Recurrent Neural Networks (RNN)

CS5242

Lee Hwee Kuan & **Wang Wei**

Teaching assistant:

Connie Kou Khor Li, Ji Xin, Ouyang Kun

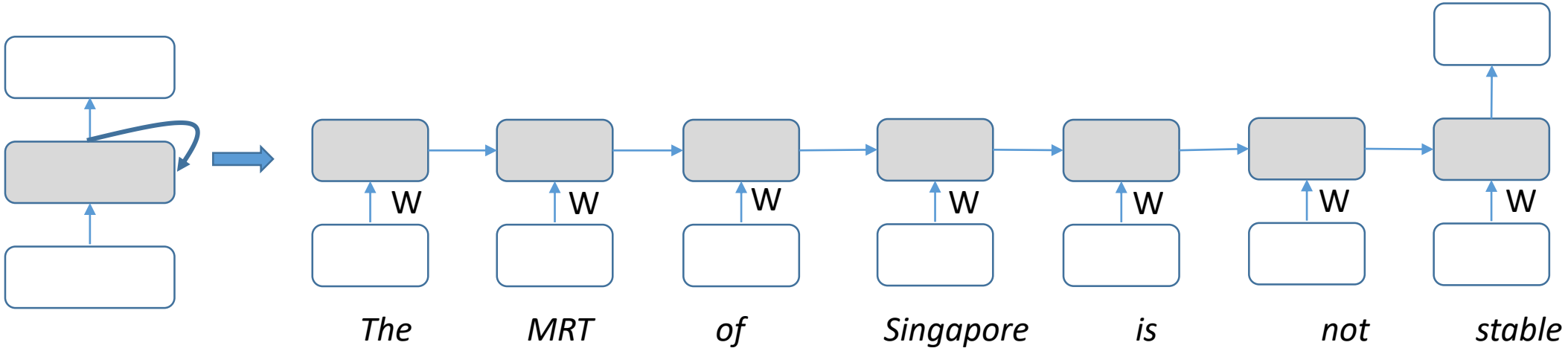
cs5242@comp.nus.edu.sg

Administrative

- Quiz 20%
  - Oct. 26, 18:30-19:30, open book
- Assignment 2: 15%, Due date: 22 Oct. 11:59PM (extended)
- Assignment 3: to be announced in two weeks, due in reading week.
- Projects
  - Submission due date, 10 Nov. 23:59
  - Report due date, 16 Nov, 16:00
  - Top-k groups will be selected for presentation, slides due date 12 Nov. 23:59
- Saturday session
  - 15:30-17:30
  - LT 19
  - CNN applications, others ? IVLE

Recap

- RNN
  - processes sequential data by applying the same transformation recurrently
    - Tied weights
  - Hidden feature from position/time t summarizes history
  - Inputs of any length

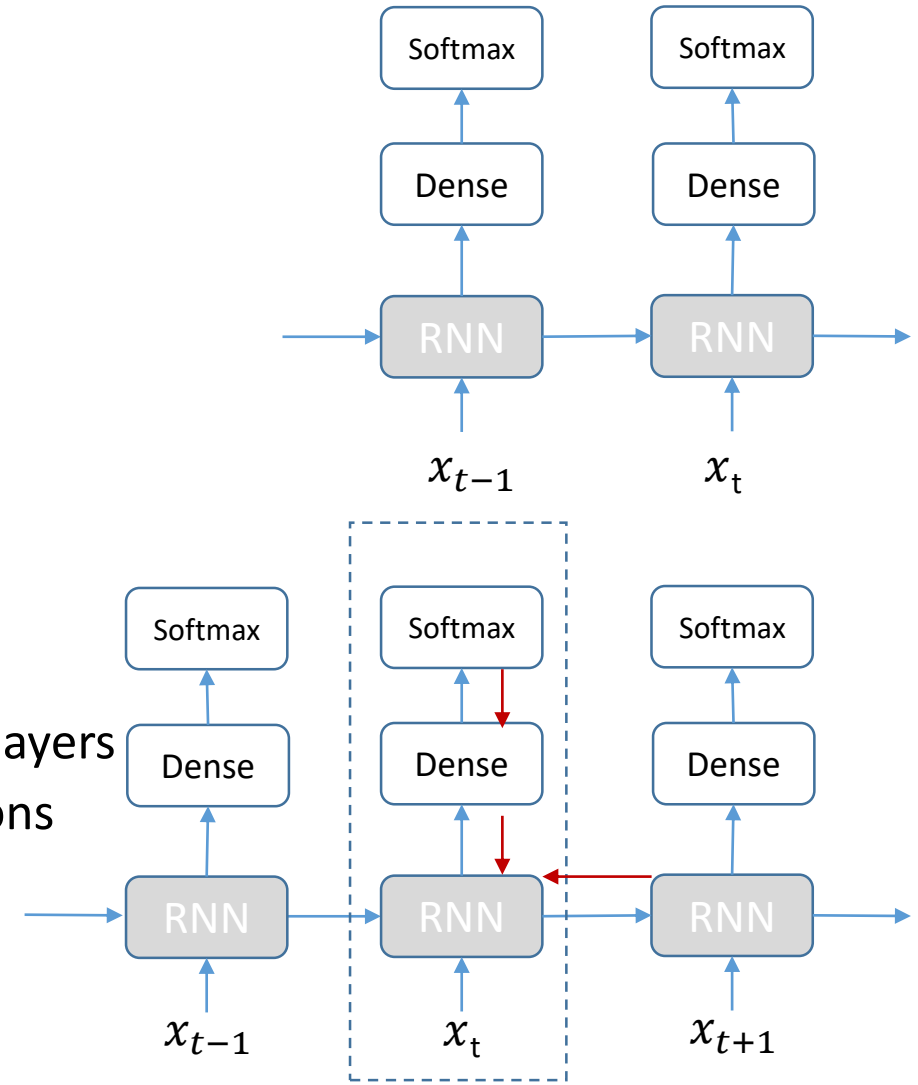


Recap

- Vanilla RNN for language modelling
  - Given a corpus D of text (e.g. sentences), model the probability of a sentence (i.e. a sequence of words)
    - $max \sum_{x \in D} P(x_1, x_2, \dots, x_n)$, x_i represents a word
  - $P(x_1, x_2, \dots, x_n) = \prod_t P(x_t | x_{t-1}, x_{t-2}, \dots, x_1)$
    - $logP(x_1, x_2, \dots, x_n) = \sum_t logP(x_t | x_{t-1}, x_{t-2}, \dots, x_1)$
  - For sentences from the training corpus, we train the RNN parameters to maximize the log-likelihood
    - → minimize negative log-likelihood -$\sum_t logP(x_t | x_{t-1}, x_{t-2}, \dots, x_1)$
    - → minimize the cross-entropy loss $-(l_t == k) \log P(x_t = k | x_{t-1}, x_{t-2}, \dots, x_1)$ for all t
      - l_t is the ground truth word at position t, k is the predicted word
    - → a classification problem
      - Given $x_{t-1}, x_{t-2}, \dots, x_1$, predict x_t
      - *The MRT of Singapore → is*

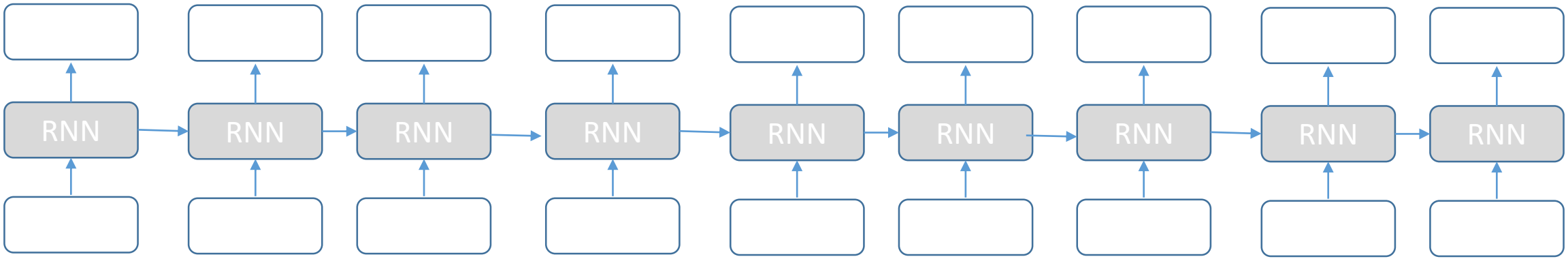
Recap

- Back-propagation through time for training
  - forward
    - $h_t = f(h_{t-1}, x_t|\theta)$
    - $o_t = Vh_t + c$
    - $y_t = softmax(o_t)$
  - Backward
    - Like BP for MLP for each position
    - Hidden layer aggregates gradients from top and right layers
    - Gradients of parameters are aggregated across positions
  - Gradient exploding or vanishing
    - W is multiplied repeatedly



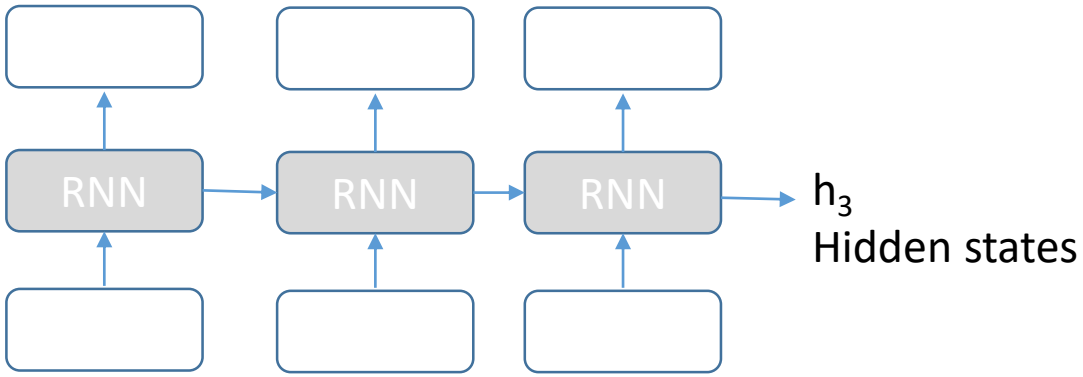
Recap

- Tricks
  - Gradient clipping for gradient exploding
  - Truncated BPTT for very long sequence
    - Gradient vanishing
    - Efficiency (less memory for intermediate results)



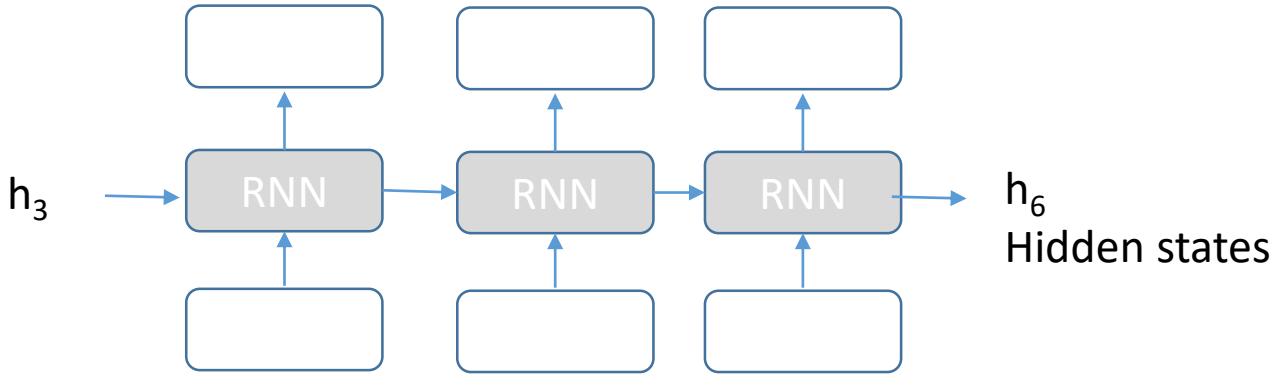
Recap

- Tricks
  - Gradient clipping for gradient exploding
  - Truncated BPTT for very long sequence
    - Gradient vanishing
    - Efficiency (less memory for intermediate results)
      - The memory for storing the hidden states will be erased after training each sub-sentence



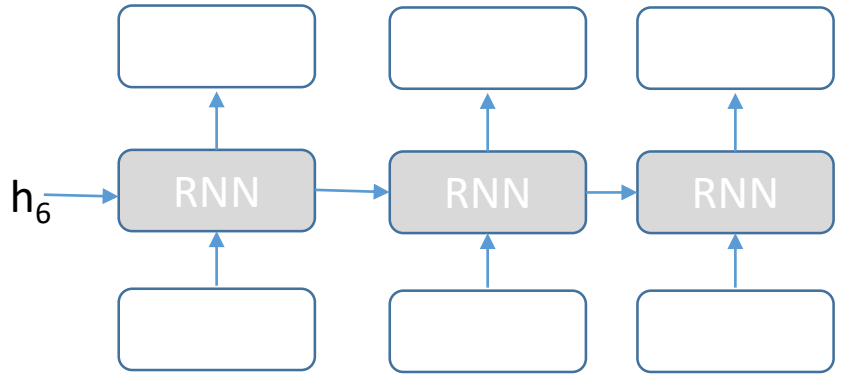
Recap

- Tricks
  - Gradient clipping for gradient exploding
  - Truncated BPTT for very long sequence
    - Gradient vanishing
    - Efficiency (less memory for intermediate results)



