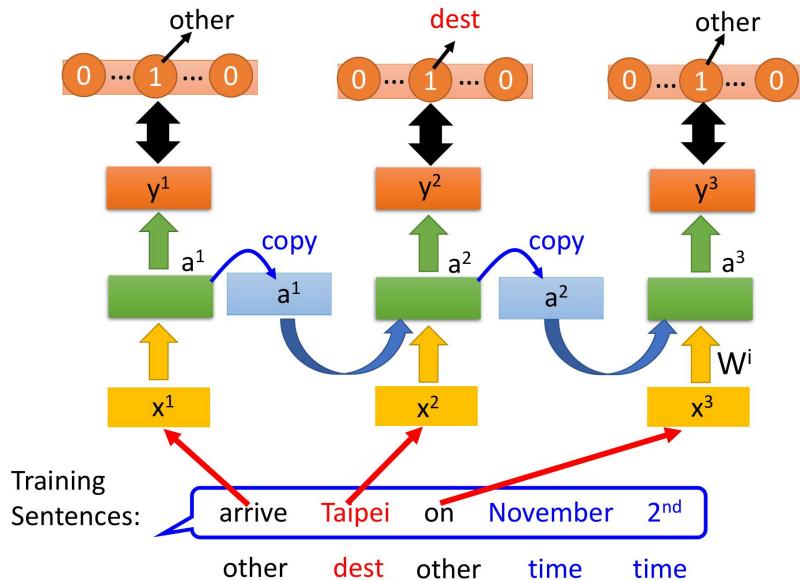


# Recurrent Neural Network

臺灣大學人工智慧中心 科技部人工智慧技術暨全幅健康照護聯合研究中心 <http://ai.ntu.edu.tw/>

## 如何學習 How to Learning

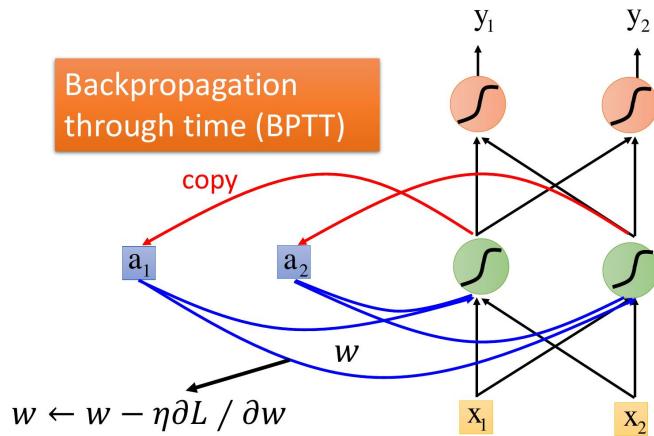
### Learning Target



- 須先定義 Cost function 以及 Loss
- Example: Slot Filling
  - 每一個output有其對應的slot
    - 第一個 word 屬於 other 這個 slot
    - 台北 屬於 destination 這個 slot
    - on 屬於 other slot
    - November 2nd 屬於抵達時間的 slot
  - 算 每一個時間點 的 RNN output 跟 reference vector 的 cross entropy
    - slot 的 cross entropy
  - 得到 loss function, 接著 gradient decent
  - Back propagation

## Back propagation Through Time

# Learning



- BPTT, Back propagation 的進階版
- 因為 RNN 是在 time sequence 上運作，所以 BPTT 需要考慮時間的information
- 不詳細說明，只需要知道 RNN 用 gradient decent train 可以 train

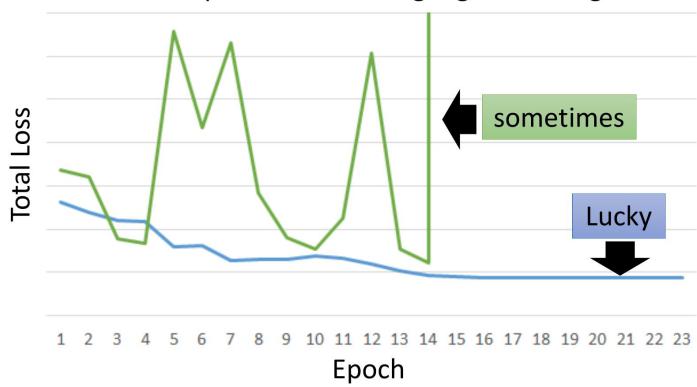
## 困難點

感謝 曾柏翔 同學  
提供實驗結果

Unfortunately .....

