CS 6501 Natural Language Processing

Recurrent Neural Networks

Yangfeng Ji

October 29, 2018

Department of Computer Science
University of Virginia

Overview

1. Recurrent Neural Networks
2. RNN Language Modeling
3. Challenge of Training RNNs
4. Variants of RNNs
5. Applications

Recurrent Neural Networks

RNNs

A simple RNN is defined as

$$\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \tag{1}$$

where $\boldsymbol{x}_t$ and $\boldsymbol{h}_t$ is the input and hidden state at time t, and $\boldsymbol{b}$ is the bias. $\mathbf{W}_h$ and $\mathbf{W}_i$ are the weight matrices for hidden states and inputs respectively.

Transition Function

For the simplest case, f is an element-wise sigmoid function as

$$\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) = \boldsymbol{f}(\mathbf{W}_h \boldsymbol{h}_{t-1} + \mathbf{W}_i \boldsymbol{x}_t + \boldsymbol{b}) \tag{2}$$

Unfolding RNNs

Recursive:

$$\boldsymbol{h}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \tag{3}$$

Unfolded:

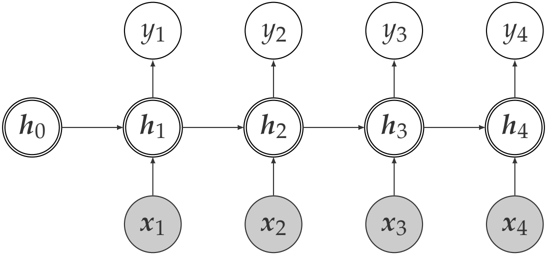
$$\begin{aligned}
\boldsymbol{h}_t &= f(\boldsymbol{x}_t, f(\boldsymbol{x}_{t-1}, \boldsymbol{h}_{t-2})) \\
&= f(\boldsymbol{x}_t, f(\boldsymbol{x}_{t-1}, f(\boldsymbol{x}_{t-2}, \boldsymbol{h}_{t-3}))) \\
&= \cdots \\
&= f(\boldsymbol{x}_t, f(\boldsymbol{x}_{t-1}, f(\boldsymbol{x}_{t-2}, \cdots f(\boldsymbol{x}_1, \boldsymbol{h}_0) \cdots)))
\end{aligned} \tag{4}$$

Base Condition

$$h_t = f(x_t, f(x_{t-1}, f(x_{t-2}, \cdots f(x_1, h_0) \cdots))) \qquad (5)$$

- h_0: zero vector or parameter
- x_1: input at time $t = 1$

Plot



Output

Loss at single time step t

$$L_t(y_t, \hat{y}_t) = \|y_t - \hat{y}_t\|_2^2 \tag{6}$$

where y_t and $\hat{y}_t = g(\boldsymbol{h}_t)$ are the ground truth and predicted output respectively.

The total loss is given as

$$\ell = \sum_{t=1}^{T} L_t \tag{7}$$

RNN Language Modeling

RNN Language Models

For a given sentence $\{x_1, \ldots, x_T\}$, the input at time t is word embedding $\boldsymbol{x}_t$. The probability distribution of next word X_{t+1}

$$P(X_{t+1} = x) = \frac{\exp(\boldsymbol{w}_{o,x}^\top \boldsymbol{h}_t)}{\sum_{x'} \exp(\boldsymbol{w}_{o,x'}^\top \boldsymbol{h}_t)} \tag{8}$$

where $\boldsymbol{w}_{o,x}$ is the output weight vector related to word x.

Special Cases

$$\{\texttt{start}, x_1, \ldots, x_T, \texttt{stop}\}$$

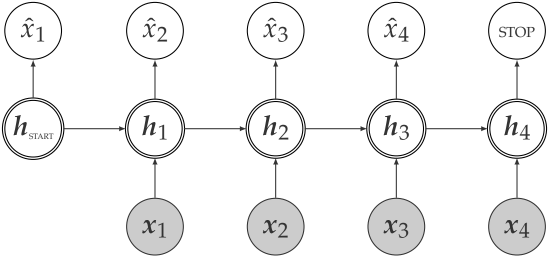
At time $t = 1$

$$P(X_1 = x) \propto \exp(\boldsymbol{w}_{o,x}^\top \boldsymbol{h}_{\texttt{start}}) \tag{9}$$

At time $t = T$

$$P(X_T = \texttt{stop}) \propto \exp(\boldsymbol{w}_{o,x}^\top \boldsymbol{h}_T) \tag{10}$$

Plot



$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_4$
STOP
h_{START}
h_1
h_2
h_3
h_4
x_1
x_2
x_3
x_4