Recap

- Tricks
  - Gradient clipping for gradient exploding
  - Truncated BPTT for very long sequence
    - Gradient vanishing
    - Efficiency (less memory for intermediate results)



Roadmap

- LSTM
- GRU
- Bi-directional RNN

- Machine translation
- Question answering
- Image caption generation

Intended learning outcomes

01

Compare vanilla RNN, GRU and LSTM

02

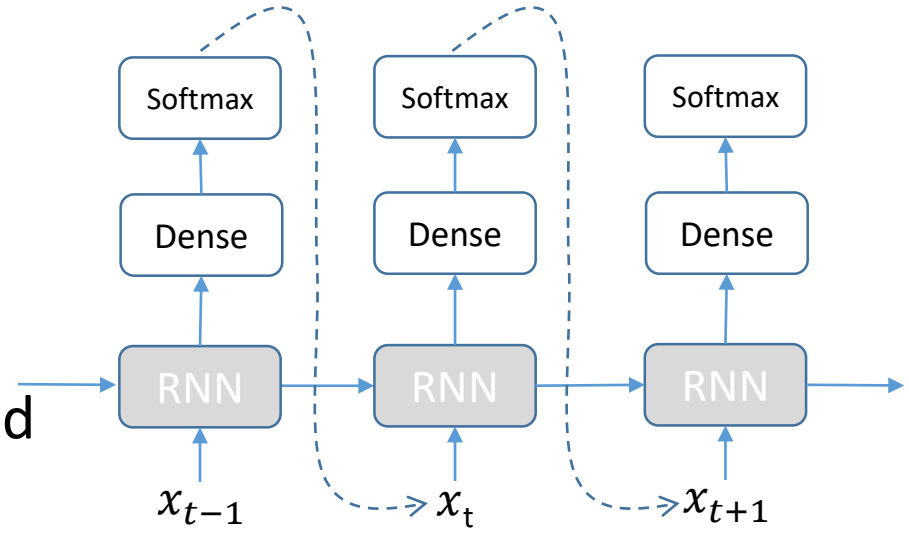
Implement the BP and inference of RNN

03

Understand the RNN architectures for different applications

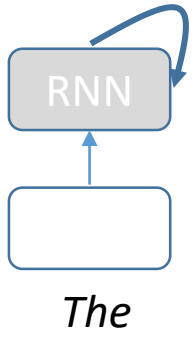
Inference

- Input some seeding words
  - A special word for the starting of a sentence
    - START
  - Or a few words, “The MRT of”
- Generate the rest words one by one
  - The output from position (or timestamp) t-1 is used as the input of position t, i.e. x_t
  - Until a special word for the ending of a sentence
    - END (EOS)



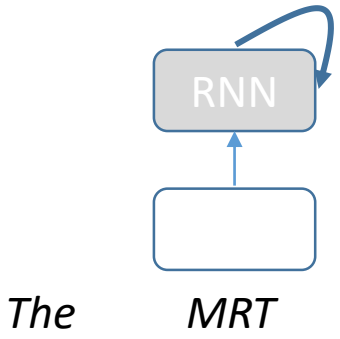
Inference

- Example
  - Input “The MRT of”



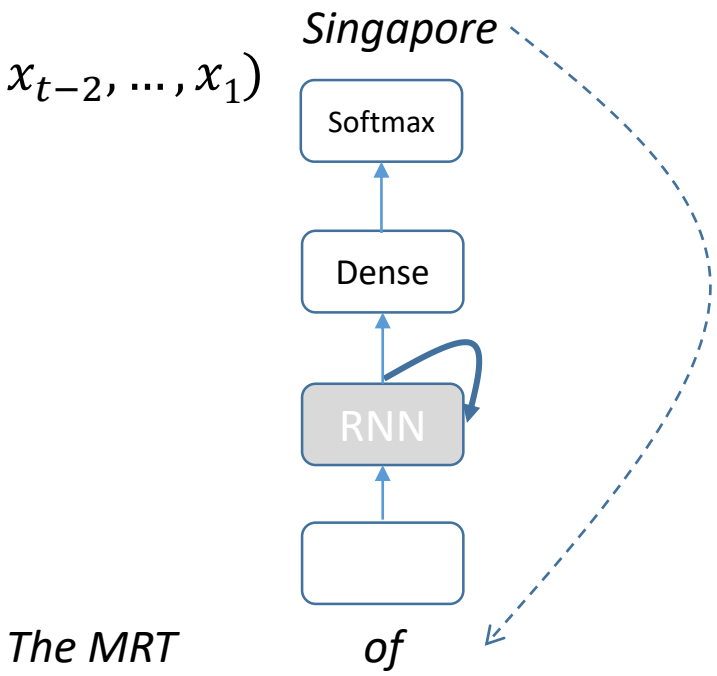
Inference

- Example
  - Input “The MRT of”



Inference

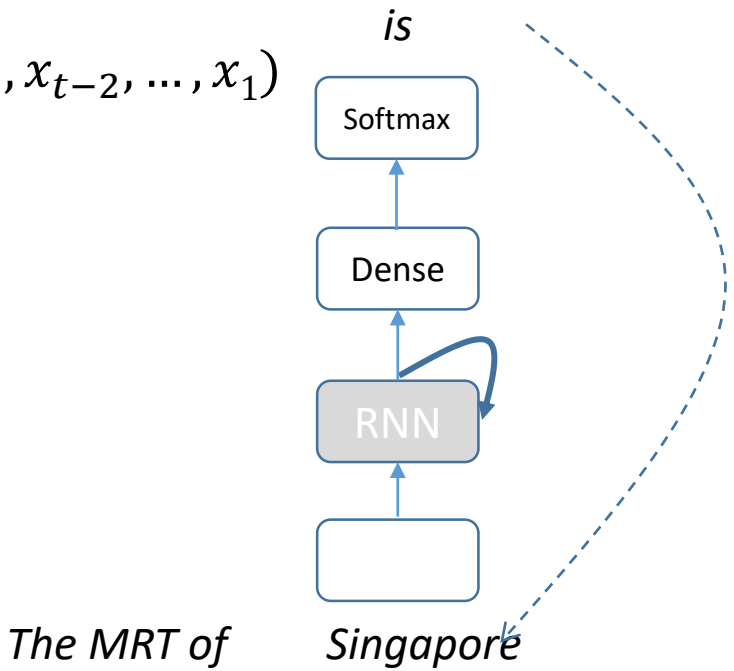
- Example
  - Input “The MRT of”
  - Greedy search
    - Simple, P($x_t|x_{t-1}, x_{t-2}, \dots, x_1$)



Singapore: 0.8
MRT : 0.1
Is : 0.05
Stable : 0.04
…

Inference

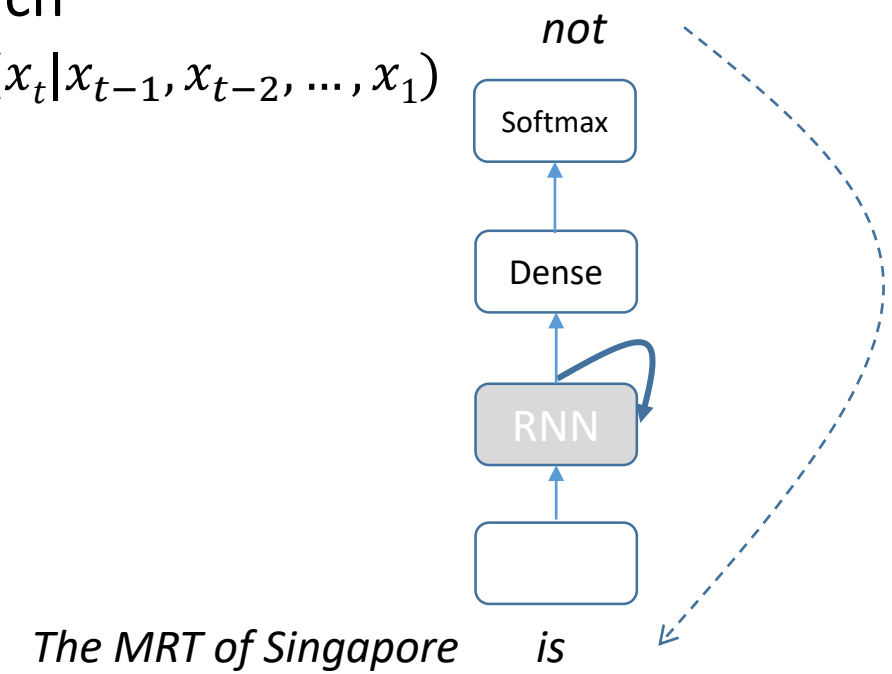
- Example
  - Input “The MRT of”
  - Greedy search
    - Simple, P($x_t|x_{t-1}, x_{t-2}, \dots, x_1$)



Is : 0.9
Singapore: 0.05
MRT : 0.025
Stable : 0.01
…

Inference

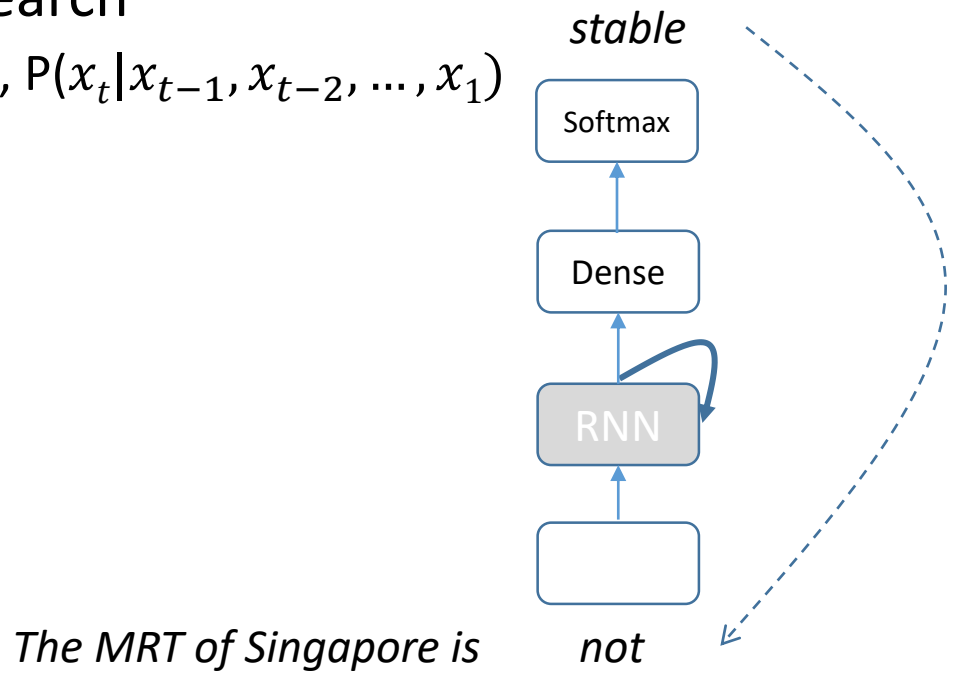
- Example
  - Input “The MRT of”
  - Greedy search
    - Simple, P($x_t|x_{t-1}, x_{t-2}, \dots, x_1$)



Not : 0.6
Stable : 0.3
Singapore: 0.05
MRT : 0.025
…

Inference

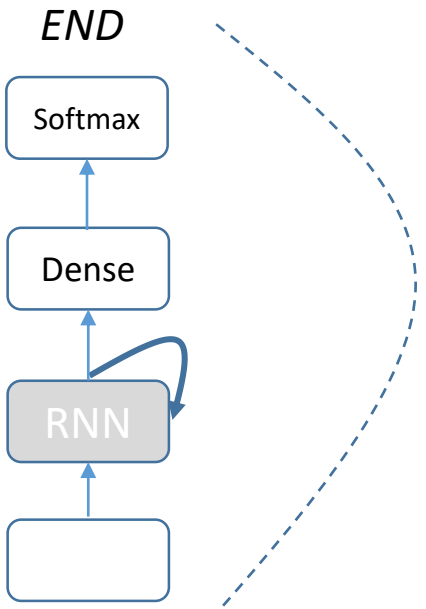
- Example
  - Input “The MRT of”
  - Greedy search
    - Simple, P($x_t|x_{t-1}, x_{t-2}, \dots, x_1$)



Stable : 0.8
Singapore: 0.1
MRT : 0.025
Not : 0.001
…

Inference

- Example
  - Input “The MRT of”
  - Greedy search
    - Simple, P($x_t|x_{t-1}, x_{t-2}, \dots, x_1$)