- RNN-based network is not always easy to learn

Real experiments on Language modeling

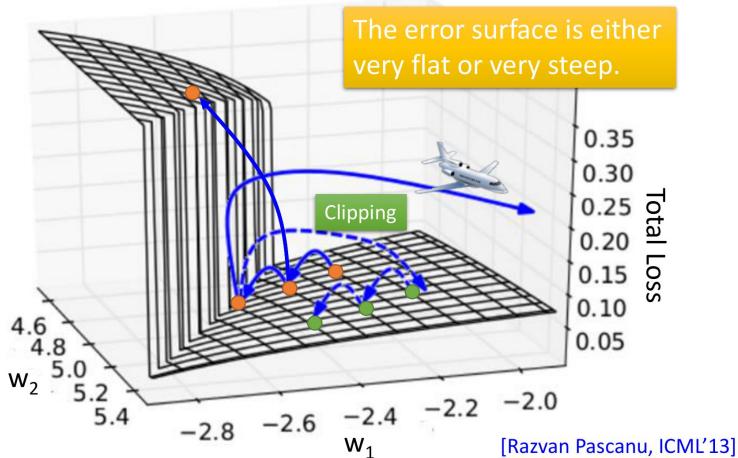


有時訓練無法像藍色一樣順利，會向綠色一樣

這個 learning curve，抖到某個地方，就突然 NaN，然後程式就 segmentation fault

## RNN 的 Error Surface

The error surface is rough.



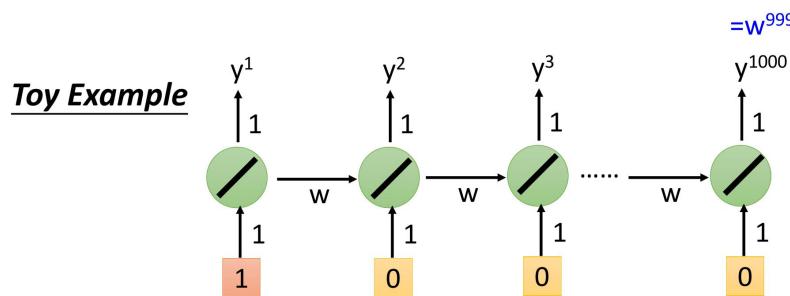
- 分析原因：RNN 的 error surface 非常的陡峭
  - error surface 就是 Total Loss 對參數的變化
  - 崎嶇的意思是：這個 error surface 有些地方非常平坦，有些地方非常陡峭
- 舉例，參考上圖：
  - 假設從橙色那個點開始，跳到下一個橙色的點
  - 可能正好就跳過一個懸崖 => Loss 會突然暴增
    - 因為之前 gradient 很小，所以 learning rate 調得比較大
    - gradient 突然很大 => 很大的 gradient 再乘上很大的 learning rate
    - 參數就 update 很多，就 NaN => 程式就 segmentation fault
- 解決辦法
  - clipping
    - 當 gradient 大於某一個threshold 時後，不要讓他超過那個 threshold
    - Ex: 當 gradient 大於 15 的時候就等於 15

---

## 為什麼 RNN 會有這種奇特的特性

## Why?

$w = 1 \rightarrow y^{1000} = 1$	$w = 1.01 \rightarrow y^{1000} \approx 20000$	Large $\partial L / \partial w$	Small Learning rate?
$w = 0.99 \rightarrow y^{1000} \approx 0$	$w = 0.01 \rightarrow y^{1000} \approx 0$	small $\partial L / \partial w$	Large Learning rate?



- 用直觀方法來知道一個 gradient 的大小
  - 把某一個參數做小小的變化，看他對 network output 的變化有多大
- 假設：
  - 一個簡單的 RNN，一個 linear neuron，weight 是 1，沒有 bias
  - 一個 input，output weight 也是 1，transition 部分的 weight 是 w
  - 如果 input 是  $[1 \ 0 \dots \ 0 \ 0]$  => 第 1000 個時間點的 output 值是 w 的 999 次方
- 觀察：
  - 假設  $w = 1$ ，network 在最後時間點的 output 也是 1
  - 假設  $w = 1.01$ ，network 在最後時間點的 output 是 20000
  - 假設  $w = 0.99$ ，network 在最後時間點的 output 是 0
- 在 1 這個地方有很大的 gradient，但在 0.99 的地方 gradient 就突然變得非常非常的小
  - 有時候需要一個很大的 learning rate
  - 設 learning rate 很麻煩，error surface 很崎嶇
- RNN training 的問題
  - 來自於 RNN 把同樣的東西，在 transition 的時候反覆使用
  - 造成 gradient vanishing 以及 gradient explode

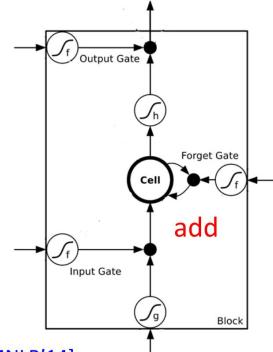
## 解決方法

# Helpful Techniques

## • Long Short-term Memory (LSTM)

- Can deal with gradient vanishing (not gradient explode)
- Memory and input are added
- The influence never disappears unless forget gate is closed
- No Gradient vanishing (If forget gate is opened.)