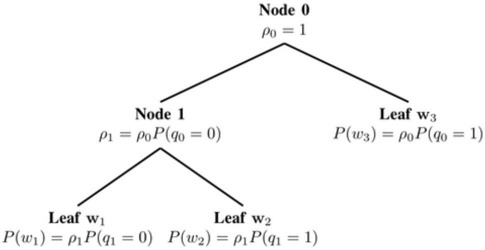
Normalization Term

$$P(X_{t+1} = x) = \frac{\exp(\boldsymbol{w}_{o,x}^{\top}\boldsymbol{h}_t)}{\sum_{x'} \exp(\boldsymbol{w}_{o,x'}^{\top}\boldsymbol{h}_t)} \quad (11)$$

Options:

- Negative sampling (x)
- Hierarchical softmax
- Class-factored softmax

Hierarchical Softmax



Class-factored Softmax: Definition

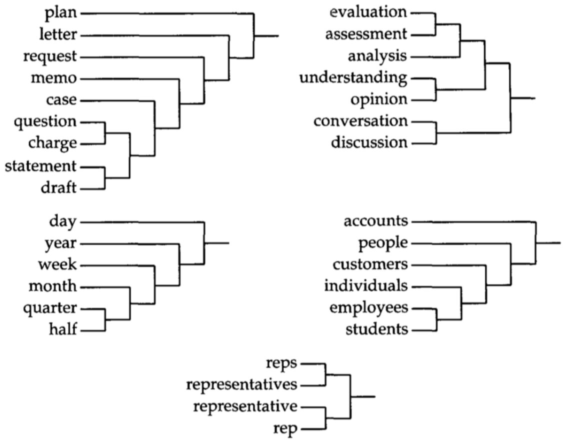
- Partition the vocab into K classes $\{\mathscr{C}_1, \ldots, \mathscr{C}_K\}$, such that $\mathscr{V} = \cup \mathscr{C}_k$ and $\mathscr{C}_k \cap \mathscr{C}_{k'} = \emptyset$ for any $k' \neq k$
- Define the probability distribution of word as

$$\begin{aligned} P(X_{t+1} = x; \boldsymbol{h}_t) = & P(X_{t+1} = x, C_{t+1} = c; \boldsymbol{h}_t) \\ = & P(X_{t+1} = x \mid C_{t+1} = c; \boldsymbol{h}_t) \\ & \cdot P(C_{t+1} = c \mid \boldsymbol{h}_t) \end{aligned} \tag{12}$$

[Baltescu and Blunsom, 2014]

Class-factored Softmax: Word clusters

Brown clusters



[Brown et al., 1992]

Plot

Computational Complexity

Model	Training/Decoding
Standard	$\mathcal{O}(\lvert\mathcal{V}\rvert \cdot D)$
Hierarchical	$\mathcal{O}(\log \lvert\mathcal{V}\rvert \cdot D)$
Class-factored	$\mathcal{O}(\sqrt{\lvert\mathcal{V}\rvert} \cdot D)$

Table: Computational complexities of different softmax functions.

Challenge of Training RNNs

Backpropagation Through Time

The algorithm used to train RNNs is called *Backpropagation Through Time* [Rumelhart et al., 1985, BPTT].

Consider the gradient of ℓ with respect to the network parameters $\boldsymbol{\theta} = \{\mathbf{W}_h, \mathbf{W}_i, \boldsymbol{b}\}$,

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial \boldsymbol{\theta}} \tag{13}$$

Gradients

For each time step t, we have

$$\frac{\partial L_t}{\partial \boldsymbol{\theta}} = \sum_{i=1}^{t} \frac{\partial L_t}{\partial \boldsymbol{h}_t} \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_i} \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{\theta}} \tag{14}$$

where both $\frac{\partial L_t}{\partial \boldsymbol{h}_t}$ and $\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{\theta}}$ are the intermediate gradients

Gradients (Cont.)

Computation of $\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_i}$ requires the chain rule in calculus,

$$\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_i} = \prod_{j=i+1}^{t} \frac{\partial \boldsymbol{h}_j}{\partial \boldsymbol{h}_{j-1}}. \tag{15}$$

which can be justified by the unfolded version of $\boldsymbol{h}_t$

$$
\begin{aligned}
\boldsymbol{h}_t &= f(\boldsymbol{x}_t, f(\boldsymbol{x}_{t-1}, \boldsymbol{h}_{t-2})) \\
&= f(\boldsymbol{x}_t, f(\boldsymbol{x}_{t-1}, f(\boldsymbol{x}_{t-2}, \boldsymbol{h}_{t-3}))) \\
&= \cdots \\
&= f(\boldsymbol{x}_t, f(\boldsymbol{x}_{t-1}, f(\boldsymbol{x}_{t-2}, \cdots f(\boldsymbol{x}_1, \boldsymbol{h}_0) \cdots))) \qquad (16)
\end{aligned}
$$

Challenges

$$\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_i} = \prod_{j=i+1}^{t} \frac{\partial \boldsymbol{h}_j}{\partial \boldsymbol{h}_{j-1}}. \tag{17}$$

- vanishing gradients
- exploding gradients

[Pascanu et al., 2013]

Exploding Gradients

Solution: **norm clipping** [Pascanu et al., 2013].