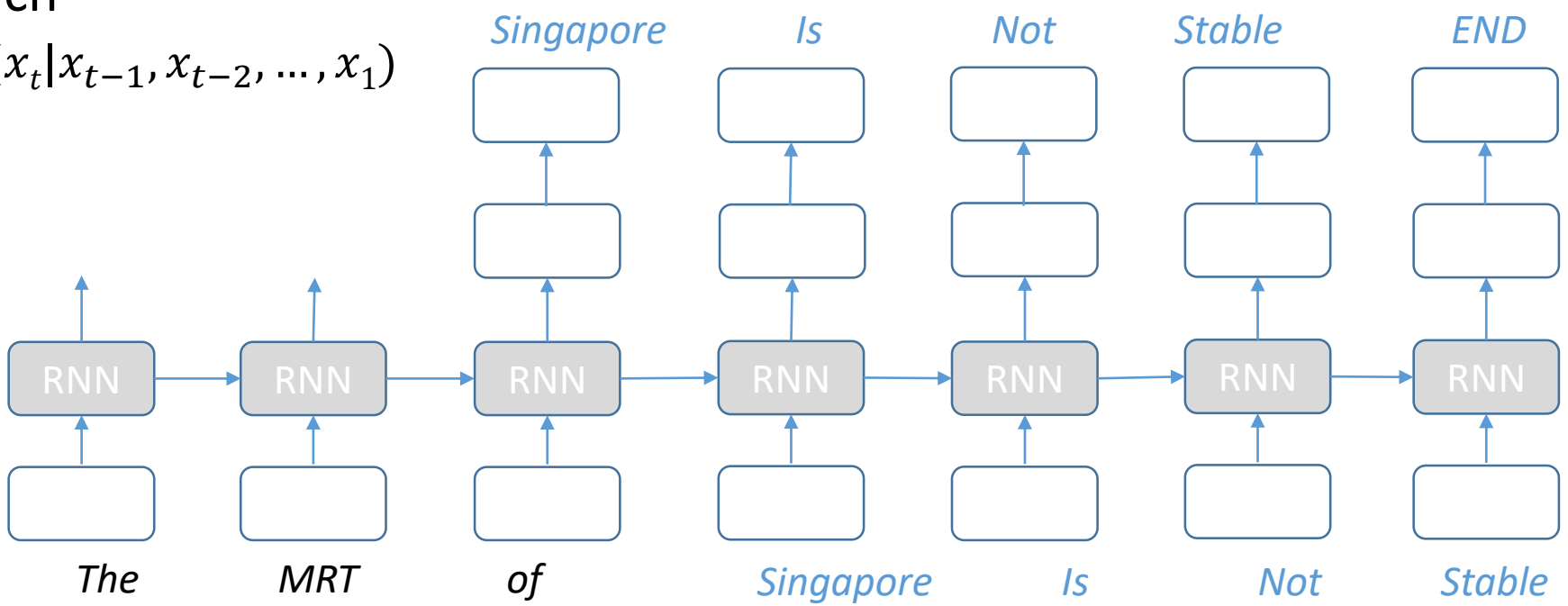


END : 0.7
MRT : 0.1
Is : 0.05
Stable : 0.04
…

The MRT of Singapore is not *stable*

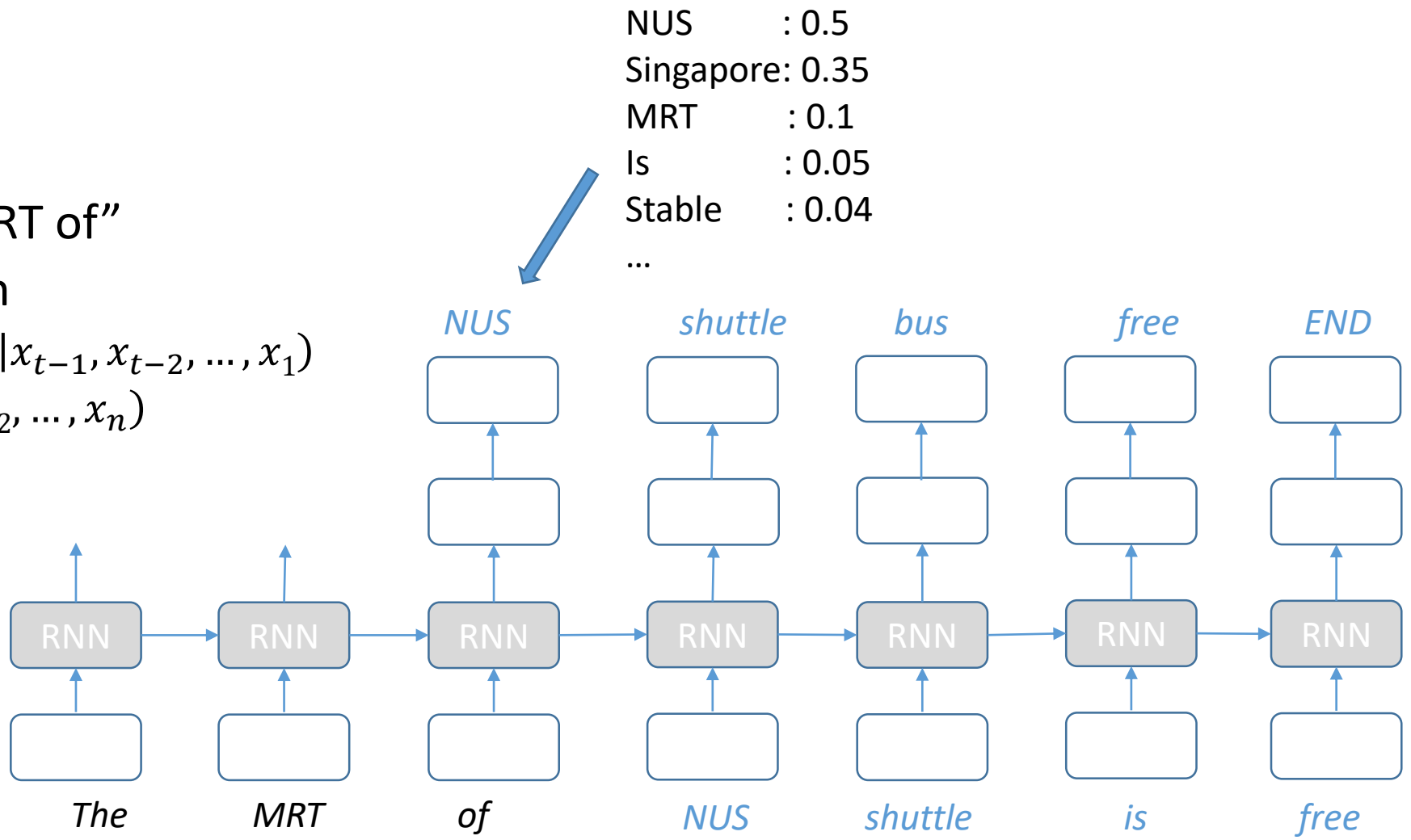
Inference

- Example
  - Input “The MRT of”
  - Greedy search
    - Simple, P($x_t|x_{t-1}, x_{t-2}, \dots, x_1$)
    - **?**



Inference

- Example
  - Input “The MRT of”
  - Greedy search
    - Simple, P($x_t|x_{t-1}, x_{t-2}, \dots, x_1$)
    - NOT $P(x_1, x_2, \dots, x_n)$

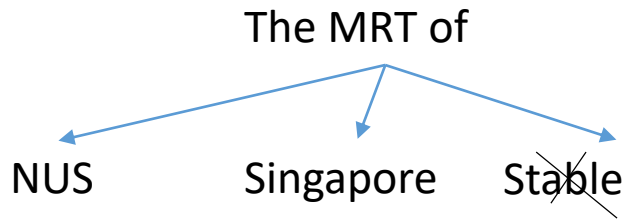


Inference

- Beam search
  - Suppose we have the top-K assignments of $(x_1, x_2, \ldots x_t)_i$ (i=1…k) for position t.
    - Sorted by $P(x_1, x_2, \ldots x_t)$
  - To select the words for $x_{t+1,}$
    - For each word w_j from the vocabulary V
      - For each top-K assignment $(x_1, x_2, \ldots x_t)_i$
        - Compute $P_{ij}=P((x_1, x_2, \ldots x_t)_i, x_{t+1}=w_j) =P(x_{t+1} =w_j \mid (x_1, x_2, \ldots x_t)_i)*P(x_1, x_2, \ldots x_t)_i$
    - Sort P_{ij} to keep the top-K assignments
- How to set K
  - 2-5 is suggested [13]

