IN5550: Neural Methods in Natural Language Processing
Sub-lecture 6.3
How to use RNNs

Andrey Kutuzov

University of Oslo

24 February 2022



Contents



1. How to use RNNs?

A Note on Architecture Design

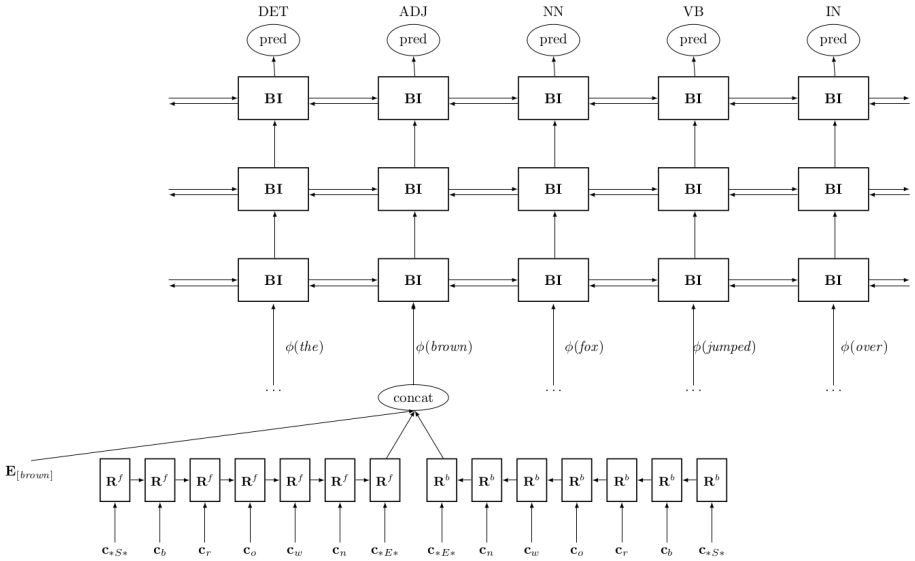


While it is not theoretically clear what is the additional power gained by the deeper architectures, it was observed empirically that deep RNNs work better than shallower ones on some tasks. [...]

Many works report results using layered RNN architectures, but do not explicitly compare to one-layer RNNs. In the experiments of my research group, using two or more layers indeed often improves over using a single one.

[Goldberg, 2017]

RNNs as Feature Extractors



Common Applications of RNNs (in NLP)



- Acceptors, e.g. sentence-level sentiment classification:

$$
\begin{aligned}
P(c = k | w_{1:n}) &= \hat{\boldsymbol{y}}_{[k]} \\
\hat{\boldsymbol{y}} &= \text{softmax}(\text{MLP}([\text{RNN}^{\text{f}}(\boldsymbol{x_{1:n}}); \text{RNN}^{\text{b}}(\boldsymbol{x_{n:1}})])) \\
\boldsymbol{x_{1:n}} &= \boldsymbol{E_{[w_1]}}, \ldots, \boldsymbol{E_{[w_n]}}
\end{aligned}
$$

Common Applications of RNNs (in NLP)



- Acceptors, e.g. sentence-level sentiment classification:

$$
\begin{aligned}
P(c = k|w_{1:n}) &= \hat{\boldsymbol{y}}_{[k]} \\
\hat{\boldsymbol{y}} &= \text{softmax}(\text{MLP}([\text{RNN}^{\text{f}}(\boldsymbol{x_{1:n}})_{[n]}; \text{RNN}^{\text{b}}(\boldsymbol{x_{n:1}})_{[1]}])) \\
\boldsymbol{x_{1:n}} &= \boldsymbol{E}_{[w_1]}, \ldots, \boldsymbol{E}_{[w_n]}
\end{aligned}
$$

Common Applications of RNNs (in NLP)



- Acceptors, e.g. sentence-level sentiment classification:

$$
\begin{aligned}
P(c = k|w_{1:n}) &= \hat{\boldsymbol{y}}_{[k]} \\
\hat{\boldsymbol{y}} &= \text{softmax}(\text{MLP}([\text{RNN}^{\text{f}}(\boldsymbol{x_{1:n}})_{[n]}; \text{RNN}^{\text{b}}(\boldsymbol{x_{n:1}})_{[1]}])) \\
\boldsymbol{x_{1:n}} &= \boldsymbol{E_{[w_1]}}, \ldots, \boldsymbol{E_{[w_n]}}
\end{aligned}
$$

- transducers, e.g. part-of-speech tagging:

$$
\begin{aligned}
P(c_i = k|w_{1:n}) &= \text{softmax}(\text{MLP}([\text{RNN}^{\text{f}}(\boldsymbol{x_{1:n}})_{[i]}; \text{RNN}^{\text{b}}(\boldsymbol{x_{n:1}})_{[i]}]))_{[k]} \\
\boldsymbol{x_i} &= \boldsymbol{E_{[w_i]}}
\end{aligned}
$$

Common Applications of RNNs (in NLP)