Consider the gradient $\boldsymbol{g} = \frac{\partial \ell}{\partial \boldsymbol{\theta}}$,

$$\hat{\boldsymbol{g}} \leftarrow \tau \cdot \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|} \tag{18}$$

when $\|\boldsymbol{g}\| > \tau$. Usually, $\tau = 3$ or 5 in practice.

Vanishing Gradients

Solution:

- initialize parameters carefully
- replace hidden state function $\sigma()$ with other options
  - LSTM [Hochreiter and Schmidhuber, 1997]
  - GRU [Cho et al., 2014]

Long Short-Term Memory

$$\boldsymbol{i}_t = \sigma(\mathbf{W}_{xi}\boldsymbol{x}_t + \mathbf{W}_{hi}\boldsymbol{h}_{t-1} + \mathbf{W}_{ci}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i) \quad (19)$$

$$\boldsymbol{f}_t = \sigma(\mathbf{W}_{xf}\boldsymbol{x}_t + \mathbf{W}_{hf}\boldsymbol{h}_{t-1} + \mathbf{W}_{cf}\boldsymbol{c}_{t-1} + \boldsymbol{b}_f) \quad (20)$$

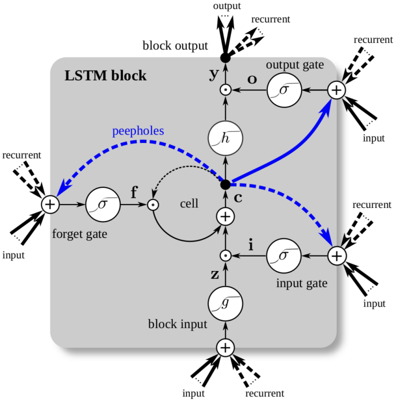
$$\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tanh(\mathbf{W}_{xc}\boldsymbol{x}_t + \mathbf{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \quad (21)$$

$$\boldsymbol{o}_t = \sigma(\mathbf{W}_{xo}\boldsymbol{x}_t + \mathbf{W}_{ho}\boldsymbol{h}_{t-1} + \mathbf{W}_{co}\boldsymbol{c}_t + \boldsymbol{b}_o) \quad (22)$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \circ \tanh(\boldsymbol{c}_t) \quad (23)$$

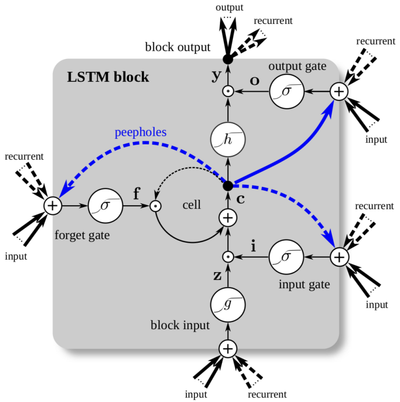
$\circ$ is the element-wise multiplication [Graves, 2013]

LSTM



output
recurrent
block output
recurrent
LSTM block
y
o
output gate
σ
input
peepholes
h
recurrent
f
cell
c
σ
forget gate
recurrent
input
i
σ
z
input gate
input
g
block input
input
recurrent

LSTM



- Forget gate f_t — discounting on the memory cell
- Peephole connections (connections in blue color) [Gers and Schmidhuber, 2000]

A Simple LSTM

A LSTM without forget gate and peephole connections

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\mathbf{W}_{xi}\boldsymbol{x}_t + \mathbf{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) && (24)\\
\boldsymbol{o}_t &= \sigma(\mathbf{W}_{xo}\boldsymbol{x}_t + \mathbf{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) && (25)\\
\boldsymbol{c}_t &= \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tanh(\mathbf{W}_{xc}\boldsymbol{x}_t + \mathbf{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) && (26)\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \circ \tanh(\boldsymbol{c}_t) && (27)
\end{aligned}$$

[Greff et al., 2017]

Gated Recurrent Units

A gated recurrent unit (GRU) was proposed in [Cho et al., 2014].

$$
\begin{aligned}
\boldsymbol{r}_t &= \sigma(\mathbf{W}_{rx}\boldsymbol{x}_t + \mathbf{W}_{rh}\boldsymbol{h}_{t-1}) && (28)\\
\tilde{\boldsymbol{h}}_t &= \tanh(\mathbf{W}_{hx}\boldsymbol{x}_t + \mathbf{W}_{hr}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})) && (29)\\
\boldsymbol{z}_t &= \sigma(\mathbf{W}_{zx}\boldsymbol{x}_t + \mathbf{W}_{zh}\boldsymbol{h}_{t-1}) && (30)\\
\boldsymbol{h}_t &= (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t && (31)\\
& && (32)
\end{aligned}
$$

Empirical results show GRU units are *comparable* to LSTM units [Chung et al., 2014].