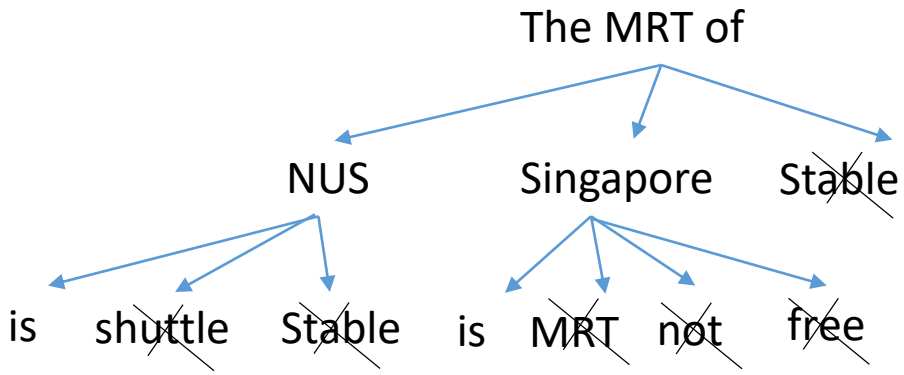
Inference

- Beam search
  - K=2



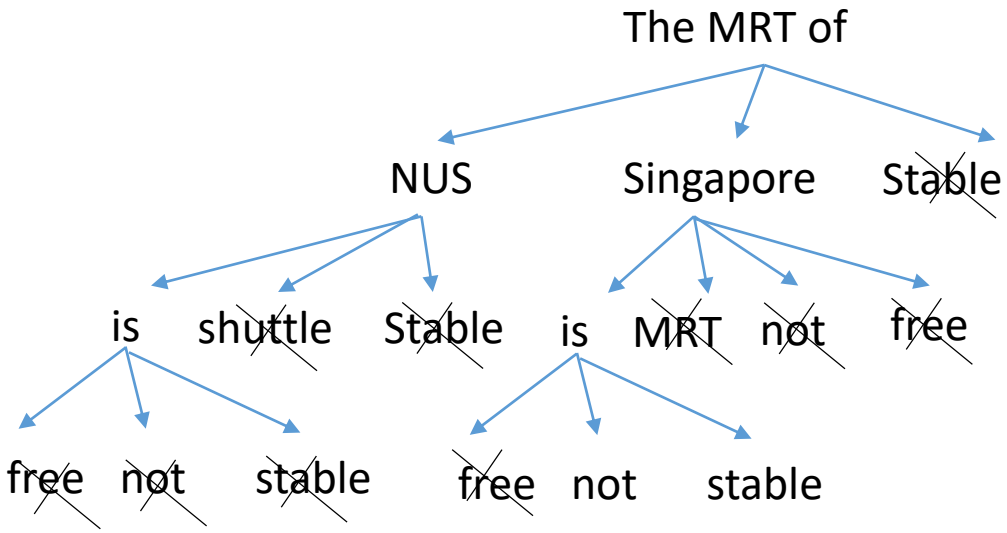
Inference

- Beam search
  - K=2



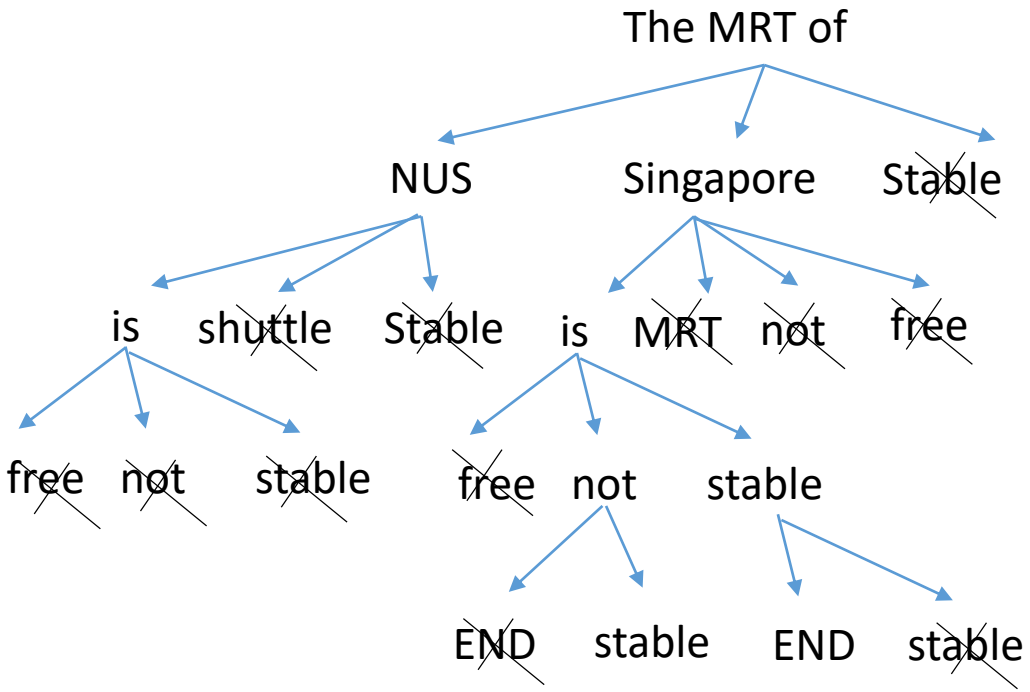
Inference

- Beam search
  - K=2



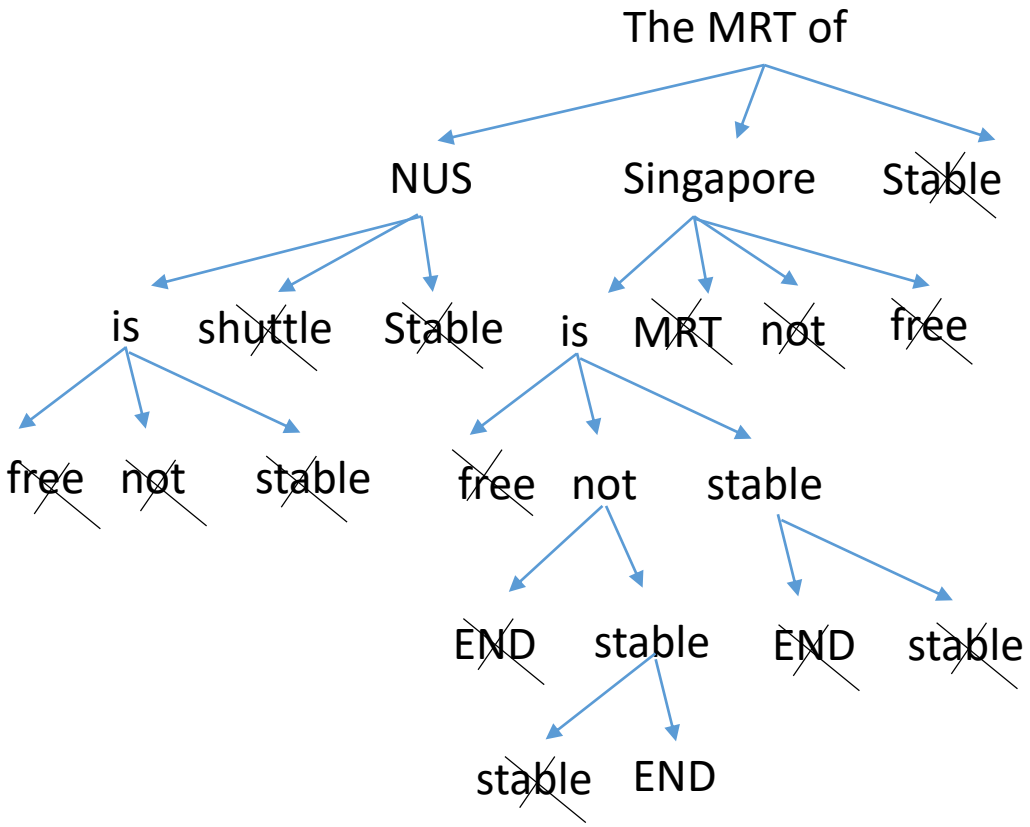
Inference

- Beam search
  - K=2



Inference

- Beam search
  - K=2



Inference

- Beam search code from
  - https://gist.github.com/udibr/67be473cf053d8c38730

```
def beamsearch(predict=keras_rnn_predict,
               k=1, maxsample=400, use_unk=False, oov=oov, empty=empty, eos=eos):
    """return k samples (beams) and their NLL scores, each sample is a sequence of labels,
    all samples starts with an `empty` label and end with `eos` or truncated to length of `maxsample`.
    You need to supply `predict` which returns the label probability of each sample.
    `use_unk` allow usage of `oov` (out-of-vocabulary) label in samples
    """

    live_k = 1 # samples that did not yet reached eos
    live_samples = [[empty]]
    live_scores = [0]

    while live_k                 :
        # for every possible live sample calc prob for every possible label
        probs = predict(live_samples, empty=empty)

        # total score for every sample is sum of -log of word prb
        cand_scores = np.array(live_scores)[:,None] - np.log(probs)
        if not use_unk and oov is not None:
            cand_scores[:,oov] = 1e20
        cand_flat = cand_scores.flatten()

        # find the best (lowest) scores we have from all possible samples and new words
        ranks_flat = cand_flat.argsort()[:(k-dead_k)]
        live_scores = cand_flat[ranks_flat]

        # append the new words to their appropriate live sample
        voc_size = probs.shape[1]
        live_samples = [live_samples[r//voc_size]+[r%voc_size] for r in ranks_flat]
```

Input->RNN->Dense->Softmax

$P(x_t \mid x_1, x_2, \ldots x_{t-1})$

$-\log P(x_1, x_2, \ldots x_{t-1}) - \log P(x_t \mid x_1, x_2, \ldots x_{t-1})$

Char-RNN

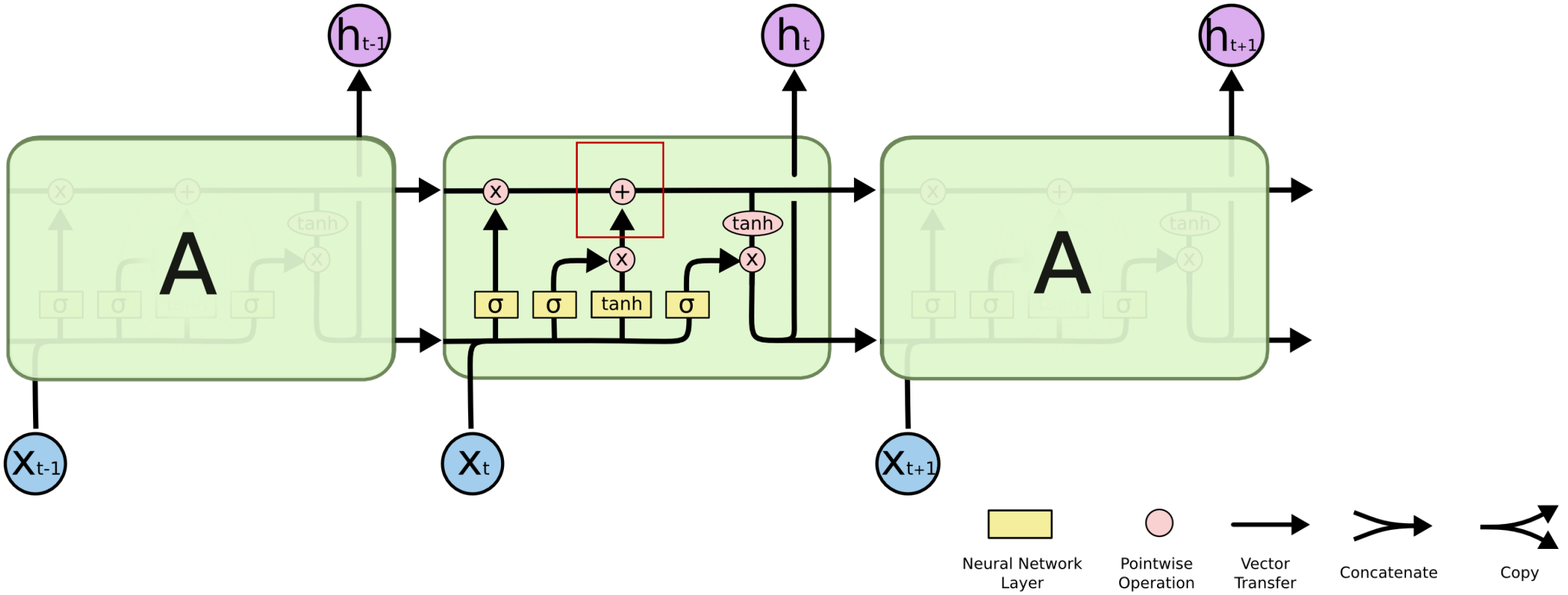
- https://gist.github.com/karpathy/d4dee566867f8291f086
- https://github.com/tensorflow/models/blob/master/tutorials/rnn/ptb/ptb_word_lm.py

LSTM, GRU and Bi-directional RNN

For gradient vanishing

- Adding gates to the RNN layer
  - Gate controls the information flow

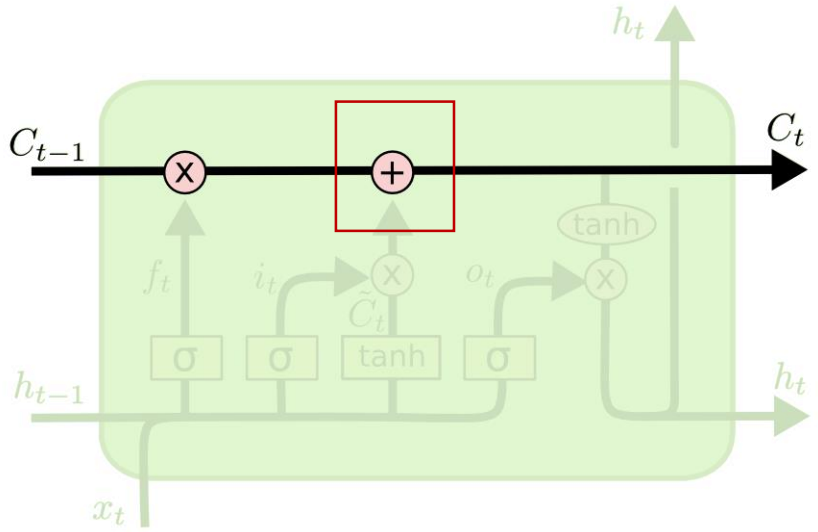
LSTM



Source from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

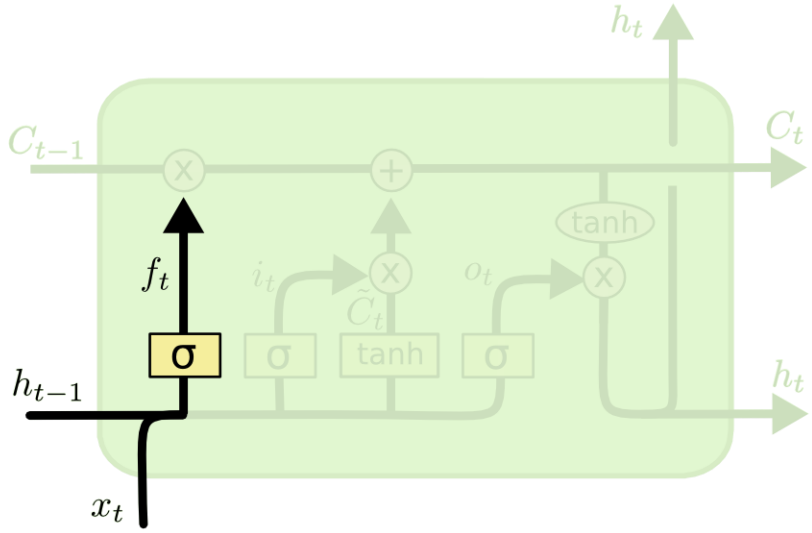
LSTM

- Leaky unit
  - C_t is a linear combination of C_{t-1} and other input
  - Easy to back-propagate gradients



LSTM

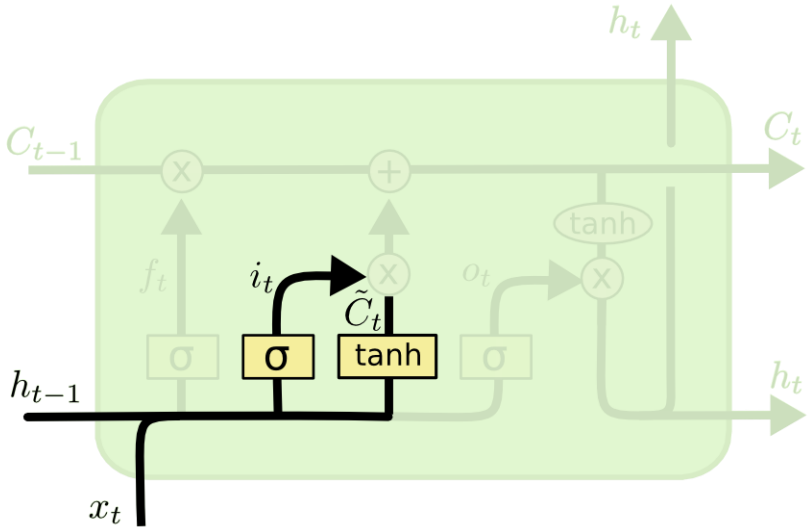
- Forget gate
  - To control the information flow about the cell state



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM

- Input gate
  - Decide how much information from the input can be added into the cell state

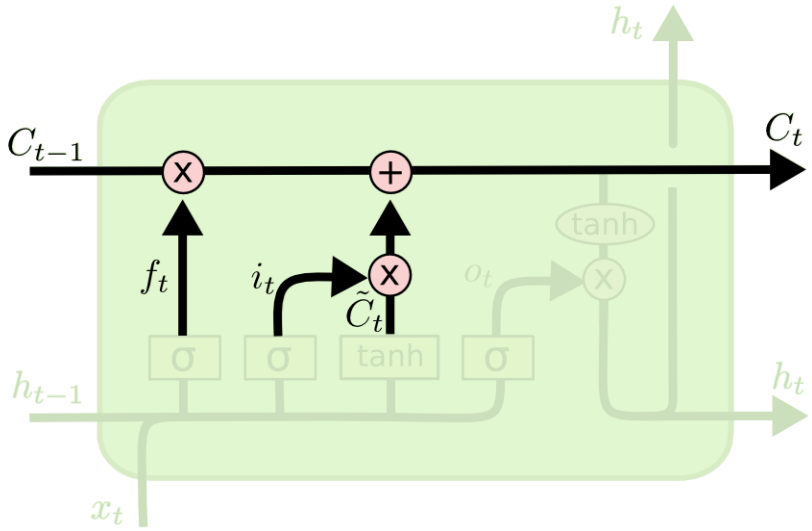


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM

- Cell state
  - Tanh (not sigmoid)?



