Recurrent Neural Networks (RNN)

CS5242

Lee Hwee Kuan & **Wang Wei**

Teaching assistant:

Connie Kou Khor Li, Ji Xin, Ouyang Kun

cs5242@comp.nus.edu.sg

Mid-term survey

- Topics
  - Basics (output size, computation cost) VS advanced stuff (architectures)
  - More software/implementation practices VS more mathematics
  - More Introduction/background/motivation
- Slides
  - More explanation/notes
  - Fonts, figures
- Workload
  - Assignment and projects
- Voice & Pronunciation

Announcement

- Quiz 20%
    - Oct. 26, 18:30-19:30, open book
- NO assignment 4
    - Assignment 1: 10%
    - Assignment 2: 15%, Due date: 22 Oct. 11:59PM (extended)
    - Assignment 3: 15%
- Saturday session
    - 15:30-17:30
    - Lecture room (I3 auditorium room)
    - IVLE survey

Roadmap

Motivation of RNN

Vanilla RNN

GRU , LSTM and Bi-directional RNN

Intended learning outcomes

01

Understand the properties of RNN compared with feed-forward NN

02

Implement the BP of vanilla RNN

03

Know the problem of vanilla RNN and the properties of LSTM/GRU

04

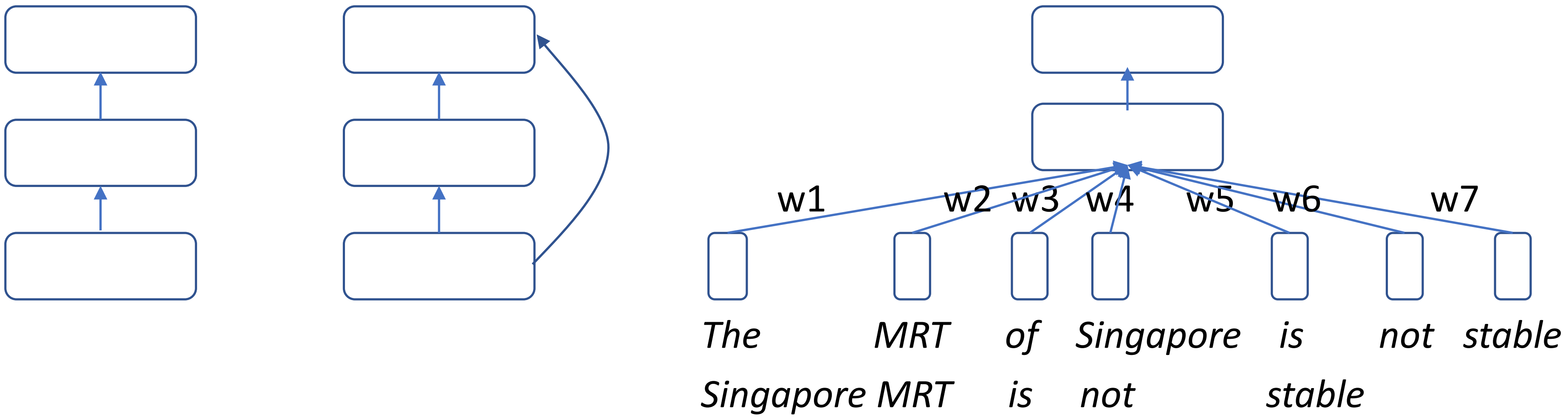
Train RNN (vanilla/LSTM/GRU) for language modelling

Motivation

From feed-forward NN to RNN

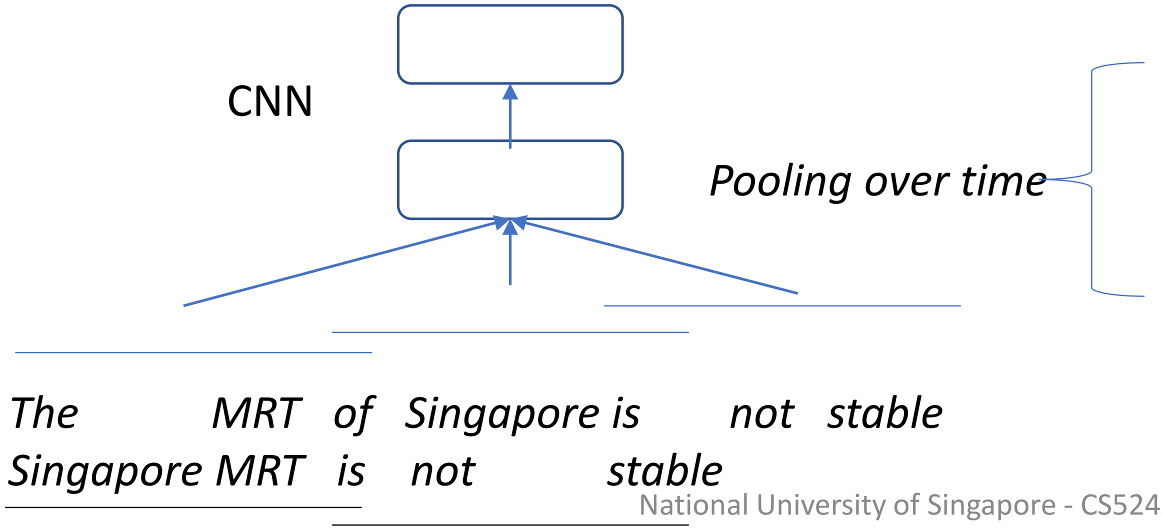
- Feed-forward NN (acyclic)
  - Accept single/static input sample, e.g. image
  - Not good at processing a sequence of data
    - E.g. a sentence of words for sentiment analysis; **how to do it using MLP?**