Variants of RNNs

Overview

- Bi-directional RNNs
- Stacked (*or* Multi-layer) LSTM
- Memory networks [Weston et al., 2014]
- Recurrent neural network grammars [Dyer et al., 2016]

Bi-directional RNNs

To construct a bi-directional RNN, we need another uni-directional RNN running from the end of the sequence to the beginning, as

$$\boldsymbol{u}_t = f(\boldsymbol{x}_t, \boldsymbol{u}_{t+1}). \tag{33}$$

where $\boldsymbol{u}_t$ is the hidden state at time t in this new model.

[Schuster and Paliwal, 1997]

Stacked LSTM

Use the hidden state $\boldsymbol{h}_t^{(k)}$ from the current layer as input $\boldsymbol{x}_t^{(k+1)}$ to the next layer [Sutskever et al., 2014],

$$\boldsymbol{x}_t^{(k+1)} = \boldsymbol{h}_t^{(k)}. \tag{34}$$

[Sutskever et al., 2014]

Applications

Applications

- Language modeling
- POS tagging
- Named entity recognition
- Code switch
- ...

NER

Example

`Atlantis touched down at Kennedy Space Center`

NER

Example

[Atlantis]$_{\text{MSIC}}$ touched down at [Kennedy Space Center]$_{\text{LOC}}$

NER

Example

`[Atlantis]`$_{\text{MSIC}}$ `touched down at [Kennedy Space Center]`$_{\text{LOC}}$

Tag set

- B: beginning
- I: inside
- O: outside

Category

- Person
- Location
- Organization
- Msic

NER

Example

`[Atlantis]`$_{\text{MSIC}}$ `touched down at [Kennedy Space Center]`$_{\text{LOC}}$

Tag set

- B: beginning
- I: inside
- O: outside

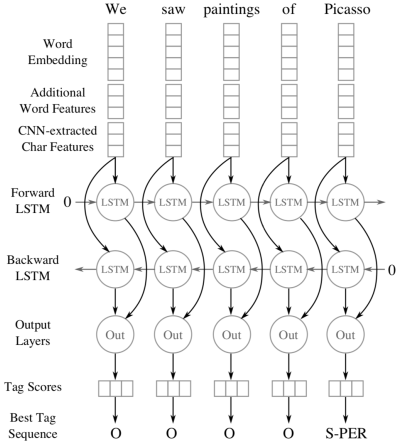
Category

- Person
- Location
- Organization
- Msic

Atlantis	touched	down	at	Kennedy	Space	Center	.
B_{MSIC}	O	O	O	B_{LOC}	I_{LOC}	I_{LOC}	O

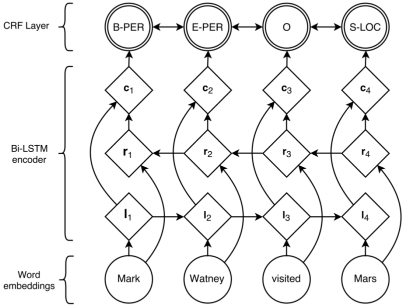
NER (Cont.)

As classification



NER (Cont.)

As sequential modeling (CRFs)



Summary

1. Recurrent Neural Networks
2. RNN Language Modeling
3. Challenge of Training RNNs
4. Variants of RNNs
5. Applications

Reference

Baltescu, P. and Blunsom, P. (2014).
Pragmatic neural language modelling in machine translation.
arXiv preprint arXiv:1412.7119.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992).
Class-based n-gram models of natural language.
Computational linguistics, 18(4):467–479.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014).
On the properties of neural machine translation: Encoder-decoder approaches.
arXiv preprint arXiv:1409.1259.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014).
Empirical evaluation of gated recurrent neural networks on sequence modeling.
arXiv preprint arXiv:1412.3555.

Dyer, C., Kuncoro, A., Ballesteros, M., and Smith, N. A. (2016).
Recurrent neural network grammars.
arXiv preprint arXiv:1602.07776.

Gers, F. A. and Schmidhuber, J. (2000).
Recurrent nets that time and count.
In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194. IEEE.

Graves, A. (2013).
Generating sequences with recurrent neural networks.
arXiv preprint arXiv:1308.0850.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2017).
Lstm: A search space odyssey.