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

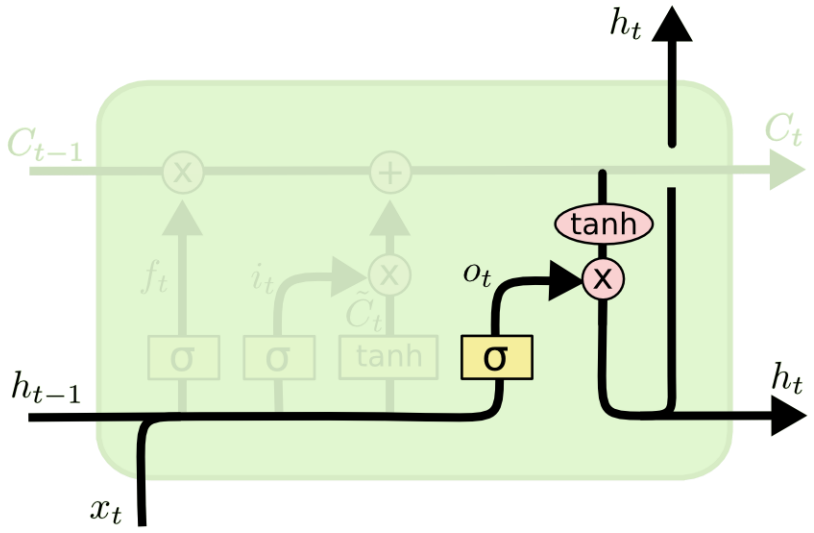
Forget gate and input gate are (0, 1). Their sum may not be 1, like γ and (1- γ)

C_t is not computed using affine transformation like $C_t = WC_{t-1} + \cdots$, hence we do not have $\frac{\partial L}{\partial C_{t-k}} = (W^T)^k \frac{\partial L}{\partial C_t}$

Then, gradient vanishing/exploding is prevented.

LSTM

- Output gate
  - Determine the information flows out of the unit



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM

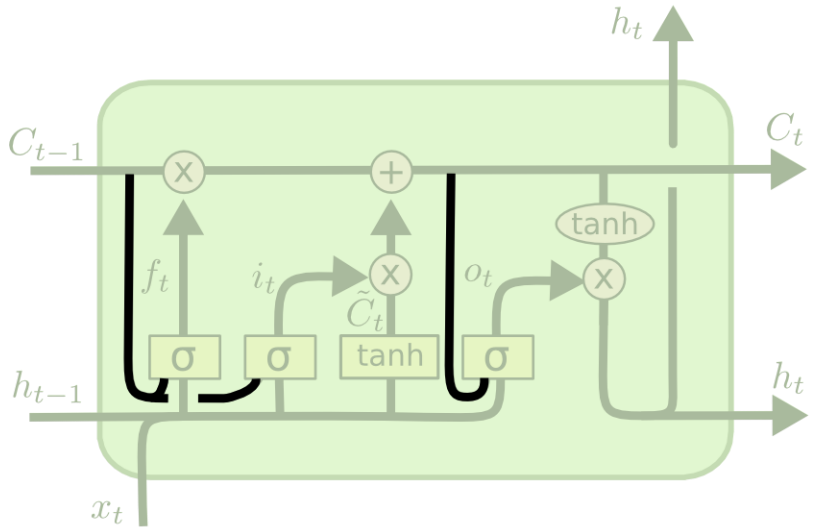
- Gates + States

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM

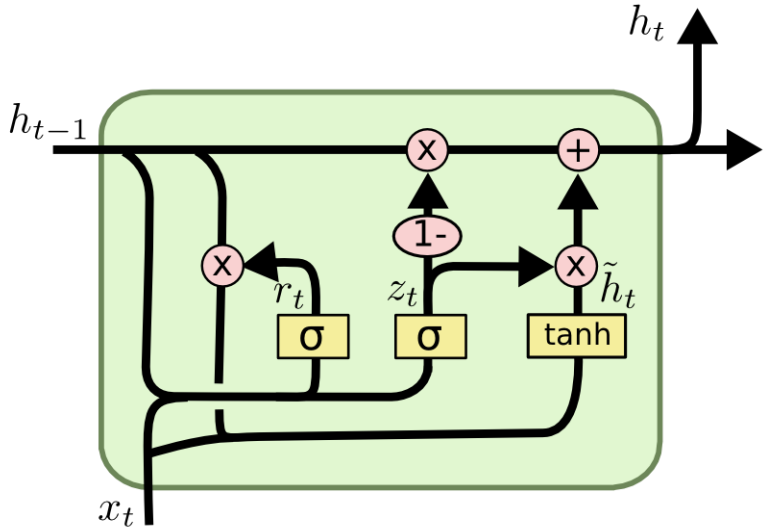
- With peephole
  - Cell state has effect on the gate values



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

- Other variants [12]

GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$ Update gate

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$ Reset gate

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

LSTM for Char-RNN

Each character is represented by a V-dim one-hot vector

```
from __future__ import print_function
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.layers import LSTM
from keras.optimizers import RMSprop
from keras.utils.data_utils import get_file
import numpy as np
import random
import sys

path = get_file('nietzsche.txt', origin='https://s3.amazonaws.com/text-datasets/nietzsche.txt')
text = open(path).read().lower()
print('corpus length:', len(text))

chars = sorted(list(set(text)))
print('total chars:', len(chars))
char_indices = dict((c, i) for i, c in enumerate(chars))
indices_char = dict((i, c) for i, c in enumerate(chars))

# cut the text in semi-redundant sequences of maxlen characters
maxlen = 40
step = 3
sentences = []
next_chars = []
for i in range(0, len(text) - maxlen, step):
    sentences.append(text[i: i + maxlen])
    next_chars.append(text[i + maxlen])
print('nb sequences:', len(sentences))
```

Truncate the character stream into sentences

```
X = np.zeros((len(sentences), maxlen, len(chars)), dtype=np.bool)
y = np.zeros((len(sentences), len(chars)), dtype=np.bool)
for i, sentence in enumerate(sentences):
    for t, char in enumerate(sentence):
        X[i, t, char_indices[char]] = 1
    y[i, char_indices[next_chars[i]]] = 1


# build the model: a single LSTM
print('Build model...')
model = Sequential()
model.add(LSTM(128, input_shape=(maxlen, len(chars))))
model.add(Dense(len(chars)))
model.add(Activation('softmax'))

optimizer = RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

https://github.com/fchollet/keras/blob/master/examples/lstm_text_generation.py

```python
# train the model, output generated text after each iteration
for iteration in range(1, 60):
    print()
    print('-' * 50)
    print('Iteration', iteration)
    model.fit(X, y,
              batch_size=128,
              epochs=1)

    start_index = random.randint(0, len(text) - maxlen - 1)

    for diversity in [0.2, 0.5, 1.0, 1.2]:
        print()
        print('----- diversity:', diversity)

        generated = ''
        sentence = text[start_index: start_index + maxlen]
        generated += sentence
        print('----- Generating with seed: "' + sentence + '"')
        sys.stdout.write(generated)

        for i in range(400):
            x = np.zeros((1, maxlen, len(chars)))
            for t, char in enumerate(sentence):
                x[0, t, char_indices[char]] = 1.

            preds = model.predict(x, verbose=0)[0]
            next_index = sample(preds, diversity)
            next_char = indices_char[next_index]

            generated += next_char
            sentence = sentence[1:] + next_char

            sys.stdout.write(next_char)
            sys.stdout.flush()
```

```python
def sample(preds, temperature=1.0):
    # helper function to sample an index from a probability array
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

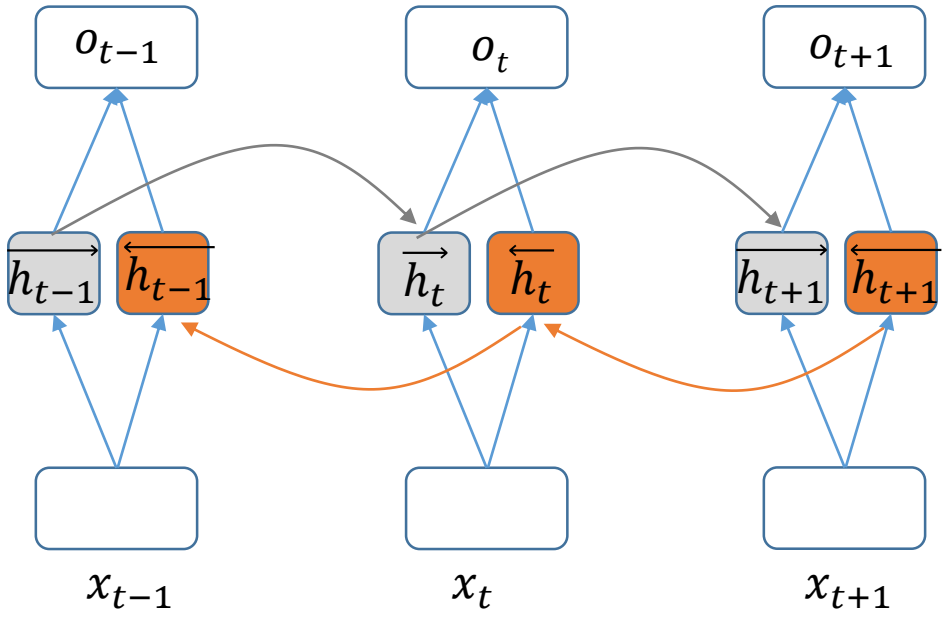
Robust Softmax

Greedy sampling

Bi-directional RNN

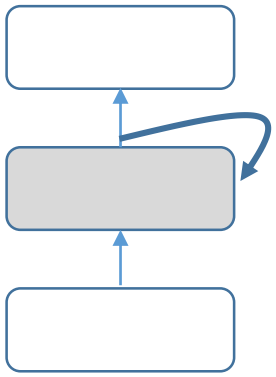
1st RNN: → a, b, c, d, x, x, x
2nd RNN: x, x, x, a, b, c, d ←
Concatenate the hidden features for the same word

- Using vanilla RNN
  - $\overrightarrow{h_t} = \tanh\left(\vec{U}x_t + \vec{W}\overrightarrow{h_{t-1}} + \vec{b}\right)$
  - $\overleftarrow{h_t} = \tanh\left(\overleftarrow{U}x_t + \overleftarrow{W}\overleftarrow{h_{t+1}} + \vec{b}\right)$
  - $o_t = V[\overrightarrow{h_t}, \overleftarrow{h_t}] + c$
    - [,] concatenate
- Using LSTM/GRU
  - $\overrightarrow{h_t} = \text{LSTM}_{\text{lr}}(\overrightarrow{h_{t-1}}, \overrightarrow{c_{t-1}}, x_t)$
  - $\overleftarrow{h_t} = \text{LSTM}_{\text{rl}}(\overleftarrow{h_{t+1}}, \overleftarrow{c_{t+1}}, x_t)$
  - $o_t = V[\overrightarrow{h_t}, \overleftarrow{h_t}] + c$
    - [,] concatenate
- Widely used for processing sentences