Feed-forward NN


w1
w2
w3
w4
w5
w6
w7
The MRT of Singapore is not stable
Singapore MRT is not stable

From feed-forward NN to RNN

- Feed-forward NN (acyclic)
  - Accept single/static input sample, e.g. image
  - Not good at processing a sequence of data
    - CNN’s receptive fields share the parameters (i.e. kernels)
    - Kernel size typically > 1 -> words within the receptive field are processed differently
      - “MRT of Singapore” != “Singapore MRT is”



CNN

Pooling over time

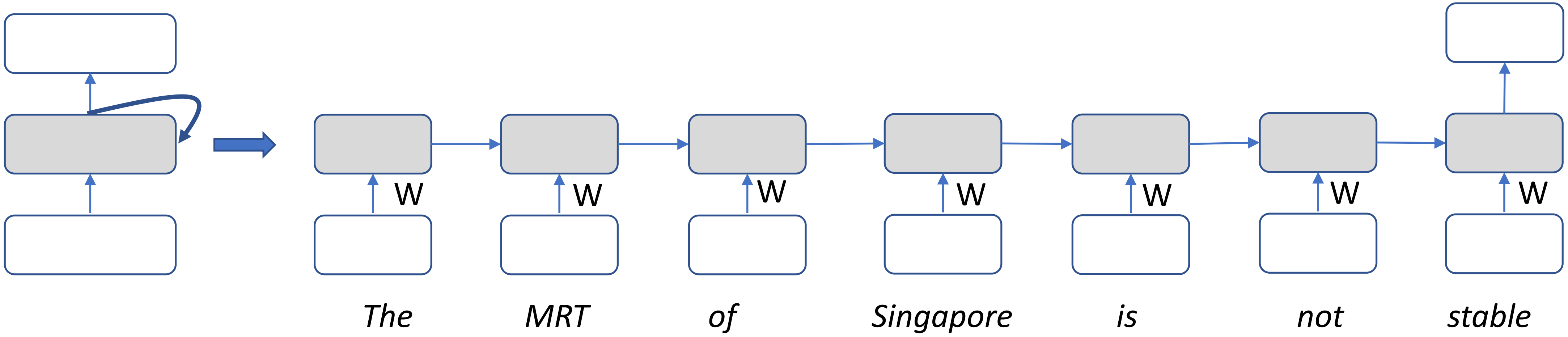
Output of the conv layer:
*f * 3 and f * 2*
Output of the pooling layer:
f and f

The MRT of Singapore is not stable

Singapore MRT is not stable

From feed-forward NN to RNN

- Feed-forward NN (acyclic)
- RNN
  - Accept dynamic/sequence data (length not fixed)
    - Words are processed in the same way recurrently
      - # unfold units = input sequence length
      - weights are tied



RNN

- Applications
  - Language modelling
    - Predict the next word given the previous words in a sentence
  - Machine translation
    - Translate the input English sentence to French
  - Speech recognition
    - Recognize and translate spoken language into text
  - Question answering
    - Generate text answers for (simple) questions
  - Etc.

