CS11-747 Neural Networks for NLP

Recurrent Neural Networks

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

Site
https://phontron.com/class/nn4nlp2017/
NLP and Sequential Data
NLP and Sequential Data

• NLP is full of sequential data
NLP and Sequential Data

- NLP is full of sequential data
  - Words in sentences
NLP and Sequential Data

- NLP is full of sequential data
  - Words in sentences
  - Characters in words
NLP and Sequential Data

- NLP is full of sequential data
  - Words in sentences
  - Characters in words
  - Sentences in discourse
NLP and Sequential Data

- NLP is full of sequential data
  - Words in sentences
  - Characters in words
  - Sentences in discourse
  - ...

Long-distance Dependencies in Language
Long-distance Dependencies in Language

• Agreement in number, gender, etc.
Long-distance Dependencies in Language

- Agreement in number, gender, etc.

He does not have very much confidence in himself. She does not have very much confidence in herself.
Long-distance Dependencies in Language

• Agreement in number, gender, etc.

He does not have very much confidence in **himself**.
She does not have very much confidence in **herself**.

• Selectional preference
Long-distance Dependencies in Language

• Agreement in number, gender, etc.

  He does not have very much confidence in himself. She does not have very much confidence in herself.

• Selectional preference

  The reign has lasted as long as the life of the queen. The rain has lasted as long as the life of the clouds.
Can be Complicated!
Can be Complicated!

• What is the referent of “it”? 
Can be Complicated!

• What is the referent of “it”?

The trophy would not fit in the brown suitcase because it was too big.
Can be Complicated!

• What is the referent of “it”? The trophy would not fit in the brown suitcase because it was too big. Trophy
Can be Complicated!

- What is the referent of “it”?

The trophy would not fit in the brown suitcase because it was too big.

The trophy would not fit in the brown suitcase because it was too small.
Can be Complicated!

• What is the referent of “it”?

The trophy would not fit in the brown suitcase because it was too big.

Trophy

The trophy would not fit in the brown suitcase because it was too small.

Suitcase
Can be Complicated!

- What is the referent of “it”?

The trophy would not fit in the brown suitcase because it was too **big**.

  Trophy

The trophy would not fit in the brown suitcase because it was too **small**.

  Suitcase

(from Winograd Schema Challenge: http://commonsensereasoning.org/winograd.html)
Recurrent Neural Networks
(Elman 1990)
Recurrent Neural Networks
(Elman 1990)

• Tools to “remember” information
Recurrent Neural Networks
(Elman 1990)

- Tools to “remember” information

Feed-forward NN
Recurrent Neural Networks
(Elman 1990)

• Tools to “remember” information
Unrolling in Time

• What does processing a sequence look like?

I ▼

hate ▼

this ▼

movie ▼
Unrolling in Time

• What does processing a sequence look like?

I  hate  this  movie
Unrolling in Time

• What does processing a sequence look like?

I → hate → this → movie

RNN
Unrolling in Time

• What does processing a sequence look like?

RNN

predict

label

I

hate

this

movie
Unrolling in Time

• What does processing a sequence look like?

RNN

I

hate

this

movie

RNN

predict

label
Unrolling in Time

• What does processing a sequence look like?

RNN \rightarrow \text{predict} \rightarrow \text{label} \rightarrow \text{hate} \rightarrow \text{predict} \rightarrow \text{label} \rightarrow \text{this} \rightarrow \text{Movie}
Unrolling in Time

- What does processing a sequence look like?

<table>
<thead>
<tr>
<th>I</th>
<th>hate</th>
<th>this</th>
<th>movie</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>RNN</td>
<td>RNN</td>
<td></td>
</tr>
<tr>
<td>predict</td>
<td>predict</td>
<td></td>
<td></td>
</tr>
<tr>
<td>label</td>
<td>label</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unrolling in Time

• What does processing a sequence look like?

I, hate, this, movie

RNN → predict
→ label

RNN → predict
→ label

RNN → predict
→ label
Unrolling in Time

• What does processing a sequence look like?