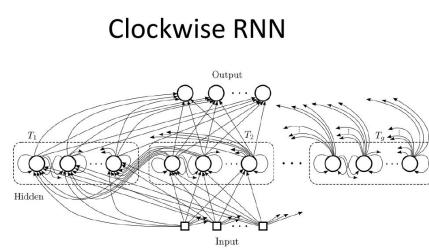
Gated Recurrent Unit (GRU):  
simpler than LSTM



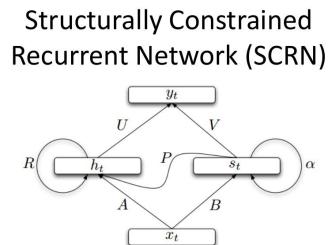
- LSTM 可以讓 error surface 不要那麼崎嶇
  - 解決 gradient vanishing
  - 不會解決 gradient explode
- 關於 LSTM 的常見問答
  - 為什麼我們把 RNN 換成 LSTM ?
    - 因為 LSTM 可以 handle gradient vanishing 的問題
    - 為什麼 LSTM 可以 handle gradient vanishing 的問題呢?
      - 因為它們在面對 memory 的時候，處理的 operation 不一樣
        - RNN 在每一個時間點 memory 裡面的資訊，都會被洗掉
        - LSTM 會透過 forget gate 決定要不要洗去memory，如果過去的memory被影響，那這個影響有高機率會留著
      - 註：早期的LSTM是為了解決 gradient vanishing 的問題，一開始沒有 forget gate，後來才加上去的
  - Gated Recurrent Unit (GRU)
    - 用 gate 操控 memory 的 cell
    - GRU 的 gate 只有 2 個
      - 參數量是比較少的，training時間少，比較robust
    - 精神：舊的不去，新的不來
      - input gate 跟 forget gate 連動起來
      - input gate 被打開的時候，forget gate 就會被自動關閉，反之亦然

## 更多處理 gradient vanishing 的 techniques

# Helpful Techniques



[Jan Koutnik, JMLR'14]



[Tomas Mikolov, ICLR'15]

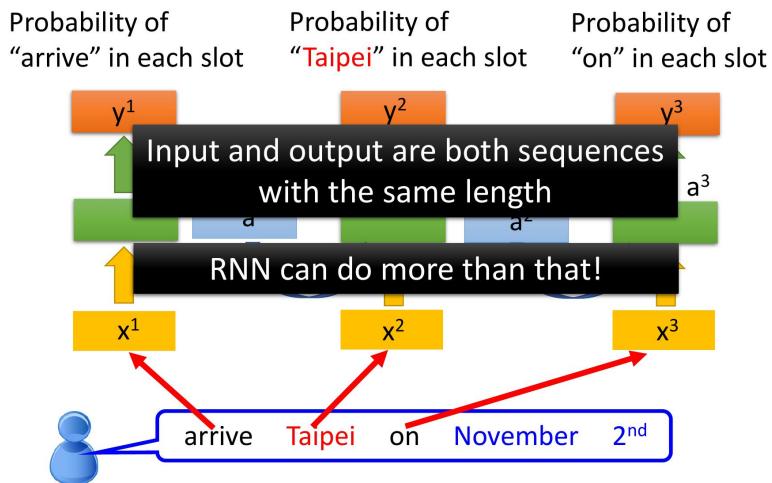
Vanilla RNN Initialized with Identity matrix + ReLU activation function [Quoc V. Le, arXiv'15]

- Outperform or be comparable with LSTM in 4 different tasks

- Clockwise RNN
- SCRN
- Vanilla(一般) RNN + identity matrix + ReLU activation function