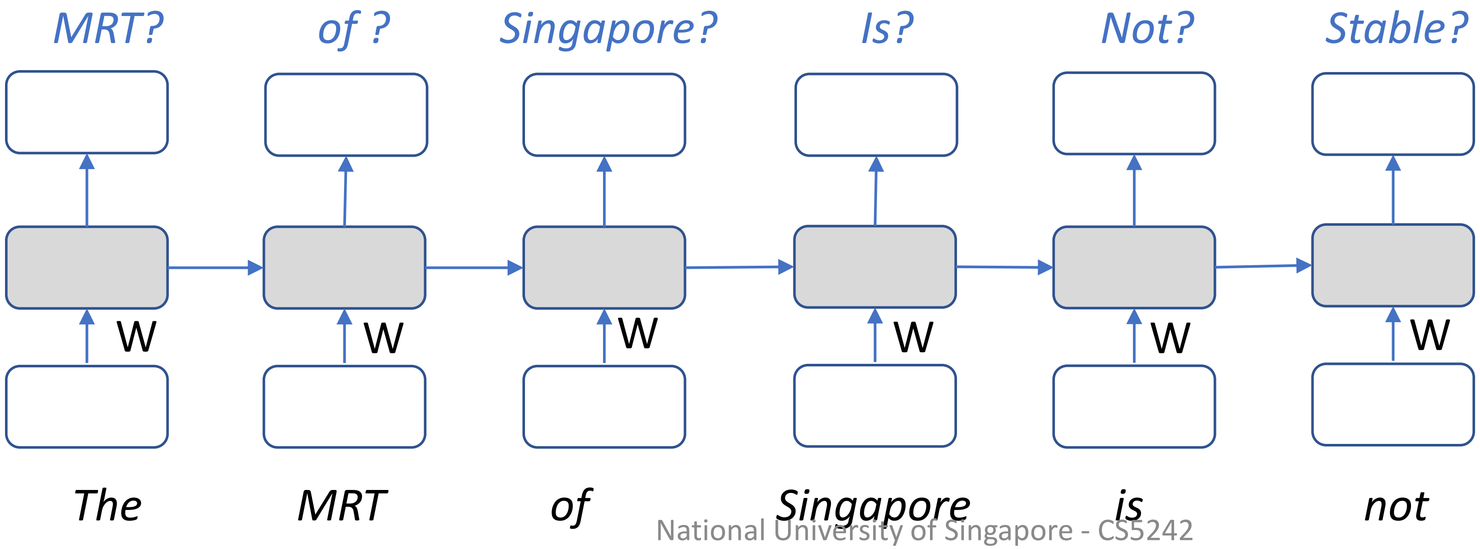
Vanilla RNN

Language modelling example

- Given a corpus of text (e.g. sentences), model the probability of a sentence (i.e. a sequence of words)
  - $P(x_1, x_2, \ldots, x_n)$
  - Useful for many applications involving text/sentence generation**?**
    - Machine translation, speech recognition, question answering, etc.
      - P("Singapore MRT is not **stable**") > P("Singapore MRT is not **NUS**")
      - P("Singapore **MRT is** not stable") > P("Singapore **is MRT** not stable")
  - Refer to [5] for traditional approaches for this problem

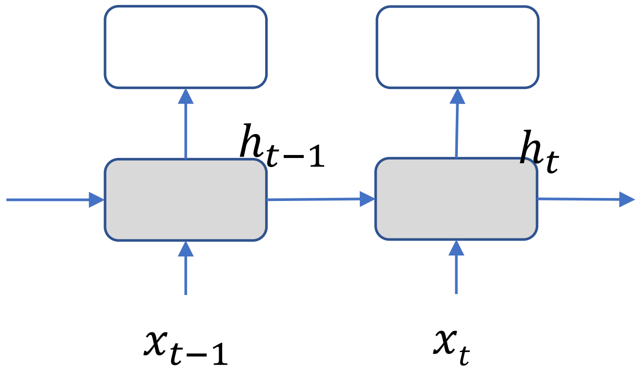
Language modelling example

- $P(x_1, x_2, \ldots, x_n) = \prod_t P(x_t | x_{t-1}, x_{t-2}, \ldots, x_1)$
  - $logP(x_1, x_2, \ldots, x_n) = \sum_t logP(x_t | x_{t-1}, x_{t-2}, \ldots, x_1)$ → **?**
  - Maximize log-likelihood == minimize cross-entropy
    - Train classifiers to predict the next word given the preceding words
    - P(MRT|The), P(of|The MRT), P(Singapore|The MRT of), …



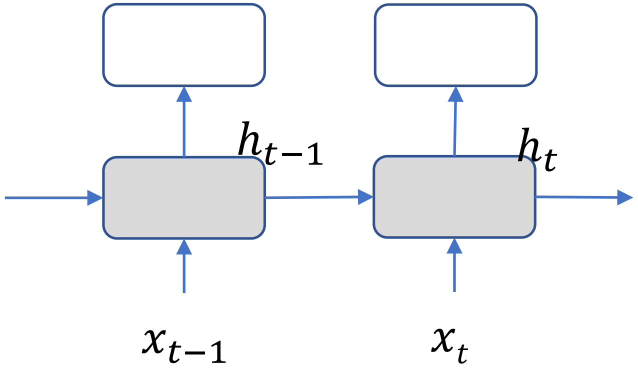
Hidden Layer

- Denote the vocabulary of all words as V
  - {MRT:0, Singapore:1, Stable:2, …}.
- Represent the word at position t as a column vector $x_t \in R^d$,
  - E.g., word vectors [2,3]; d is word vector length, e.g. 32, 64, 128.
  - Retrieve the word vector from the downloaded file using the word as key



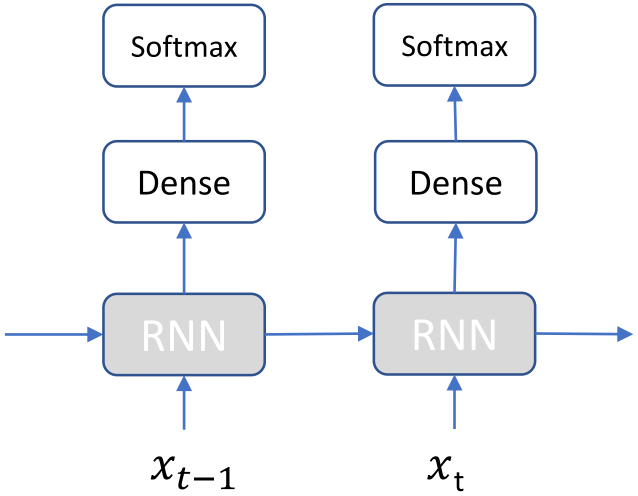
Hidden Layer

- Denote the vocabulary of all words as V
  - {MRT:0, Singapore:1, Stable:2, …}.
- Represent the word at position t as a column vector $x_t \in R^d$,
  - E.g., word vectors [2,3]; d is word vector length, e.g. 32, 64, 128.
- Denote the hidden layer at position t as $h_t \in R^k$
  - k is defined by users
  - $h_t = f(h_{t-1}, x_t|\theta)$**?**
    - $a_t = Ux_t + Wh_{t-1} + b,$
      - $\theta = \{U \in R^{k\times d}, W \in R^{k\times k}, b \in R^k\}$
    - $h_t = \tanh(a_t), h_t \in R^k$



Output

- $o_t = Vh_t + c,$
  - V $\in R^{|V|\times k}, c \in R^{|V|}, ot \in R^{|V|}$
- $y_t = softmax(o_t)$
  - If |V| is very large, the prediction layer needs special optimization [6]
  - $y_t \in R^{|V|}$, a probability vector

