequence Labelling using LSTM (Pytorch)

class LSTMTagger(nn.Module):
    def __init__(self, embedding_dim, hidden_dim, vocab_size, tagset_size):
        super(LSTMTagger, self).__init__()
        self.hidden_dim = hidden_dim

        self.word_embeddings = nn.Embedding(vocab_size, embedding_dim)

        # The LSTM takes word embeddings as inputs, and outputs hidden states
        # with dimensionality hidden_dim.
        self.lstm = nn.LSTM(embedding_dim, hidden_dim)

        # The linear layer that maps from hidden state space to tag space
        self.hidden2tag = nn.Linear(hidden_dim, tagset_size)

    def forward(self, sentence):
        embeds = self.word_embeddings(sentence)
        lstm_out, _ = self.lstm(embeds.view(len(sentence), 1, -1))
        tag_space = self.hidden2tag(lstm_out.view(len(sentence), -1))
        tag_scores = F.log_softmax(tag_space, dim=1)
        return tag_scores
model = LSTMTagger(EMBEDDING_DIM, HIDDEN_DIM, len(word_to_ix), len(tag_to_ix))
loss_function = nn.NLLLoss()
optimizer = optim.SGD(model.parameters(), lr=0.1)

# See what the scores are before training
# Note that element i, j of the output is the score for tag j for word i.
# Here we don't need to train, so the code is wrapped in torch.no_grad()
with torch.no_grad():
    inputs = prepare_sequence(training_data[0][0], word_to_ix)
    tag_scores = model(inputs)
    print(tag_scores)

for epoch in range(300):  # again, normally you would NOT do 300 epochs, it is toy data
    for sentence, tags in training_data:
        # Step 1. Remember that Pytorch accumulates gradients.
        # We need to clear them out before each instance
        model.zero_grad()

        # Step 2. Get our inputs ready for the network, that is, turn them into
        # Tensors of word indices.
        sentence_in = prepare_sequence(sentence, word_to_ix)
        targets = prepare_sequence(tags, tag_to_ix)

        # Step 3. Run our forward pass.
        tag_scores = model(sentence_in)

        # Step 4. Compute the loss, gradients, and update the parameters by
        # calling optimizer.step()
        loss = loss_function(tag_scores, targets)
        loss.backward()
        optimizer.step()
with torch.no_grad():
    inputs = prepare_sequence(training_data[0][0], word_to_ix)
    tag_scores = model(inputs)
Better Loss Function (advanced)

- Loss using Conditional Random Fields

\[- \log(P(y \mid X)) = - \log \left( \frac{\exp \left( \sum_{k=1}^{\ell} U(x_k, y_k) + \sum_{k=1}^{\ell-1} T(y_k, y_{k+1}) \right)}{Z(X)} \right)\]

\[= \log(Z(X)) - \log \left( \exp \left( \sum_{k=1}^{\ell} U(x_k, y_k) + \sum_{k=1}^{\ell-1} T(y_k, y_{k+1}) \right) \right)\]

\[= \log(Z(X)) - \left( \sum_{k=1}^{\ell} U(x_k, y_k) + \sum_{k=1}^{\ell-1} T(y_k, y_{k+1}) \right)\]

\[= Z \log(X) - \left( \sum_{k=1}^{\ell} U(x_k, y_k) + \sum_{k=1}^{\ell-1} T(y_k, y_{k+1}) \right)\]
Encoder-decoder framework

Decoder

Encoder

input

output

A generic formulation
ImageCaption
Text-to-Image Generation
ASR (speech-to-text)
MT (text-to-text)
Sequence To Sequence (Seq2seq)

- Machine translation as directly learning a function mapping from source sequence to target sequence

Source: 天气

Encoder: LSTM

e_a e_b e_c e_d

Decoder: LSTM

t_1 t_2 t_3 t_4

h_1 h_2 h_3 h_4

The weather is nice

Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014
Sequence To Sequence (Seq2seq)

- Machine translation as directly learning a function mapping from source sequence to target sequence

Encoder: LSTM
Decoder: LSTM

Source: 天气很好

Target: The weather is nice

Training loss: Cross-Entropy

\[ P(Y|X) = \prod P(y_t|y_{<t}, x) \]

\[ l = - \sum_n \sum_t \log f_\theta(x_n, y_{n,1}, \ldots, y_{n,t-1}) \]

Teacher-forcing during training.
(prepret to know groundtruth for prefix)

Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014
Performance (2014)

Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014
Stacked LSTM for seq-2-seq

- More layers of LSTM
Attention
LSTM Seq2seq with Attention

Bahdanau et al., Neural Machine Translation by Jointly Learning to Align and Translate. 2015
A context vector $c_n$ will be predicted before, which represents the related source context for current predicted word.