RNN Applications and Architectures

RNN Architectures



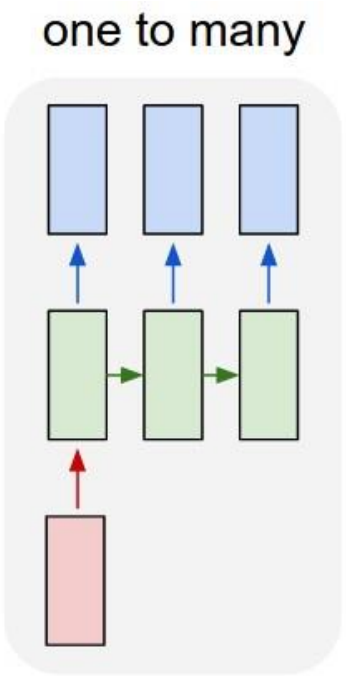
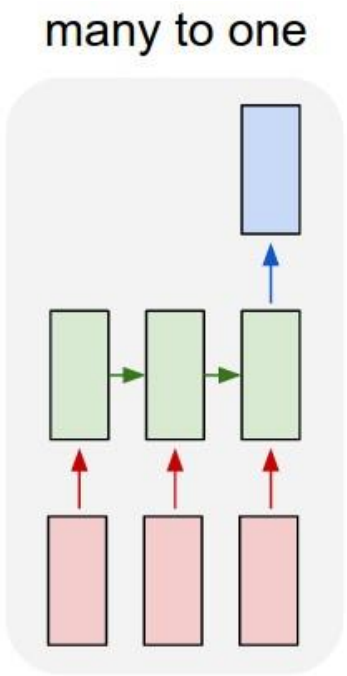
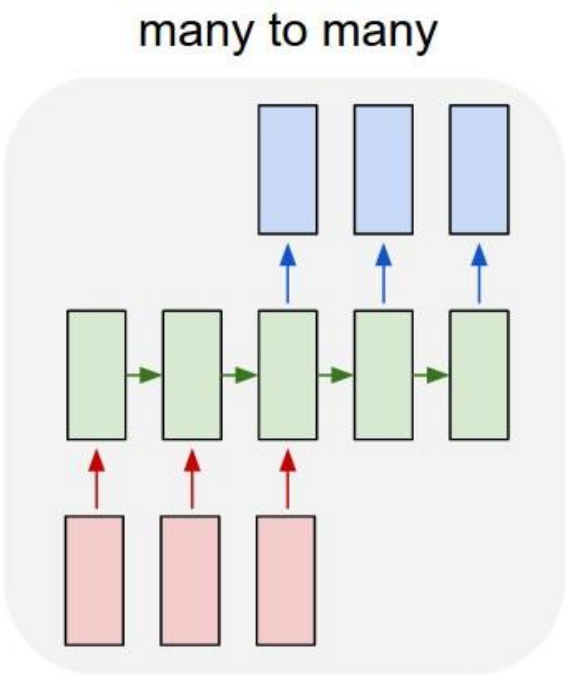


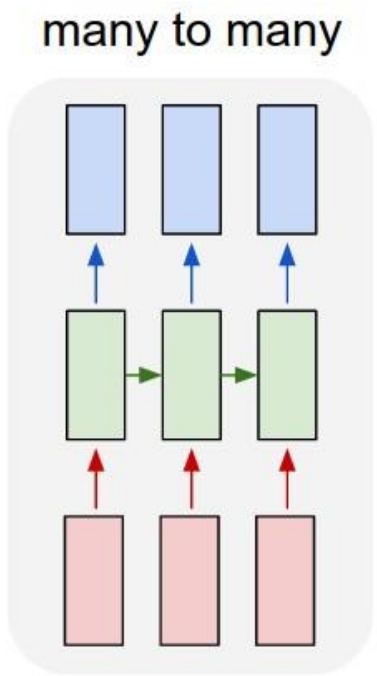
Image caption



Sentiment analysis



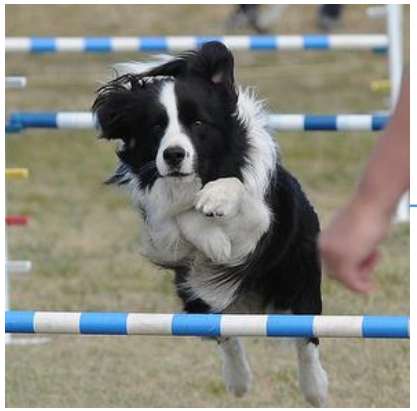
Machine translation

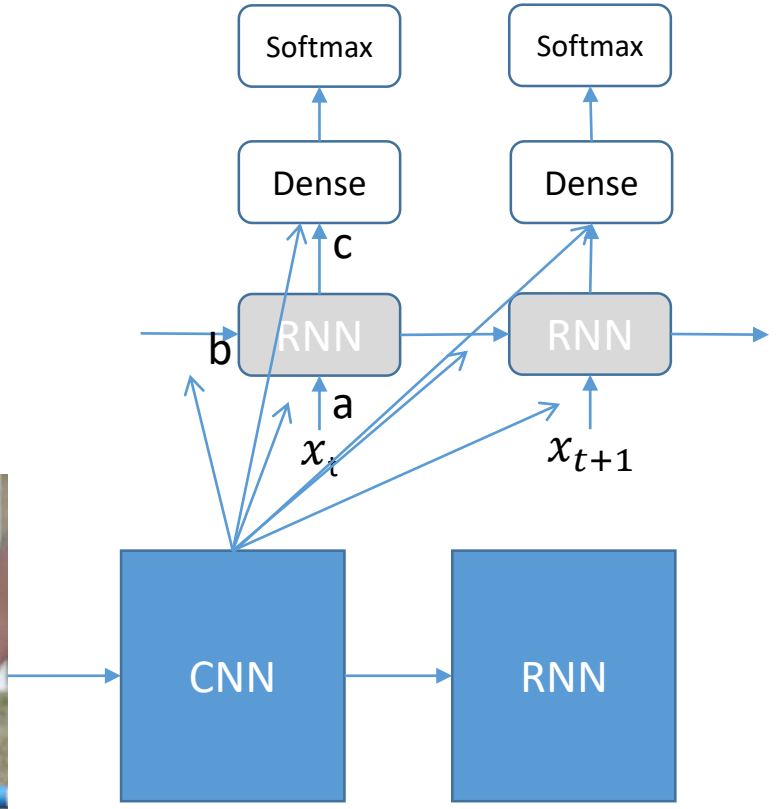


Language modelling

Image caption generation [14]

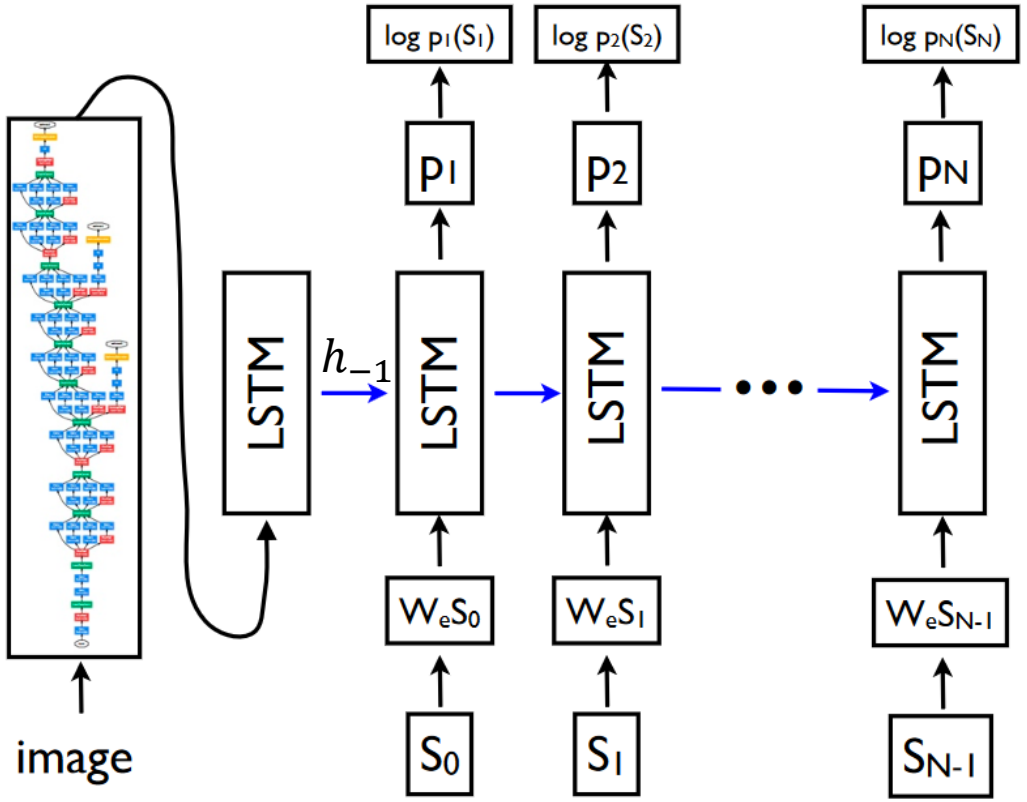
- Given an image
  - Generate a description for it
  - $P(x_1, x_2, \ldots, x_n|I) = \prod_t P(x_t|I, x_{t-1}, x_{t-2}, \ldots, x_1)$
- Datasets
  - Flickr 8K, Flickr 30K and MS COCO
- Metric
  - BULE [16]
- Solution
  - CNN for visual feature extraction
  - RNN for word generation
  - How to combine the two models?





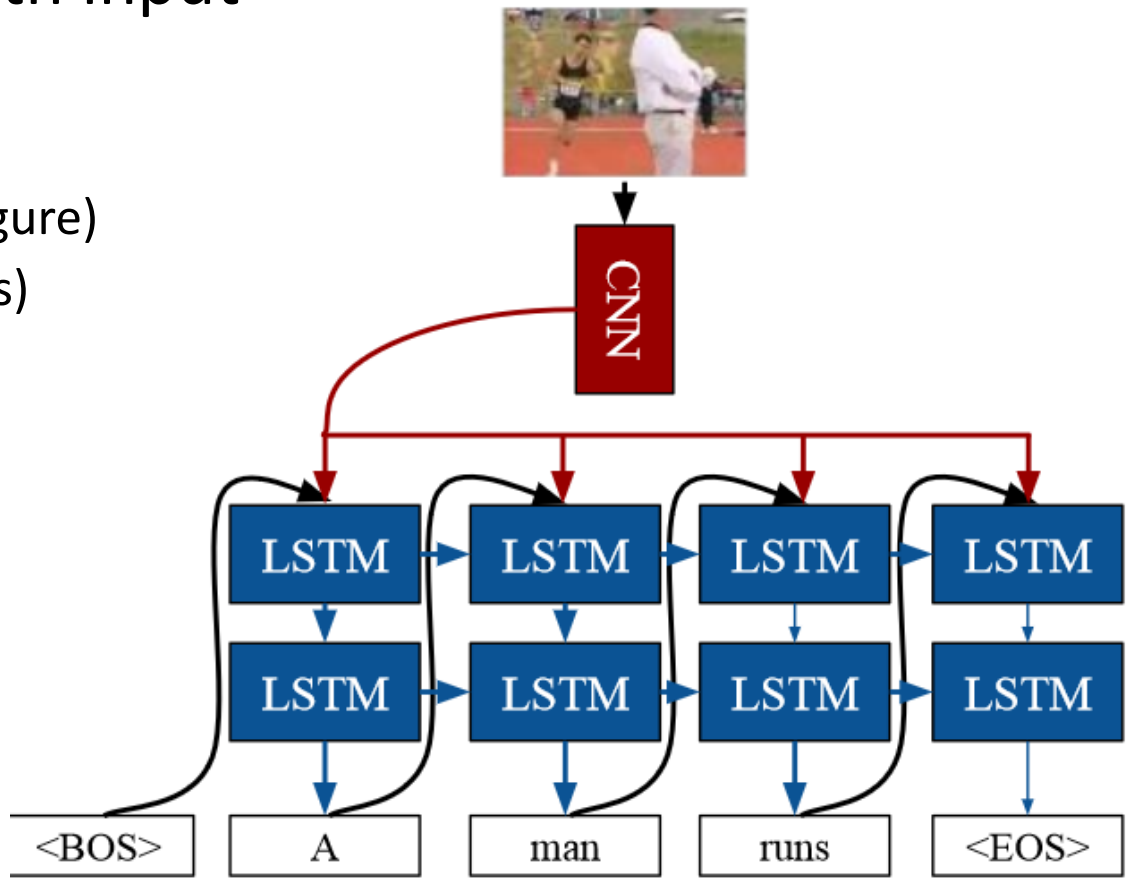
Show and Tell: A Neural Image Caption Generator [15]

- $P(x_t|I, x_{t-1}, x_{t-2}, \dots, x_1, x_0 = S_0)$
  - S_0 a special word, 'START'
  - $P(x_1|I,\ x_0 = S_0)$
    - $= P(x_1|x_0 = S_0, h_{-1})$
    - h_{-1} is generated from image I
  - $P(x_t|I, x_{t-1}, x_{t-2}, \dots, x_1, x_0 = S_0)$
    - $P(x_t|x_{t-1}, x_{t-2}, \dots, x_1, x_0 = S_0, h_{-1})$
- CNN feature is used only once
  - Inferior results if feed it into every position
- Fine-tune the top layer of CNN



Long-term Recurrent Convolutional Networks for Visual Recognition and Description [18]

- CNN feature is concatenated with input
  - Detailed experiments on
    - beam width (3-5)
    - Integration of CNN feature (right figure)
    - Fine-tuning of CNN parameters (yes)
    - LSTM VS Vanilla RNN (LSTM)
    - Multi-stack LSTM (2)