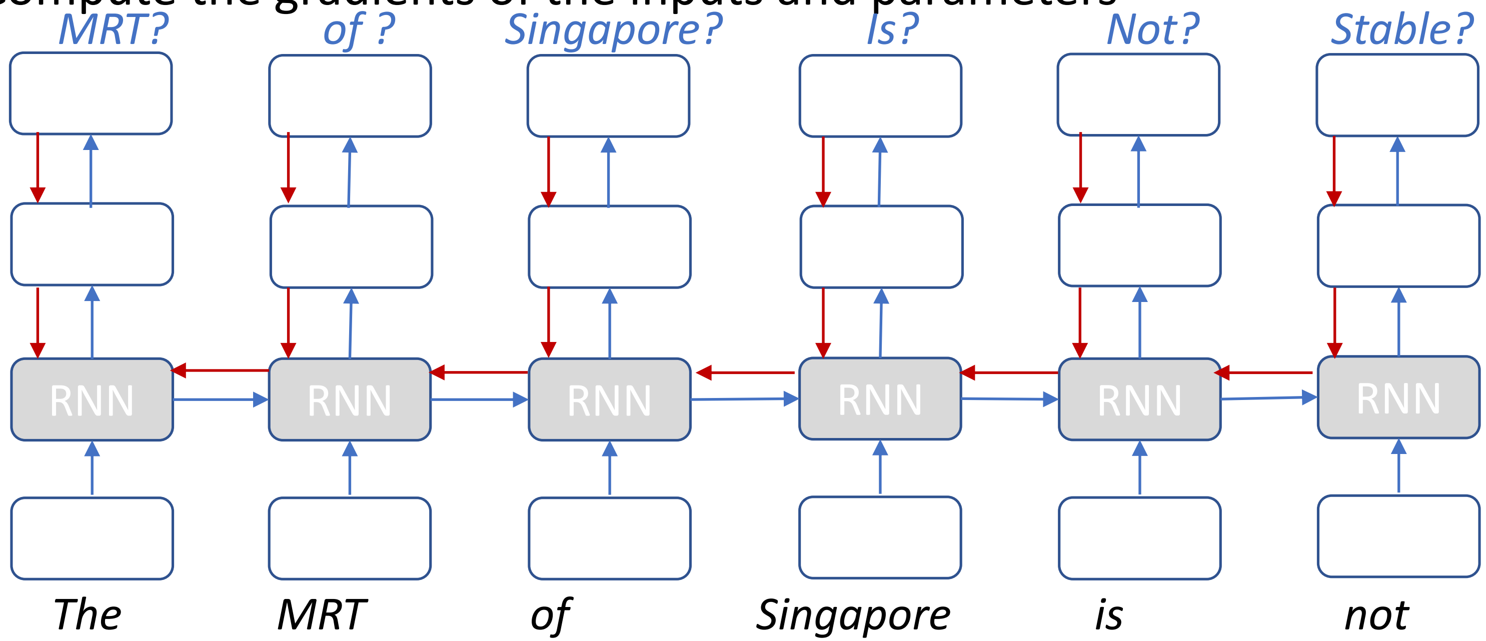


Training

- Given a corpus of text data (e.g. sentences), train the parameters of the RNN.
- Objective:
  - Maximize $logP(x_1, x_2, \dots, x_n) = \sum_t logP(\, x_t | x_{t-1}, x_{t-2}, \dots, x_1)$
  - → minimize the cross-entropy loss at each position t, denoted as L_t
  - $L = \sum_t L_t$
- SGD
  - For each data sample (e.g. a sentence)
    - Compute the gradients of each parameter
    - Update the parameters by $\theta = \theta - \alpha \times \frac{\partial L}{\partial \theta}$

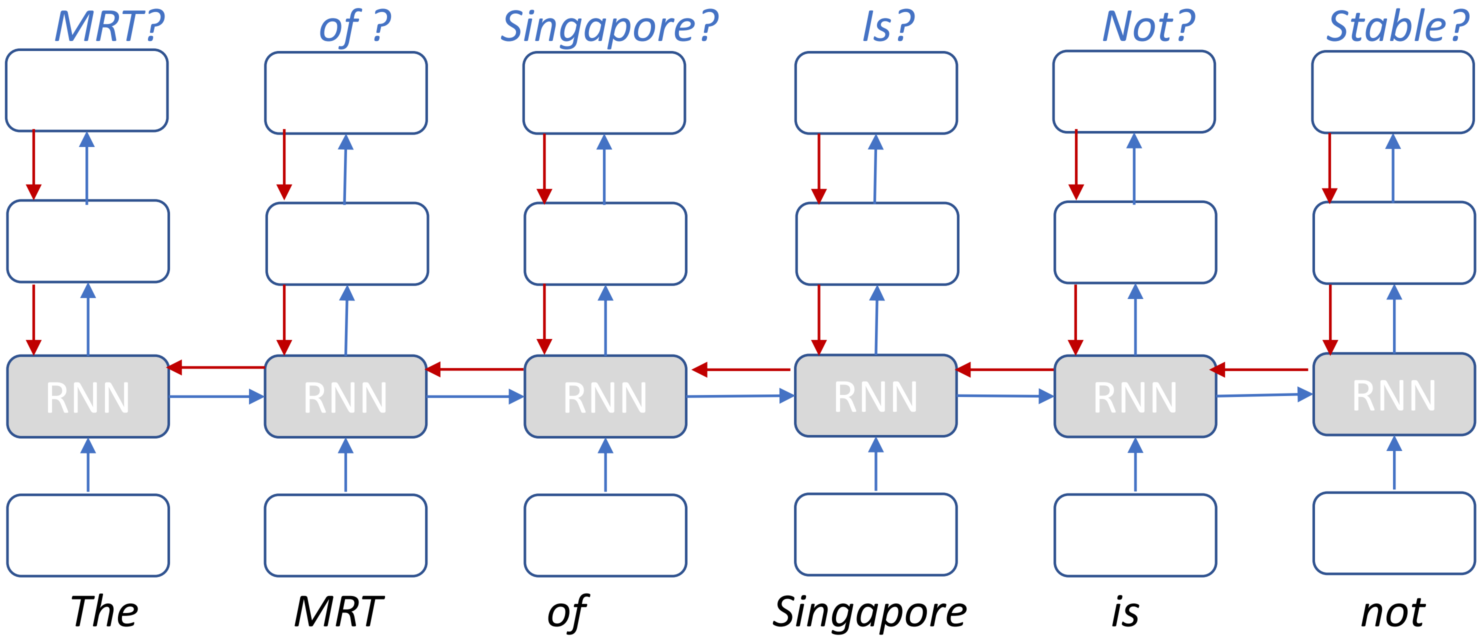
Back-propagation through time (BPTT)