$$\alpha_{nj} = \text{Softmax}(D(s_n, h_{1\ldots n-1})) = \frac{\exp(D(s_n, h_j))}{\sum_k \exp(D(s_n, h_k))}$$

$$c_n = \sum_j \alpha_{nj} h_j$$

The probability of word $y_i$ is computed as:

$$p(y_i) \propto \exp(Wh_i) \quad \Rightarrow \quad p(y_i) \propto \exp(Wh_i + Vc_i)$$

Decoding
Autoregressive Generation

greedy decoding: output the token with max next token prob

But, this is not necessary the best
Inference

• Now already trained a model \( \theta \)
• Decoding/Generation: Given an input sentence \( x \), to generate the target sentence \( y \) that maximize the probability \( P(y \mid x; \theta) \)
  \[
  \arg\max_y P(y \mid x) = f_\theta(x, y)
  \]
• Two types of error
  – the most probable translation is bad \( \rightarrow \) fix the model
  – search does not find the most probably translation \( \rightarrow \) fix the search
• Most probable translation is not necessary the highest BLEU one!
Decoding

- \( \text{argmax } P(y \mid x) = f_\theta(x, y) \)
- naive solution: exhaustive search
  - too expensive
- Beam search
  - (approximate) dynamic programming
Beam Search

- start with empty S
- at each step, keep k best partial sequences
- expand them with one more forward generation
- collect new partial results and keep top-k
best_scores = []
add {[0], 0.0} to best_scores # 0 is for beginning of sentence token
for i in 1 to max_length:
    new_seqs = PriorityQueue()
    for (candidate, s) in best_scores:
        if candidate[-1] is EOS:
            prob = all -inf
            prob[EOS] = 0
        else:
            prob = using model to take candidate and compute next token probabilities (logp)
        pick top k scores from prob, and their index
        for each score, index in the top-k of prob:
            new_candidate = candidate.append(index)
            new_score = s + score
        if not new_seqs.full():
            new_seqs.put((new_score, new_candidate))
Beam Search

forward by network

<BLANK>

We 0.3

He 0.1

She 0.1

They 0.01

forward by network

I 0.4

We 0.3

He 0.1

She 0.1

They 0.01

like 0.4
love 0.4
am 0.1
hate 0.01
want 0.01

I like 0.16
love 0.16

We like 0.12
do 0.3
are 0.2
can 0.01
say 0.01

forward by network

I like 0.16
love 0.16

We like 0.12
We do 0.09

forward by network

singing 0.6
song 0.2
shouting 0.01
going 0.01
dancing 0.01

forward by network

I like singing 0.096
I love singing 0.08
I like dancing 0.048

I like song 0.032
I love dancing 0.048

I love singing 0.08
I love dancing 0.048
Seq2seq for Machine Translation
SpaceX launched a mission Wednesday night to put four amateurs with no space experience into orbit.

SpaceX conducted a launch mission on Wednesday night, sending four amateurs with no aerospace experience into space orbit.

SpaceX conducted a launch mission Wednesday night that sent four amateurs with no spaceflight experience into orbit.

SpaceX carried out a launch mission on Wednesday night to put four amateurs without Aerospace experience into orbit.
• Measuring the precision of n-grams
  – Precision of n-gram: percentage of tokens in output sentences
    \[ p_n = \frac{\text{num. of correct token ngram}}{\text{total output ngram}} \]

• Penalize for brevity
  – if output is too short
    \[ b_p = \min(1, e^{1-r/c}) \]

• BLEU=\( b_p \cdot (\prod p_i)^{\frac{1}{4}} \)

• Notice BLEU is computed over the whole corpus, not on one sentence
Ref: A SpaceX rocket was launched into a space orbit Wednesday evening.
System A: SpaceX launched a mission Wednesday evening into a space orbit.
System B: A rocket sent SpaceX into orbit Wednesday.
Ref: A SpaceX rocket was launched into a space orbit Wednesday evening.

System A: SpaceX launched a mission Wednesday evening into a space orbit.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigram</td>
<td>9/11</td>
</tr>
<tr>
<td>Bigram</td>
<td>4/10</td>
</tr>
<tr>
<td>Trigram</td>
<td>2/9</td>
</tr>
<tr>
<td>Four-gram</td>
<td>1/8</td>
</tr>
</tbody>
</table>

\[
bp = e^{1-12/11} = 0.91
\]

\[
\text{BLEU} = 0.91 \times (9/11 \times 4/10 \times 2/9 \times 1/8)^{1/4}
\]

\[
= 28.1\%
\]
Jean et al. On Using Very Large Target Vocabulary for Neural Machine Translation. 2015
Performance with Model Ensemble

Luong et al. Effective Approaches to Attention-based Neural Machine Translation. 2015
Summary

• LSTM-RNN Language Modelling
• Sequence Labelling
  – named entity recognition/semantic role labelling/POS tagging
  – Bi-LSTM
• Seq2seq
  – End-to-end model for Machine Translation
Next Up

• Transformer