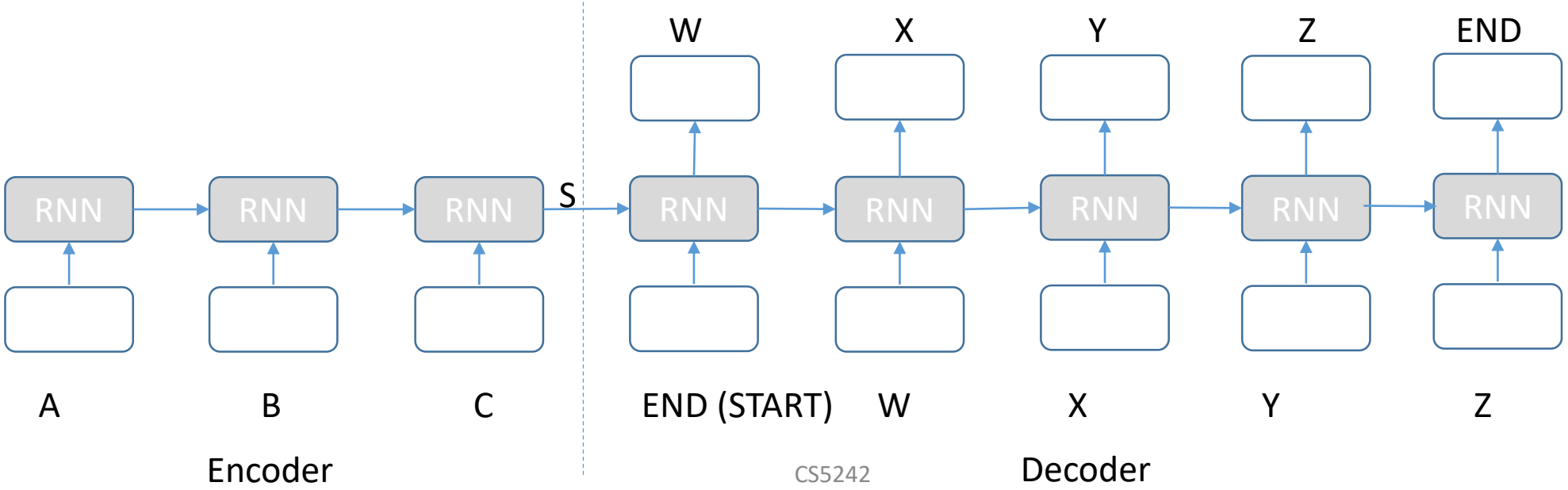


Machine translation [19]

- Given a sentence in one language, e.g. English
  - Singapore MRT is not stable
- Return a sentence in another language, e.g. Chinese
  - 新加坡地铁不稳定
- Maximize $\sum_{<x,y>} logP(y_1, y_2, \ldots, y_m | x_1, x_2, \ldots, x_n)$
  - RNN is good at processing sequence of words

Sequence to Sequence [19]

- Seq2Seq
  - $P(y_1, y_2, \ldots, y_m | x_1, x_2, \ldots, x_n) = P(y_1, y_2, \ldots, y_m | S)$
  - S is a summary of input
  - $logP(y_1, y_2, \ldots, y_m | S) = \sum_i logP(y_t | y_1, \ldots, y_{t-1}, S)$



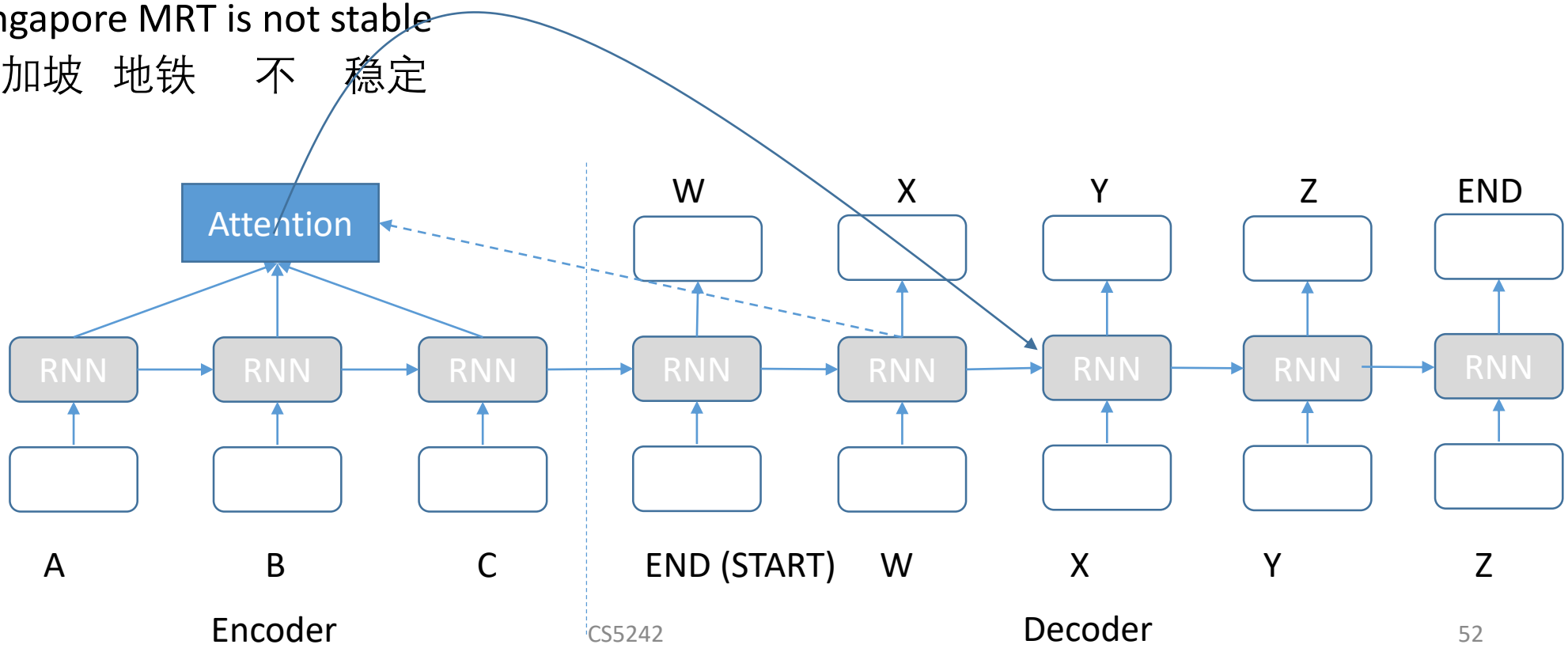
Sequence to Sequence[19]

- Seq2Seq
  - $P(y_1, y_2, \dots, y_m | x_1, x_2, \dots, x_n) = P(y_1, y_2, \dots, y_m | S)$
  - S is a summary of input
  - $logP(y_1, y_2, \dots, y_m | S) = \sum_i logP(y_t | y_1, \dots, y_{t-1}, S)$
  - Encoder and Decoder are two RNN networks
    - They have their own parameters
    - End-to-end training
  - Reverse the input sequence
    - $x_1, x_2, \dots, x_n$ → $x_n, x_{n-1}, \dots, x_1$
    - x_1 is near y_1, x_2 is near y_2, …
    - Generate correct $y_1, y_2, ..$ → overall better output sequence
    - Multiple stacks of RNN → better output sequence

Attention modelling [20]

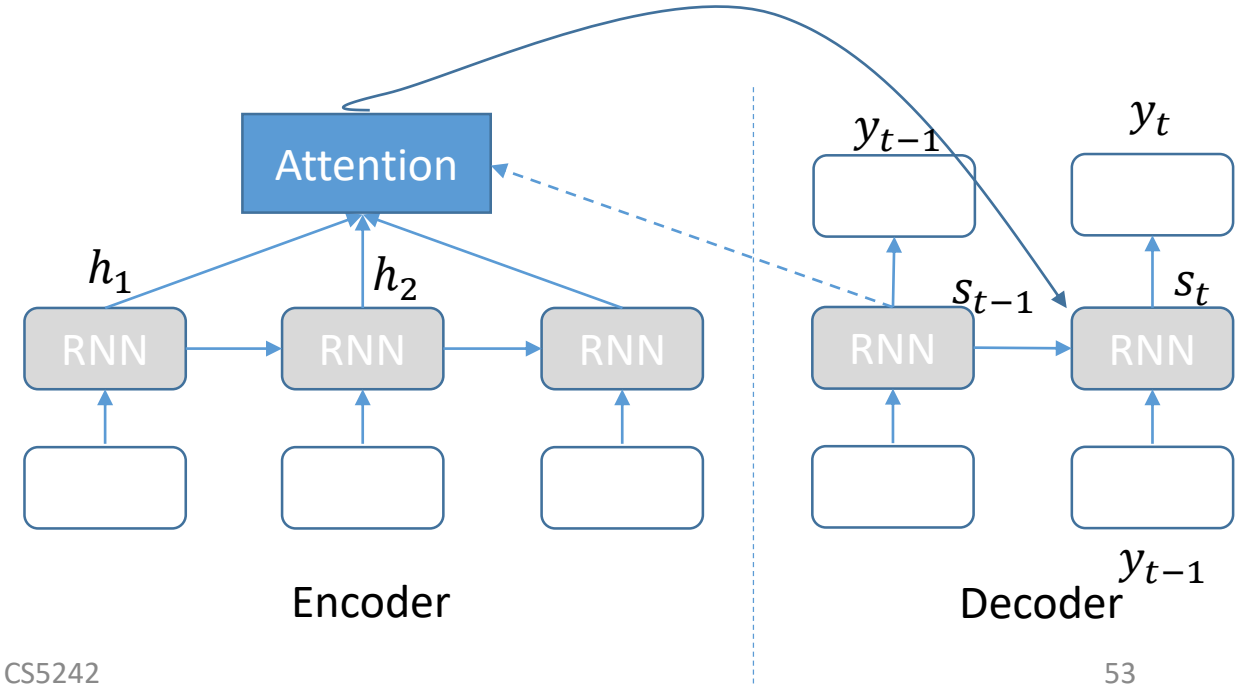
Examples will be analysed at Saturday session.

- Each output word depends on
  - All input words (hidden states), each with different contribution
    - Singapore MRT is not stable
    - 新加坡 地铁 不 稳定



Attention modelling [20]

- $logP(y_1, y_2, \dots, y_m | x_1, x_2, \dots, x_n) = \sum_t logP(y_t | y_1, \dots, y_{t-1}, x_1, x_2, \dots, x_n) =$
  $\sum_t logP(y_t | y_1, \dots, y_{t-1}, h_1, h_2, \dots h_n) = \sum_t logP(y_t | s_t) P(s_t | y_1, \dots, y_{t-1}, h_1, h_2, \dots h_n)$
- Extended GRU
  - $s_t = (1 - z_t) \circ s_{t-1} + z_t \circ \tilde{s}_t$
  - $\tilde{s}_t = \tanh(\mathrm{W}([r_t \circ s_{t-1}, \mathrm{E}y_{t-1}, c_t])$
  - $r_t = \sigma(W_r[s_{t-1}, \mathrm{E}y_{t-1}, c_t])$
  - $z_t = \sigma(W_z[s_{t-1}, \mathrm{E}y_{t-1}, c_t])$
  - $c_t = \sum_{j=1}^{n} \alpha_{tj} h_j$
    - α_{tj} attention weight
      - $\alpha_{tj=softmax(e_{tj})}$ j=1…n
      - $e_{tj} = v_a^T \tanh(W_a s_{t-1} + U_a h_j)$
    - Other attention weight?



Question answering

- Given a context, a question
- Outputs the answer
  - A word
  - A substring of the context
  - A new sentence
- P(a | q, c)
  - Representation learning for matching
  - Represent context and query as c, q or cq?
  - Represent candidate answer as a ?

Case 1

- Answer is a single word

Original Version	Anonymised Version
Context	
The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the "Top Gear" host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon "to an unprovoked physical and verbal attack." ...	the *ent381* producer allegedly struck by *ent212* will not press charges against the " *ent153* " host , his lawyer said friday . *ent212* , who hosted one of the most - watched television shows in the world , was dropped by the *ent381* wednesday after an internal investigation by the *ent180* broadcaster found he had subjected producer *ent193* " to an unprovoked physical and verbal attack . " ...
Query	
Producer **X** will not press charges against Jeremy Clarkson, his lawyer says.	producer **X** will not press charges against *ent212* , his lawyer says .
Answer	
Oisin Tymon	*ent193*

Source from [21]

Case 2

- Answer is a substring

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Source from [22]