- Back-propagation for each position as normal
  - From the cross-entropy to the RNN layer
  - For each RNN layer
    - aggregate the gradients from the top layer and the right layer
    - Compute the gradients of the inputs and parameters



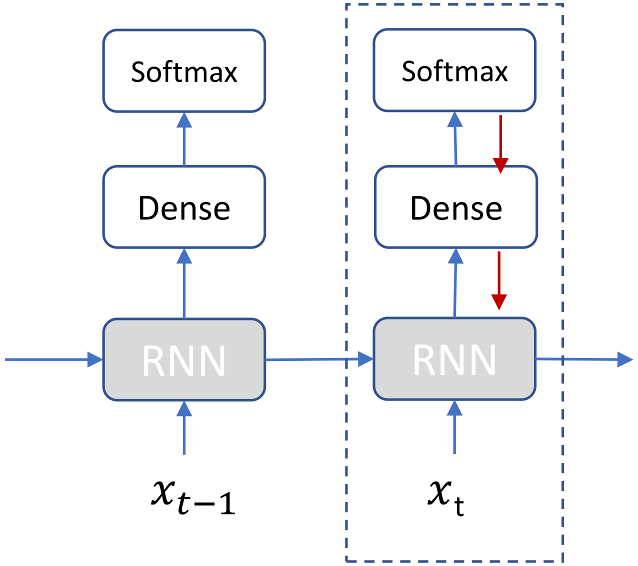
Back-propagation through time (BPTT)

- Back-propagation for each position as normal
  - From the cross-entropy to the RNN layer
  - For each RNN layer
  - For each parameter, aggregate the gradient across all positions



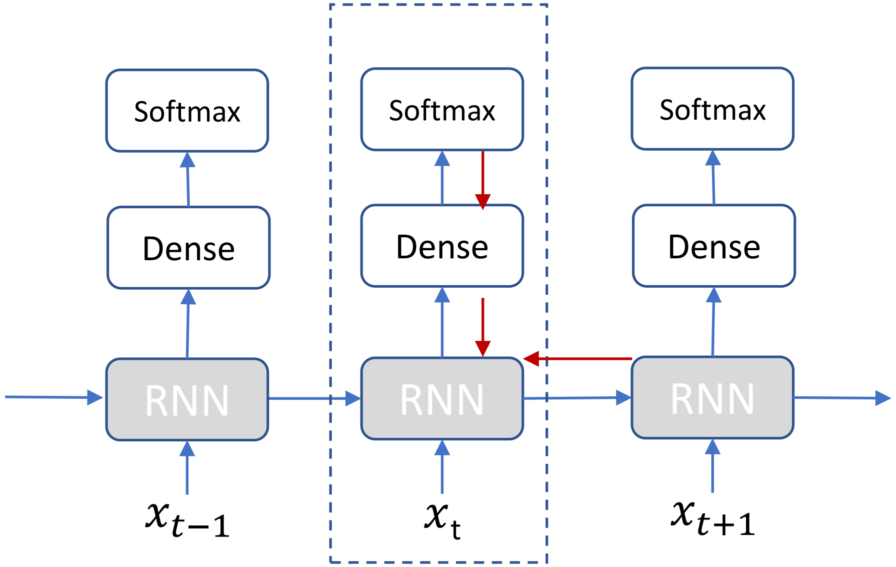
Back-propagation through time (BPTT)

- Forward
  - $a_t = Ux_t + Wh_{t-1} + b,$
  - $h_t = \tanh(a_t)$
  - $o_t = Vh_t + c, y_t = softmax(o_t)$
- Softmax+cross-entropy
  - $\frac{\partial L_t}{\partial o_t} = y_t - l_t, l_t \in \{0,1\}^{|V|}$, the ground truth vector
- Dense
  - $\frac{\partial L_t}{\partial h_t} = V^T \frac{\partial L_t}{\partial o_t}$
  - $\frac{\partial L_t}{\partial V_t} = \left(\frac{\partial L_t}{o_t}\right)(h_t)^T, \frac{\partial L_t}{\partial c_t} = \frac{\partial L_t}{\partial o_t},$ gradients of V and c from t-th position