I
hate
this
movie

RNN → RNN → RNN → RNN

predict → predict → predict → predict

label → label → label → label
Unrolling in Time

• What does processing a sequence look like?
Training RNNs

I hate this movie

RNN RNN RNN RNN

predict predict predict predict

prediction 1 prediction 2 prediction 3 prediction 4
Training RNNs

I hate this movie

RNN RNN RNN RNN

predict predict predict predict

prediction 1 prediction 2 prediction 3 prediction 4

label 1 label 2 label 3 label 4
Training RNNs

I hate this movie

RNN predict

prediction 1
loss 1
label 1

RNN predict

prediction 2
label 2

RNN predict

prediction 3
label 3

RNN predict

prediction 4
label 4
Training RNNs

I hate this movie

RNN → predict

prediction 1

loss 1

label 1

RNN → predict

prediction 2

loss 2

label 2

RNN → predict

prediction 3

label 3

RNN → predict

prediction 4

label 4
Training RNNs

I hate this movie

RNN predict prediction 1 label 1 loss 1
RNN predict prediction 2 label 2 loss 2
RNN predict prediction 3 label 3 loss 3
RNN predict prediction 4 label 4
Training RNNs

I

<table>
<thead>
<tr>
<th>RNN</th>
<th>predict</th>
<th>prediction 1</th>
<th>loss 1</th>
<th>label 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RNN</th>
<th>predict</th>
<th>prediction 2</th>
<th>loss 2</th>
<th>label 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RNN</th>
<th>predict</th>
<th>prediction 3</th>
<th>loss 3</th>
<th>label 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RNN</th>
<th>predict</th>
<th>prediction 4</th>
<th>loss 4</th>
<th>label 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Training RNNs

I hate this movie

RNN RNN RNN RNN

predict predict predict predict

prediction 1 prediction 2 prediction 3 prediction 4

loss 1 loss 2 loss 3 loss 4

label 1 label 2 label 3 label 4

sum
Training RNNs

Input sequence:
- I
- hate
- this
- movie

RNNs:
- Input to RNNs
- Predictions:
  - prediction 1
  - prediction 2
  - prediction 3
  - prediction 4

Losses:
- loss 1
- loss 2
- loss 3
- loss 4

Labels:
- label 1
- label 2
- label 3
- label 4

Total loss:
- sum → total loss
RNN Training
RNN Training

- The unrolled graph is a well-formed (DAG) computation graph—we can run backprop
RNN Training

- The unrolled graph is a well-formed (DAG) computation graph—we can run backprop

```
sum → total loss
```
RNN Training

- The unrolled graph is a well-formed (DAG) computation graph—we can run backprop.
RNN Training

• The unrolled graph is a well-formed (DAG) computation graph—we can run backprop

\[ \text{sum} \]

\[ \text{total loss} \]

• Parameters are tied across time, derivatives are aggregated across all time steps
RNN Training

- The unrolled graph is a well-formed (DAG) computation graph—we can run backprop

- Parameters are tied across time, derivatives are aggregated across all time steps

- This is historically called “backpropagation through time” (BPTT)
Parameter Tying

I hate this movie

RNN RNN RNN RNN

predict predict predict predict

prediction 1 prediction 2 prediction 3 prediction 4

loss 1 loss 2 loss 3 loss 4

label 1 label 2 label 3 label 4

sum

total loss
Parameter Tying

Parameters are shared! Derivatives are accumulated.

---

**I hate this movie**

RNN → RNN → RNN → RNN → RNN

predict → predict → predict → predict → predict

prediction 1 → prediction 2 → prediction 3 → prediction 4

loss 1 → loss 2 → loss 3 → loss 4

label 1 → label 2 → label 3 → label 4

sum → total loss
Applications of RNNs
What Can RNNs Do?
What Can RNNs Do?

• Represent a sentence
What Can RNNs Do?

• Represent a sentence
  • Read whole sentence, make a prediction
What Can RNNs Do?

• Represent a sentence
  • Read whole sentence, make a prediction
• Represent a context within a sentence
What Can RNNs Do?

- Represent a sentence
  - Read whole sentence, make a prediction
- Represent a context within a sentence
  - Read context up until that point
Representing Sentences

I hate this movie.
Representing Sentences

I hate this movie

predict

token

token

token

token
Representing Sentences

- Sentence classification
Representing Sentences

- Sentence classification
- Conditioned generation
Representing Sentences

- Sentence classification
- Conditioned generation
- Retrieval
Representing Contexts

I hate this movie
Representing Contexts

I hate this movie
Representing Contexts

- Tagging
Representing Contexts

- Tagging
- Language Modeling
Representing Contexts

• Tagging
• Language Modeling
• Calculating Representations for Parsing, etc.
e.g. Language Modeling

• Language modeling is like a tagging task, where each tag is the next word!
e.g. Language Modeling

Language modeling is like a tagging task, where each tag is the next word!
e.g. Language Modeling