Case 3

- Answer is free text

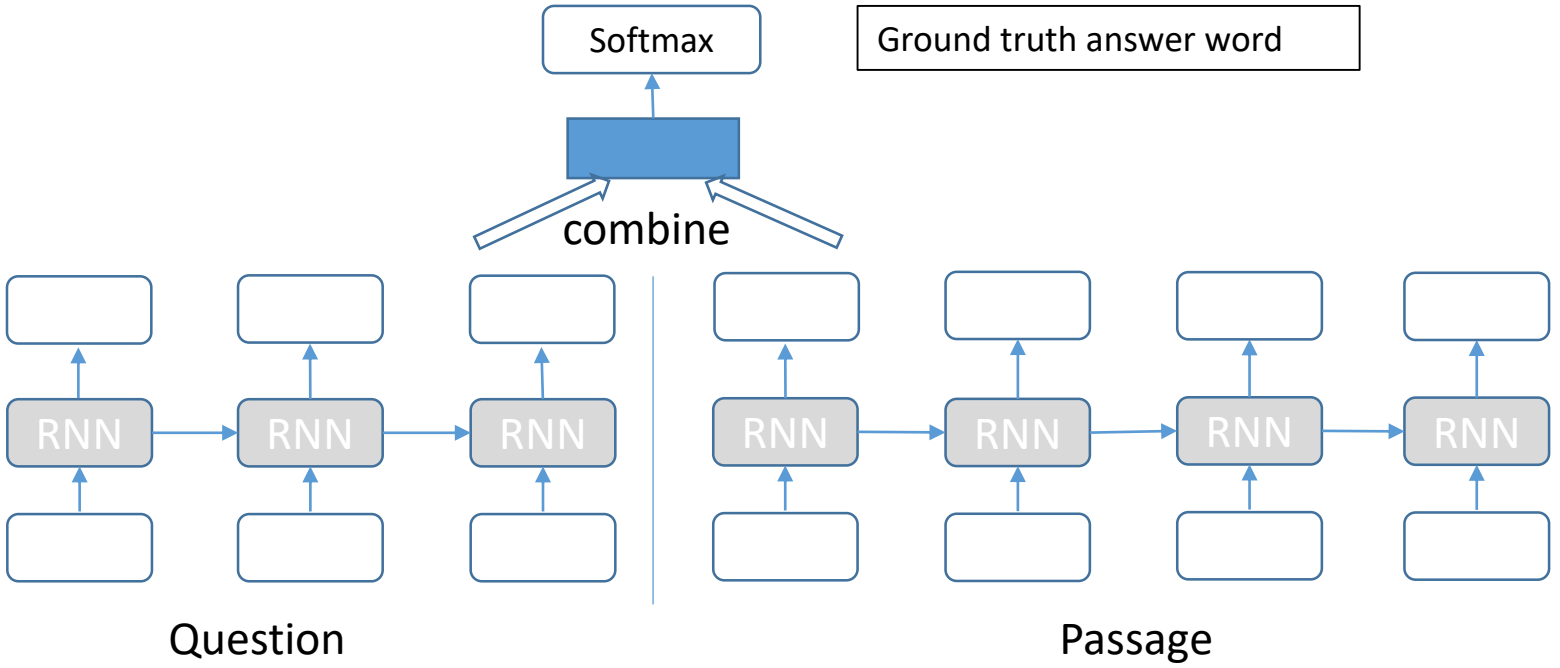
I: Jane went to the hallway.
I: Mary walked to the bathroom.
I: Sandra went to the garden.
I: Daniel went back to the garden.
I: Sandra took the milk there.
Q: Where is the milk?
A: garden
I: It started boring, but then it got interesting.
Q: What's the sentiment?
A: positive
Q: POS tags?
A: PRP VBD JJ , CC RB PRP VBD JJ .

Source from [23]

Case 1 [24]

- P(a=w|c, q) = P(a=w|cq)~similarity(w, cq)
- Steps
  1. Extract representation of question and passage (context)
  2. Combine question and context
     - Concatenation
     - Addition
  3. Generate the prediction
     - Use the combined feature to predict the answer word index
     - Matching the combined feature with each candidate word feature
       - E.g. inner-product
       - Use the similarity as input to softmax

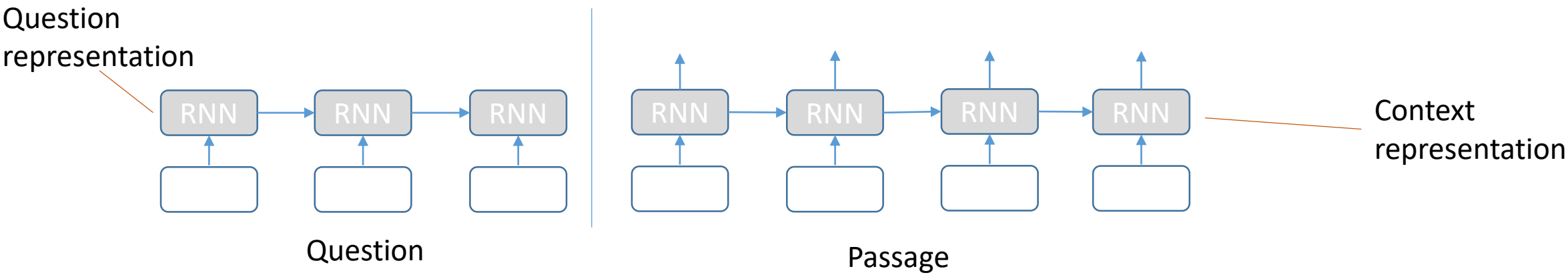
Case 1


Softmax
Ground truth answer word
combine
RNN
RNN
RNN
RNN
RNN
RNN
RNN
Question
Passage

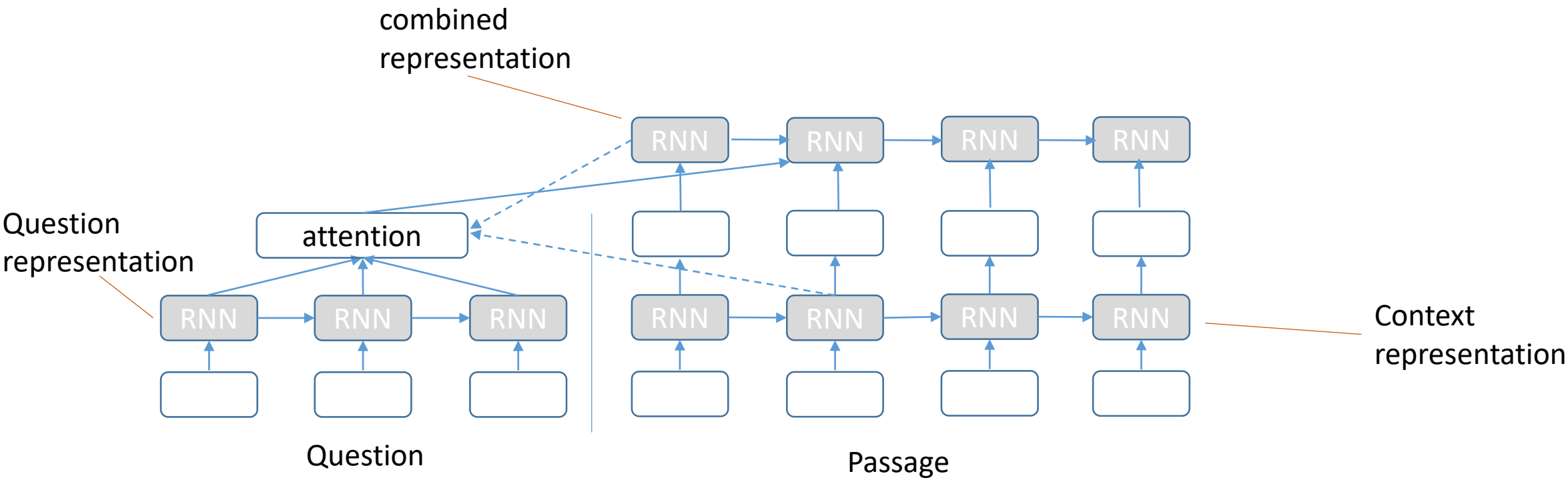
Case 2 [25]

- $P(a|q, c) = P(s, e|q, c) = P(s|q, c)\ P(e|s, q, c)$
- Steps
  - Generate question representation, q
  - Generate context representation, c
  - Combine the question and context representation cq
  - Generate the starting position, s
  - Generate the ending position, e

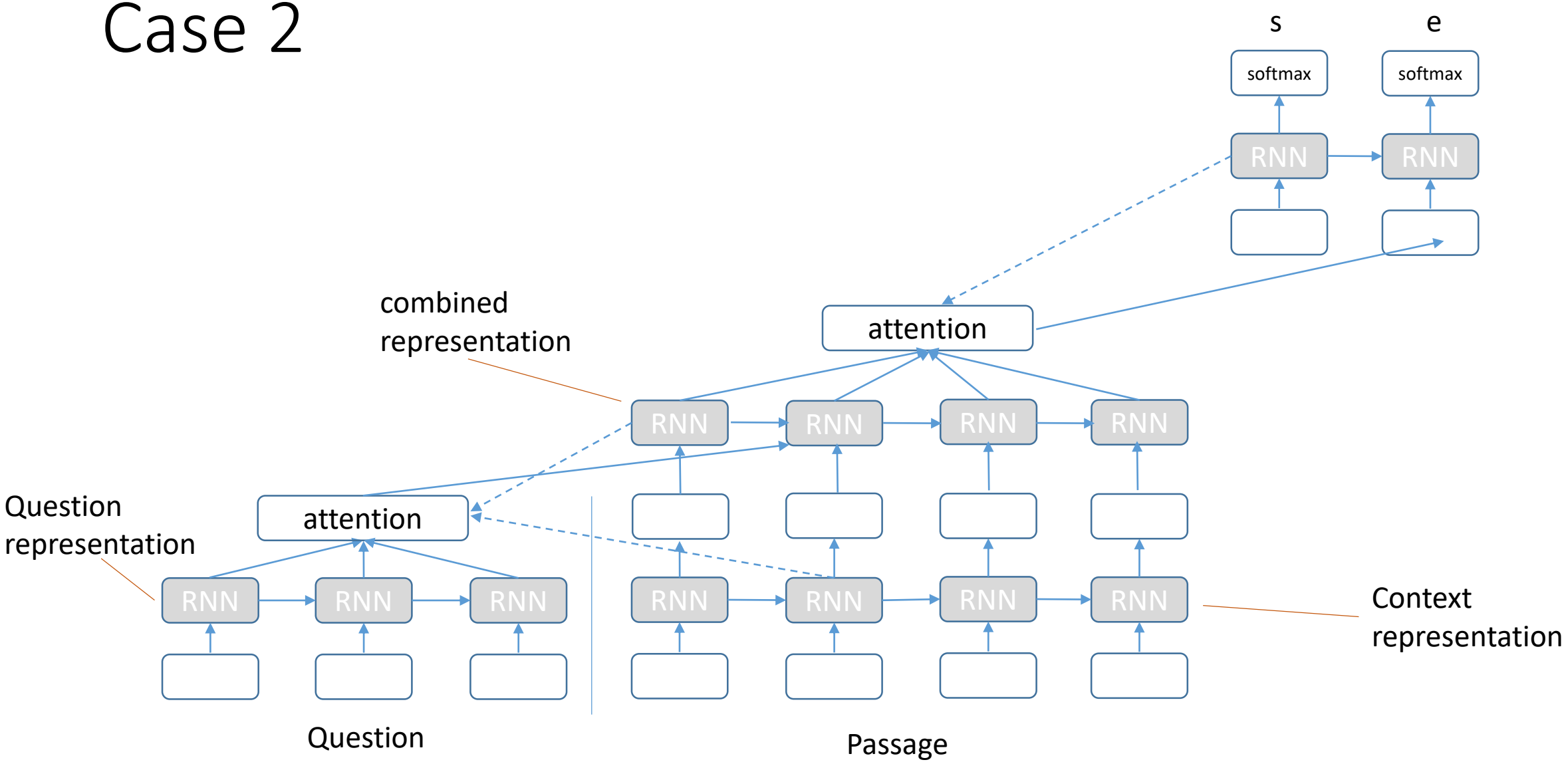
Case 2


Question representation
RNN
RNN
RNN
Question
RNN
RNN
RNN
RNN
Passage
Context representation

Case 2


combined
representation
RNN
RNN
RNN
RNN
attention
Question
representation
RNN
RNN
RNN
RNN
RNN
RNN
RNN
Context
representation
Question
Passage

Case 2


s
e
softmax
softmax
RNN
RNN
combined
representation
attention
RNN
RNN
RNN
RNN
Question
representation
attention
RNN
RNN
RNN
RNN
RNN
RNN
RNN
RNN
Context
representation
Question
Passage

Project 2

- Additional data is not allowed!
  - If you use public model pre-trained by others, make sure the model is not trained using additional (labeled) data.
  - You can use word vectors pre-trained on whole Wikipedia.

Summary

- RNN VS Feed-forward Net
- Vanilla RNN
- LSTM/GRU/Bi-directional RNN
- Conditional RNN for image caption generation
- Seq2seq (attention modelling) for machine translation
- Complex combination for question answering

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/

- [14] https://github.com/kjw0612/awesome-rnn#image-captioning
- [15] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan, Show and Tell: A Neural Image Caption Generator, arXiv:1411.4555 / CVPR 2015
- [16] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. BLEU: A method for automatic evaluation of machine translation. In ACL, 2002.
- [17] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan L. Yuille, Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN), arXiv:1412.6632 / ICLR 2015
- [18] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell, Long-term Recurrent Convolutional Networks for Visual Recognition and Description, arXiv:1411.4389 / CVPR 2015
- [19] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le, Sequence to Sequence Learning with Neural Networks, arXiv:1409.3215 / NIPS 2014
- [20] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, arXiv:1409.0473 / ICLR 2015
- [21] Karl M. Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom, Teaching Machines to Read and Comprehend, arXiv:1506.03340 / NIPS 2015
- [22] SQuAD: 100,000+ Questions for Machine Comprehension of Text. Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang. https://arxiv.org/abs/1606.05250
- [23] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Mohit Iyyer, Ishaan Gulrajani, and Richard Socher, Ask Me Anything: Dynamic Memory Networks for Natural Language Processing, arXiv:1506.07285
- [24] Question Answering Using Deep Learning. https://cs224d.stanford.edu/reports/StrohMathur.pdf
- [25] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. arXiv preprint arXiv:1608.07905, 2016
- https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks
- https://sites.google.com/site/deeplearningdialogue/references
- https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/