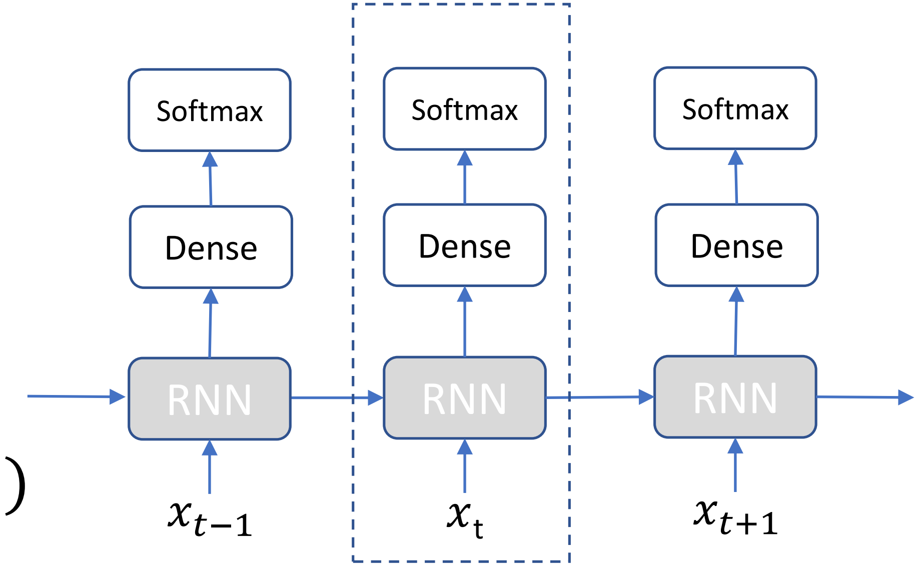
Back-propagation through time (BPTT)

- Forward
  - $a_t = Ux_t + Wh_{t-1} + b,$
  - $h_t = \tanh(a_t)$
  - $o_t = Vh_t + c$
- RNN layer
  - $\frac{\partial L}{\partial h_t} = \frac{\partial L_t}{\partial h_t} + \boxed{\frac{\partial L_{t+1}}{\partial h_t} + \cdots + \frac{\partial L_n}{\partial h_t}} = \frac{\partial L_t}{\partial h_t} + \frac{\partial L_{t+}}{\partial h_t}$
  - $\frac{\partial L}{\partial a_t} = \frac{\partial L}{\partial h_t} \times (1 - h_t^2)$
  - $\frac{\partial L}{\partial U_t} = \frac{\partial L}{\partial a_t} x_t^T, \frac{\partial L}{\partial W_t} = \frac{\partial L}{\partial a_t} h_{t-1}^T, \frac{\partial L}{\partial b_t} = \frac{\partial L}{\partial a_t}$, gradients of U, W, b from position t
  - $\frac{\partial L_{(t-1)+}}{\partial h_{t-1}} = W^T \frac{\partial L}{\partial a_t}$
- $\frac{\partial L}{\partial \theta} = \sum_t \frac{\partial L}{\partial \theta_t}$



Back-propagation through time (BPTT)

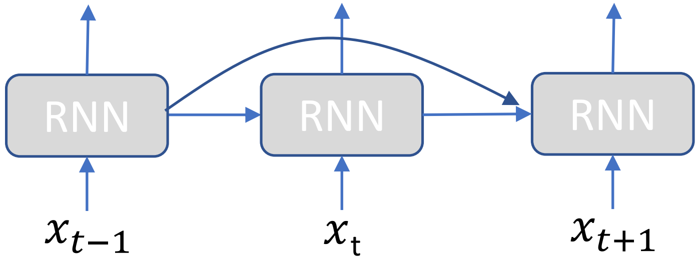
- Gradient vanishing/exploding
  - $\frac{\partial L}{\partial h_t} = \frac{\partial L_t}{\partial h_t} + \frac{\partial L_{t+1}}{\partial h_t} + \cdots + \frac{\partial L_n}{\partial h_t} = \frac{\partial L_t}{\partial h_t} + \frac{\partial L_{t+}}{\partial h_t}$
  - $\frac{\partial L}{\partial a_t} = \frac{\partial L}{\partial h_t} \times (1 - h_t^2)$
  - $\rightarrow \frac{\partial L_{(t-1)+}}{\partial h_{t-1}} = W^T \frac{\partial L}{\partial a_t} = W^T \frac{\partial L}{\partial h_t} \times (1 - h_t^2)$
  - $= W^T \left(\frac{\partial L_t}{\partial h_t} + \frac{\partial L_{t+}}{\partial h_t}\right) \times (1 - h_t^2)$
  - $\rightarrow \frac{\partial L_{(t-1)+}}{\partial h_{t-1}} \leftarrow W^T \frac{\partial L_{t+}}{\partial h_t} \ldots \leftarrow (W^T)^k \frac{\partial L_{(t+k)+}}{\partial h_t}$



Gradients from right most positions vanish when back-propagated to the left-most positions

Back-propagation through time (BPTT)

- $\frac{\partial L_{(t-1)+}}{\partial h_{t-1}} \leftarrow W^T \frac{\partial L_{t+}}{\partial h_t} \dots \leftarrow (W^T)^k \frac{\partial L_{(t+k)+}}{\partial h_t}$
  - If |W| is small, gradient vanishing
    - The losses after position t+k have little influence for the RNN layer at t-1 if k is large
    - Cannot capture long-term relationship
      - “The **red line** went down last night, which is why there are many tweets about __ “(red line).
  - If |W| is large, gradient exploding
- Solutions
  - Gradient vanishing?
    - Careful initialization
      - Identity matrix with ReLU as the activation function[7]
    - Skip-connections
    - leaky units-> LSTM and GRU
      - $h_t = \gamma h_{t-1} + (1-\gamma)\tanh(Uxt + Wh_{t-1} + b)$
  - Gradient exploding?
    - Gradient clipping