- Language modeling is like a tagging task, where each tag is the next word!
e.g. Language Modeling

- Language modeling is like a tagging task, where each tag is the next word!
e.g. Language Modeling

- Language modeling is like a tagging task, where each tag is the next word!
• Language modeling is like a tagging task, where each tag is the next word!
e.g. Language Modeling

- Language modeling is like a tagging task, where each tag is the next word!
• Language modeling is like a tagging task, where each tag is the next word!
e.g. Language Modeling

- Language modeling is like a tagging task, where each tag is the next word!
e.g. Language Modeling

- Language modeling is like a tagging task, where each tag is the next word!
e.g. Language Modeling

- Language modeling is like a tagging task, where each tag is the next word!
Language modeling is like a tagging task, where each tag is the next word!
• Language modeling is like a tagging task, where each tag is the next word!
e.g. Language Modeling

- Language modeling is like a tagging task, where each tag is the next word!
Language modeling is like a tagging task, where each tag is the next word!
e.g. Language Modeling

- Language modeling is like a tagging task, where each tag is the next word!
Bi-RNNs

- A simple extension, run the RNN in both directions
Bi-RNNs

- A simple extension, run the RNN in both directions
Bi-RNNs

- A simple extension, run the RNN in both directions

I hate this movie
Bi-RNNs

- A simple extension, run the RNN in both directions

I hate this movie
Bi-RNNs

- A simple extension, run the RNN in both directions

I hate this movie

![Diagram of Bi-RNNs with RNNs, concatenations, and softmax layers.]

- PRN
- VB
Bi-RNNs

- A simple extension, run the RNN in both directions

I hate this movie

![Diagram](image_url)
Bi-RNNs

• A simple extension, run the RNN in both directions
Bi-RNNs

• A simple extension, run the RNN in both directions
Bi-RNNs

• A simple extension, run the RNN in both directions

I hate this movie
Let’s Try it Out!
Recurrent Neural Networks in DyNet
Recurrent Neural Networks in DyNet

• Based on “*Builder” class (*=SimpleRNN/LSTM)
Recurrent Neural Networks in DyNet

- Based on “*Builder” class (*=SimpleRNN/LSTM)

- Add parameters to model (once):

```python
# LSTM (layers=1, input=64, hidden=128, model)
RNN = dy.SimpleRNNBuilder(1, 64, 128, model)
```
Recurrent Neural Networks in DyNet

• Based on “*Builder” class (*=SimpleRNN/LSTM)

• Add parameters to model (once):

```python
# LSTM (layers=1, input=64, hidden=128, model)
RNN = dy.SimpleRNNBuilder(1, 64, 128, model)
```

• Add parameters to CG and get initial state (per sentence):

```python
s = RNN.initial_state()
```
Recurrent Neural Networks in DyNet

- Based on “*Builder” class (*=SimpleRNN/LSTM)

- Add parameters to model (once):

  ```python
  # LSTM (layers=1, input=64, hidden=128, model)
  RNN = dy.SimpleRNNBuilder(1, 64, 128, model)
  ```

- Add parameters to CG and get initial state (per sentence):

  ```python
  s = RNN.initial_state()
  ```

- Update state and access (per input word/character):

  ```python
  s = s.add_input(x_t)
  h_t = s.output()
  ```
RNNLM Example: Parameter Initialization

```python
# Lookup parameters for word embeddings
WORDS_LOOKUP = model.add_lookup_parameters((nwords, 64))

# Word-level RNN (layers=1, input=64, hidden=128, model)
RNN = dy.SimpleRNNBuilder(1, 64, 128, model)

# Softmax weights/biases on top of RNN outputs
W_sm = model.add_parameters((nwords, 128))
b_sm = model.add_parameters(nwords)
```
RNNLM Example:
Sentence Initialization

```python
# Build the language model graph
def calc_lm_loss(wids):
    dy.renew_cg()

    # parameters -> expressions
    W_exp = dy.parameter(W_sm)
    b_exp = dy.parameter(b_sm)

    # add parameters to CG and get state
    f_init = RNN.initial_state()

    # get the word vectors for each word ID
    wembs = [WORDS_LOOKUP[wid] for wid in wids]

    # Start the rnn by inputting "<s>"
    s = f_init.add_input(wembs[-1])
    ...
```
RNNLM Example:
Loss Calculation and State Update

```python
# process each word ID and embedding
losses = []
for wid, we in zip(wids, wembs):

    # calculate and save the softmax loss
    score = W_exp * s.output() + b_exp
    loss = dy.pickneglogsoftmax(score, wid)
    losses.append(loss)

    # update the RNN state with the input
    s = s.add_input(we)

# return the sum of all losses
return dy.esum(losses)
```
RNN Problems and Alternatives
Vanishing Gradient