- Acceptors, e.g. sentence-level sentiment classification:

$$
\begin{aligned}
P(c = k|w_{1:n}) &= \hat{\boldsymbol{y}}_{[k]} \\
\hat{\boldsymbol{y}} &= \text{softmax}(\text{MLP}([\text{RNN}^{\text{f}}(\boldsymbol{x_{1:n}})_{[n]}; \text{RNN}^{\text{b}}(\boldsymbol{x_{n:1}})_{[1]}])) \\
\boldsymbol{x_{1:n}} &= \boldsymbol{E_{[w_1]}}, \ldots, \boldsymbol{E_{[w_n]}}
\end{aligned}
$$

- transducers, e.g. part-of-speech tagging:

$$
\begin{aligned}
P(c_i = k|w_{1:n}) &= \text{softmax}(\text{MLP}([\text{RNN}^{\text{f}}(\boldsymbol{x_{1:n}})_{[i]}; \text{RNN}^{\text{b}}(\boldsymbol{x_{n:1}})_{[i]}]))_{[k]} \\
\boldsymbol{x_i} &= [\boldsymbol{E_{[w_i]}}; \text{RNN}^{\text{f}}_{\text{c}}(\boldsymbol{c_{1:l_i}}); \text{RNN}^{\text{b}}_{\text{c}}(\boldsymbol{c_{l_i:1}})]
\end{aligned}
$$

- character-level RNNs robust to unknown words; may capture affixation

Common Applications of RNNs (in NLP)

- Acceptors, e.g. sentence-level sentiment classification:

$$\begin{aligned} P(c = k|w_{1:n}) &= \hat{\boldsymbol{y}}_{[k]} \\ \hat{\boldsymbol{y}} &= \text{softmax}(\text{MLP}([\text{RNN}^{\text{f}}(\boldsymbol{x_{1:n}})_{[n]}; \text{RNN}^{\text{b}}(\boldsymbol{x_{n:1}})_{[1]}])) \\ \boldsymbol{x_{1:n}} &= \boldsymbol{E_{[w_1]}}, \ldots, \boldsymbol{E_{[w_n]}} \end{aligned}$$

- transducers, e.g. part-of-speech tagging:

$$\begin{aligned} P(c_i = k|w_{1:n}) &= \text{softmax}(\text{MLP}([\text{RNN}^{\text{f}}(\boldsymbol{x_{1:n}})_{[i]}; \text{RNN}^{\text{b}}(\boldsymbol{x_{n:1}})_{[i]}]))_{[k]} \\ \boldsymbol{x_i} &= [\boldsymbol{E_{[w_i]}}; \text{RNN}^{\text{f}}_{\text{c}}(\boldsymbol{c_{1:l_i}}); \text{RNN}^{\text{b}}_{\text{c}}(\boldsymbol{c_{l_i:1}})] \end{aligned}$$

- character-level RNNs robust to unknown words; may capture affixation
- encoder–decoder (sequence-to-sequence) models for generation (later in the course)

Fun Outlook: Conditioned Generation with RNNs



man in black shirt is playing guitar.



construction worker in orange safety vest is working on road.



two young girls are playing with lego toy.



boy is doing backflip on wakeboard.



pizza with tomatoes and vegetables on it.



woman in bikini is jumping over a hurdle.



man in white shirt is pushing his cart down the street.



a group of people sitting on a couch with their laptops.

[Karpathy, 2016]

RNNs In a Nutshell



- State vectors s_i reflect the complete history up to time point i;
- RNNs are sensitive to basic natural language structures: sequences;

RNNs In a Nutshell



- State vectors $\boldsymbol{s_i}$ reflect the complete history up to time point i;
- RNNs are sensitive to basic natural language structures: sequences;
- applicable to arbitrary length inputs (in principle);
- few parameters: matrices $\boldsymbol{W^s}$ and $\boldsymbol{W^x}$ shared across all time points;

RNNs In a Nutshell



- State vectors $\boldsymbol{s_i}$ reflect the complete history up to time point i;
- RNNs are sensitive to basic natural language structures: sequences;
- applicable to arbitrary length inputs (in principle);
- few parameters: matrices $\boldsymbol{W^s}$ and $\boldsymbol{W^x}$ shared across all time points;
- analogous to (potentially) deep nesting: repeated multiplications;
- near-crippling practical limitation: 'exploding' or 'vanishing' gradients;

RNNs In a Nutshell



- State vectors $\boldsymbol{s_i}$ reflect the complete history up to time point i;
- RNNs are sensitive to basic natural language structures: sequences;
- applicable to arbitrary length inputs (in principle);
- few parameters: matrices $\boldsymbol{W^s}$ and $\boldsymbol{W^x}$ shared across all time points;
- analogous to (potentially) deep nesting: repeated multiplications;
- near-crippling practical limitation: 'exploding' or 'vanishing' gradients;
- Solution [$\rightarrow$] gated RNNs: LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Cho et al., 2014].

Learn about the gated architectures in the next sub-lecture 6.4!

References I

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014).
Learning phrase representations using RNN encoder–decoder for statistical machine translation.
In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Goldberg, Y. (2017).
Neural network methods for natural language processing.
Synthesis Lectures on Human Language Technologies, 10(1):1–309.

Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural computation, 9(8):1735–1780.

References II



Karpathy, A. (2016).
Connecting images and natural language.
PhD thesis, Stanford University.