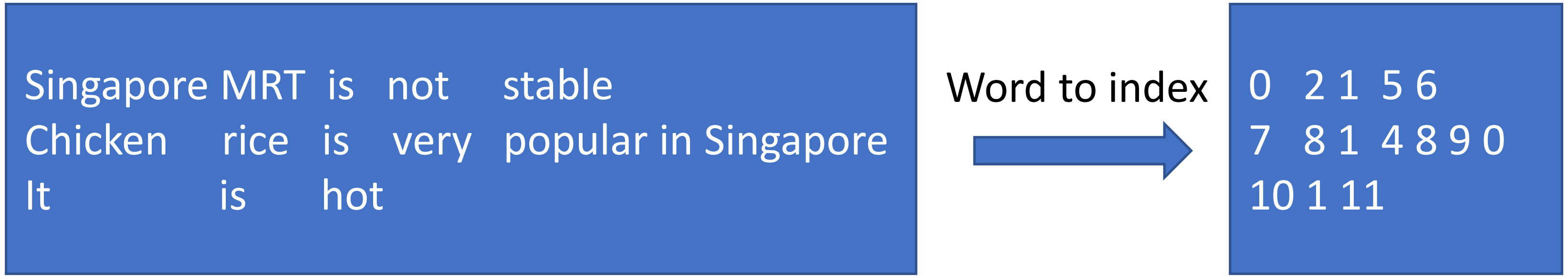


Gradient Clipping

- $W = W - \alpha \times \frac{\partial L}{\partial W}$
- Hard clipping
  - For each value of $\frac{\partial L}{\partial W}$, if it is larger than a threshold μ, set it to be μ
- Normalization (L2)
  - g= $\frac{\partial L}{\partial W}$
  - If |g|> μ, $\mathrm{g} = \frac{\mu}{|g|} g$

Mini-batch SGD

- SGD uses a single sample per iteration
- Mini-batch SGD uses multiple samples per iteration
  - To accelerate the processing by matrix (batch) operations
  - Different sentences have different lengths, e.g.



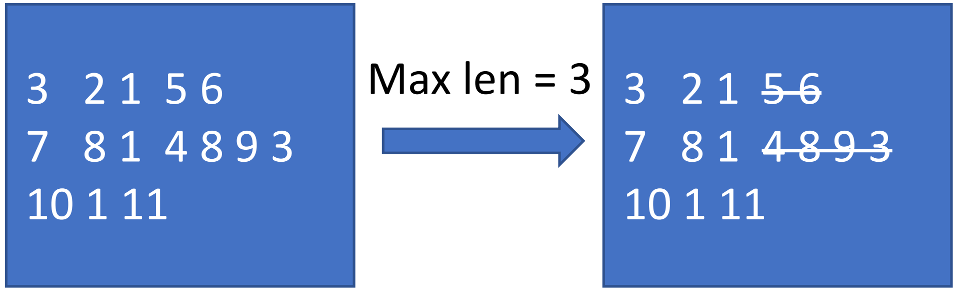
Singapore MRT is not stable
Chicken rice is very popular in Singapore
It is hot

Word to index

0 2 1 5 6
7 8 1 4 8 9 0
10 1 11

Mini-batch SGD

- Solution?
  - Truncate the sentences into the same fixed length



3 2 1 5 6
7 8 1 4 8 9 3
10 1 11

Max len = 3

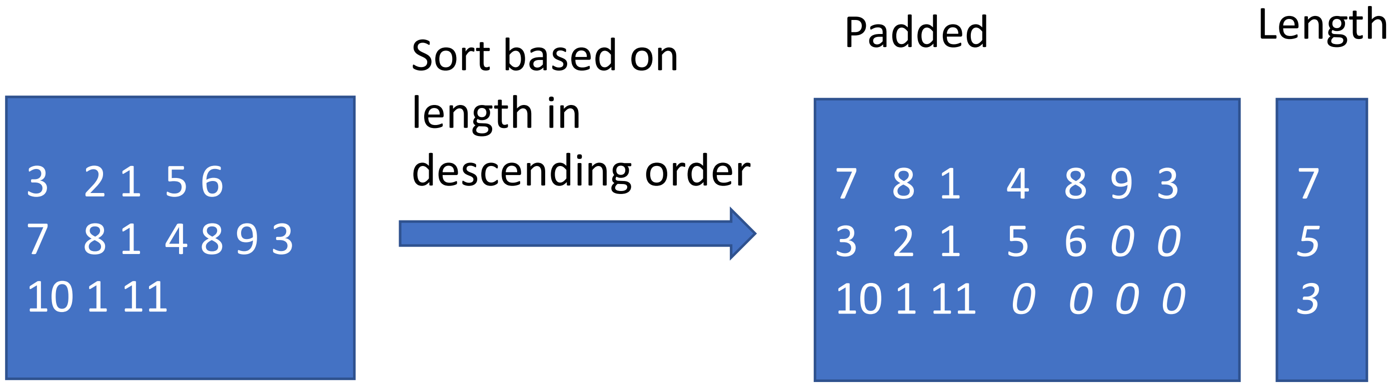
3 2 1 ~~5 6~~
7 8 1 ~~4 8 9 3~~
10 1 11

Mini-batch SGD

- Solution
  - Truncate the sentences into the same fixed length
  - Padding
    - E.g. PyTorch

```
pack = torch.nn.utils.rnn.pack_padded_sequence(batch_in, seq_lengths, batch_first=True)
```

    - Index 0 is for a special ‘PAD’ symbol. Index of words in the vocabulary starts from 1.