- Gradients decrease as they get pushed back

\[
\frac{dl}{dh_0} = \text{tiny} \quad \frac{dl}{dh_1} = \text{small} \quad \frac{dl}{dh_2} = \text{med.} \quad \frac{dl}{dh_3} = \text{large}
\]
Vanishing Gradient

- Gradients decrease as they get pushed back

\[ \frac{dl}{dh_0} = \text{tiny} \quad \frac{dl}{dh_1} = \text{small} \quad \frac{dl}{dh_2} = \text{med.} \quad \frac{dl}{dh_3} = \text{large} \]

- Why? “Squashed” by non-linearities or small weights in matrices.
A Solution:
Long Short-term Memory
(Hochreiter and Schmidhuber 1997)
A Solution: Long Short-term Memory
(Hochreiter and Schmidhuber 1997)

• **Basic idea:** make additive connections between time steps
A Solution: Long Short-term Memory
(Hochreiter and Schmidhuber 1997)

- **Basic idea:** make additive connections between time steps
- Addition does not modify the gradient, no vanishing
A Solution:
Long Short-term Memory
(Hochreiter and Schmidhuber 1997)

- **Basic idea:** make additive connections between time steps
- Addition does not modify the gradient, no vanishing
- Gates to control the information flow
LSTM Structure

update $u$: what value do we try to add to the memory cell?
input $i$: how much of the update do we allow to go through?
output $o$: how much of the cell do we reflect in the next state?
Other Alternatives
Other Alternatives

- Lots of variants of LSTMs (Hochreiter and Schmidhuber, 1997)
Other Alternatives

• Lots of variants of LSTMs (Hochreiter and Schmidhuber, 1997)

• Gated recurrent units (GRUs; Cho et al., 2014)
Other Alternatives

• Lots of variants of LSTMs (Hochreiter and Schmidhuber, 1997)

• Gated recurrent units (GRUs; Cho et al., 2014)

• All follow the basic paradigm of “take input, update state”
Code Examples

sentiment-lstm.py
lm-lstm.py
Efficiency/Memory Tricks
Handling Mini-batching
Handling Mini-batching

- Mini-batching makes things much faster!
Handling Mini-batching

• Mini-batching makes things much faster!

• But mini-batching in RNNs is harder than in feed-forward networks
Handling Mini-batching

- Mini-batching makes things much faster!
- But mini-batching in RNNs is harder than in feed-forward networks
  - Each word depends on the previous word
Handling Mini-batching

• Mini-batching makes things much faster!

• But mini-batching in RNNs is harder than in feed-forward networks
  • Each word depends on the previous word
  • Sequences are of various length
Mini-batching Method

this is an example </s>
this is another </s>
Mini-batching Method

this is an example
this is another

Padding
Mini-batching Method

this is an example
this is another

Loss Calculation

Padding
Mini-batching Method

drawing of input sequences:
```
this is an example </s>
this is another </s> </s>
```

Loss Calculation

<table>
<thead>
<tr>
<th>Loss Calculation</th>
<th>Padding</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>1</code> <code>1</code> <code>1</code> <code>1</code></td>
<td><code>1</code> <code>0</code></td>
<td></td>
</tr>
</tbody>
</table>
Mini-batching Method

Loss Calculation

```
this is an example </s>
this is another </s> </s>
```

Padding

Mask
Mini-batching Method

Loss Calculation

Take Sum

this is an example </s>
this is another </s> </s>

Padding Mask

Take Sum
Mini-batching Method

Loss Calculation

Take Sum

(Or use DyNet automatic mini-batching, much easier but a bit slower)
Bucketing/Sorting
Bucketing/Sorting

• If we use sentences of different lengths, too much padding and sorting can result in decreased performance
Bucketing/Sorting

• If we use sentences of different lengths, too much padding and sorting can result in decreased performance

• To remedy this: sort sentences so similarly-lengthed sentences are in the same batch
Code Example

lm-minibatch.py
Handling Long Sequences
Handling Long Sequences

- Sometimes we would like to capture long-term dependencies over long sequences
Handling Long Sequences

• Sometimes we would like to capture long-term dependencies over long sequences

• e.g. words in full documents
Handling Long Sequences

• Sometimes we would like to capture long-term dependencies over long sequences

• e.g. words in full documents

• However, this may not fit on (GPU) memory
Truncated BPTT

• Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass
I hate this movie

2nd Pass
It is so bad
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass  

|   | hate | this | movie |

2nd Pass

| It | is | so | bad |
Truncated BPTT

• Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

```
I
```

```
hate
```

```
this
```

```
movie
```

2nd Pass

```
It
```

```
is
```

```
so
```

```
bad
```
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

- I hate this movie

2nd Pass

- It is so bad
Truncated BPTT

• Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

2nd Pass

I hate this movie

It is so bad
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

2nd Pass
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

2nd Pass

I hate this movie

It is so bad
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

I hate this movie

2nd Pass

It is so bad
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

2nd Pass

I hate this movie

It is so bad
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

2nd Pass

It is so bad
Truncated BPTT

- Backprop over shorter segments, initialize with the state from the previous segment

1st Pass

1st Pass

2nd Pass

state only, no backprop

It is so bad
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

2nd Pass state only, no backprop
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

**1st Pass**

```
I
hate
this
movie
```

**2nd Pass**

```
It
is
so
bad
```

state only, no backprop
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

<table>
<thead>
<tr>
<th></th>
<th>RNN</th>
<th></th>
<th>RNN</th>
<th></th>
<th>RNN</th>
<th></th>
<th>RNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td></td>
<td>hate</td>
<td></td>
<td>this</td>
<td></td>
<td>movie</td>
</tr>
</tbody>
</table>

2nd Pass

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>It</td>
<td>is</td>
<td>so</td>
<td>bad</td>
</tr>
</tbody>
</table>

state only, no backprop
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

I → hate → this → movie

RNN → RNN → RNN → RNN

2nd Pass

It → is → so → bad

state only, no backprop

It → is → so → bad
Truncated BPTT

- Backprop over shorter segments, initialize with the state from the previous segment

**1st Pass**

```
I
hate
this
movie
```

**2nd Pass**

```
It
is
so
bad
```

State only, no backprop
Truncated BPTT

- Backprop over shorter segments, initialize with the state from the previous segment

**1st Pass**

```
I
hate
this
movie
```

**2nd Pass**

```
It
is
so
bad
```

state only, no backprop
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

```
I hate this movie
```

2nd Pass

```
It is so bad
```

state only, no backprop
Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment

1st Pass

2nd Pass

state only, no backprop
Pre-training/Transfer for RNNs
RNN Strengths/Weaknesses
RNN Strengths/Weaknesses

- RNNs, particularly deep RNNs/LSTMs, are quite powerful and flexible
RNN Strengths/Weaknesses

• RNNs, particularly deep RNNs/LSTMs, are quite powerful and flexible

• But they require a lot of data
RNN Strengths/Weaknesses

• RNNs, particularly deep RNNs/LSTMs, are quite powerful and flexible

• But they require a lot of data

• Also have trouble with weak error signals passed back from the end of the sentence
Pre-training/Transfer
Pre-training/Transfer

• Train for one task, solve another
Pre-training/Transfer

- Train for one task, solve another

- **Pre-training task**: Big data, easy to learn
Pre-training/Transfer

- Train for one task, solve another
- **Pre-training task:** Big data, easy to learn
- **Main task:** Small data, harder to learn
Example:
LM -> Sentence Classifier
(Luong et al. 2015)
Example:
LM -> Sentence Classifier
(Luong et al. 2015)

• Train a **language model first**: lots of data, easy-to-learn objective
Example:
LM -> Sentence Classifier
(Luong et al. 2015)

• Train a **language model first**: lots of data, easy-to-learn objective

• **Sentence classification**: little data, hard-to-learn objective
Example:
LM -> Sentence Classifier
(Luong et al. 2015)

• Train a **language model first**: lots of data, easy-to-learn objective

• **Sentence classification**: little data, hard-to-learn objective

• Results in much better classifications, competitive or better than CNN-based methods
Why Pre-training?
Why Pre-training?

• The model learns consistencies in the data (Karpathy et al. 2015)
Why Pre-training?

- The model learns consistencies in the data (Karpathy et al. 2015)
Why Pre-training?

- The model learns consistencies in the data (Karpathy et al. 2015)

- Model learns syntax (Shi et al. 2017) or semantics (Radford et al. 2017)
Questions?