3 2 1 5 6
7 8 1 4 8 9 3
10 1 11

Sort based on length in descending order

Padded

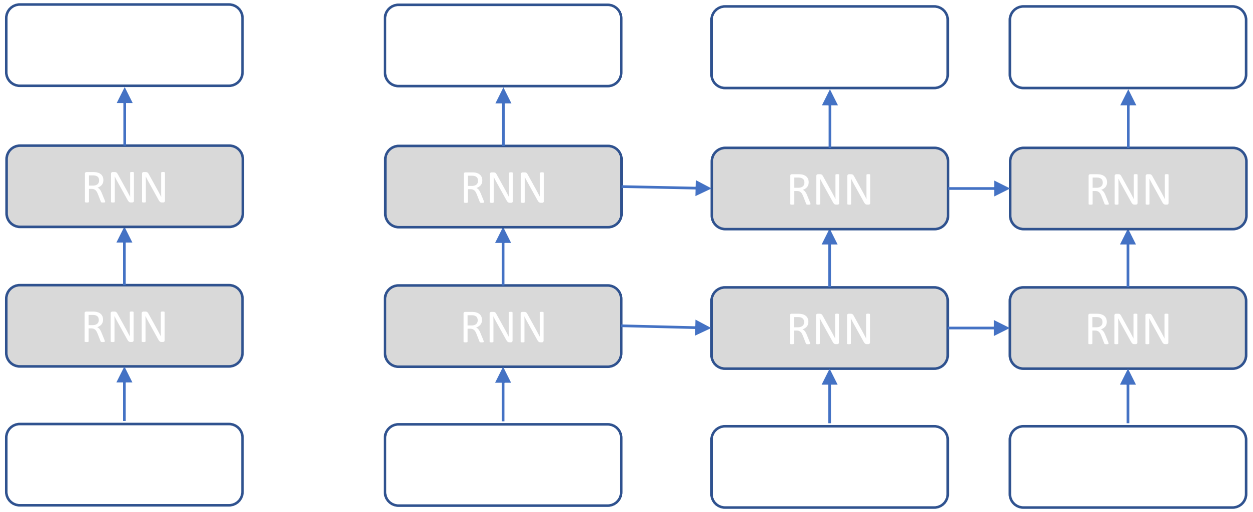
7 8 1 4 8 9 3
3 2 1 5 6 *0 0*
10 1 11 *0 0 0 0*

Length

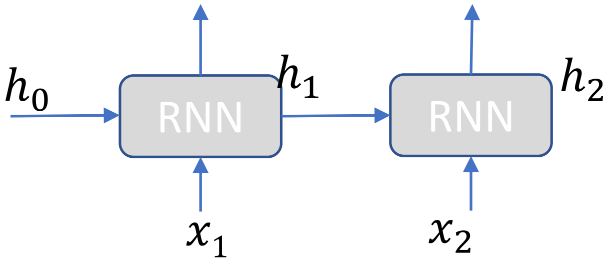
7
5
3

Other tricks for training

- Adaptive learning rate
  - E.g. Adam, RMSProp
- Normalizing the losses
  - $L = \sum_t L_t$ -> $L = \frac{1}{n \sum_t L_t}$
- Use gated RNN units
  - LSTM or GRU (not introduced yet)
- Stack multiple RNN layers
  - As shown by the right figure
- Layer normalization [8, 10]
  - Applied before activation function
- Recurrent Dropout [9, 10]



Other tricks for training



- Learn the initial state h_0 [11]
  - Typically, we set h0 to be a all 0 vector
  - It can also be learned like a bias vector
    - computing the gradient and then apply SGD update
- Trunked BPTT
  - Some sentences are very long, e.g. > 1000 positions.
  - Split the sentence into shorter sub-sentences, e.g. 200
    - Each sub-sentence is a new training sample
    - Use the last hidden vector (h_n) of the previous sub-sentence as the initial state h_0 for the next sub-sentence

Reference

- [1] https://www.quora.com/What-are-differences-between-recurrent-neural-network-language-model-hidden-markov-model-and-n-gram-language-model
- [2] https://code.google.com/archive/p/word2vec/
- [3] https://nlp.stanford.edu/projects/glove/
- [4] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber. LSTM: A Search Space Odyssey. https://arxiv.org/abs/1503.04069
- [5] http://web.stanford.edu/class/cs224n/lectures/cs224n-2017-lecture8.pdf
- [6] http://www.deeplearningbook.org/contents/applications.html (12.4.3)
- [7] Quoc V. Le, Navdeep Jaitly, Geoffrey E. Hinton**.** A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. 2015. arxiv.org/abs/1504.00941v2
- [8] “Layer Normalization" Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton. https://arxiv.org/abs/1607.06450.
- [9] “Recurrent Dropout without Memory Loss" Stanislau Semeniuta, Aliaksei Severyn, Erhardt Barth. https://arxiv.org/abs/1603.05118
- [10] https://www.tensorflow.org/api_docs/python/tf/contrib/rnn/LayerNormBasicLSTMCell
- [11] https://r2rt.com/non-zero-initial-states-for-recurrent-neural-networks.html
- [12] LSTM: A Search Space Odyssey. Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber. https://arxiv.org/abs/1503.04069
- [13] https://github.com/karpathy/char-rnn/issues/138#issuecomment-162763435
- https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html
- https://danijar.com/tips-for-training-recurrent-neural-networks/