## 更多應用

### More Applications .....

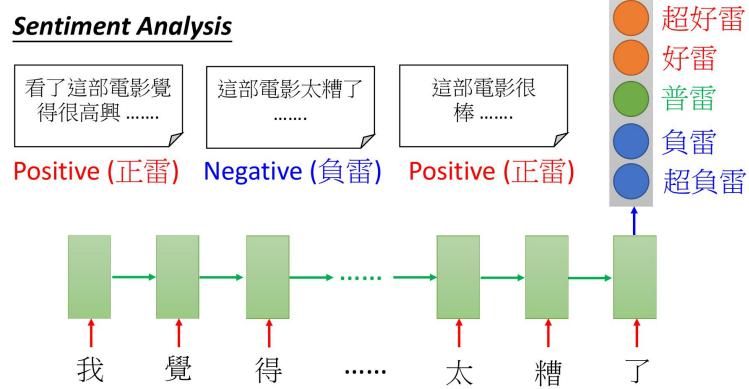


RNN 可以做到更複雜的事情

- Sentiment Analysis
  - 知道一句話是 positive 還是 negative
  - 影評分析
-

## Many to one

- Input is a vector sequence, but output is only one vector

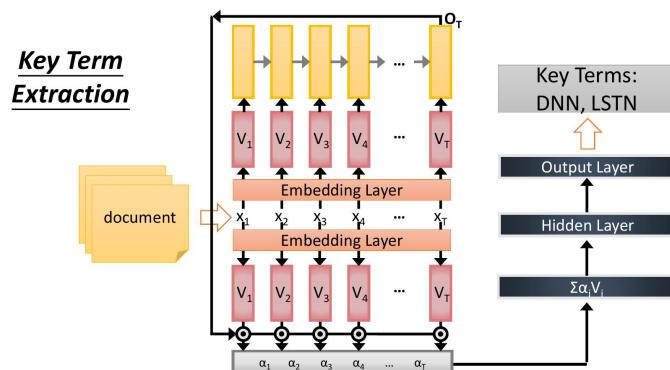


- key term extraction
  - Given 一篇文章, predict 這篇文章有那些關鍵詞彙

## Many to one

[Shen & Lee, Interspeech 16]

- Input is a vector sequence, but output is only one vector

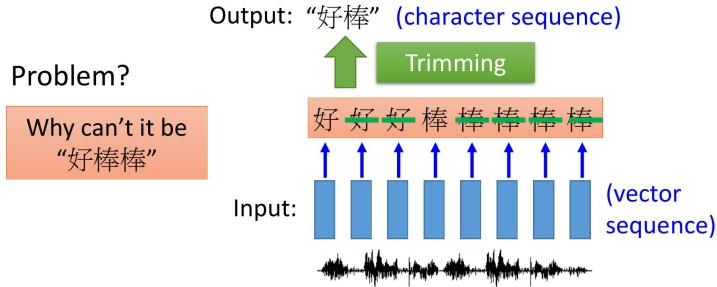


也可以是多對多的

- 語音辨識 (Speech Recognition)
  -

## Many to Many (Output is shorter)

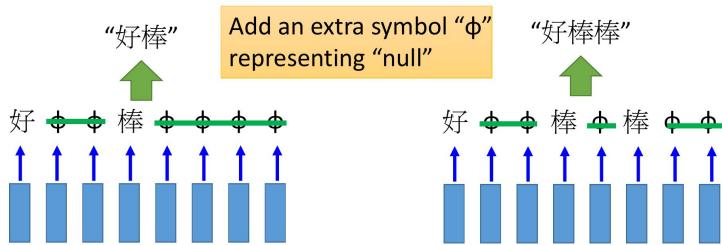
- Both input and output are both sequences, but the output is shorter.
  - E.g. Speech Recognition



- input 是一串 acoustic feature sequence (每一小段時間切一個vector, 0.01秒之類)
- output 是 character 的 sequence
- Trimming
  - 好好好棒棒棒棒棒棒 => 好棒
  - 沒有辦法辨識 好棒棒 (與 好棒 意思相反)
- CTC
- 

## Many to Many (Output is shorter)

- Both input and output are both sequences, but the output is shorter.
- Connectionist Temporal Classification (CTC) [Alex Graves, ICM'06][Alex Graves, ICM'14][Hasim Sak, Interspeech'15][Jie Li, Interspeech'15][Andrew Senior, ASRU'15]



- 在 output 的時候，不只是 output 所有中文的 character。還多 output 一個符號，叫做 Null，叫做沒有任何東西。
  - output 就會變成是 好 null null 棒 null null null null 或是 好 null null 棒 null 棒 null null，能夠區分 好棒 跟 好棒棒
  - 解決疊字的問題

## Sequence to sequence learning

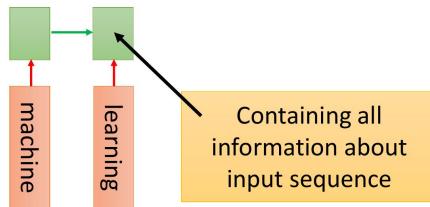
- input 和 output 都是 sequence, 但長度不一樣
  - 與CTC不同 (input長, output 短) , 不確定誰長誰短

- 實際case

- machine translation

## Many to Many (No Limitation)

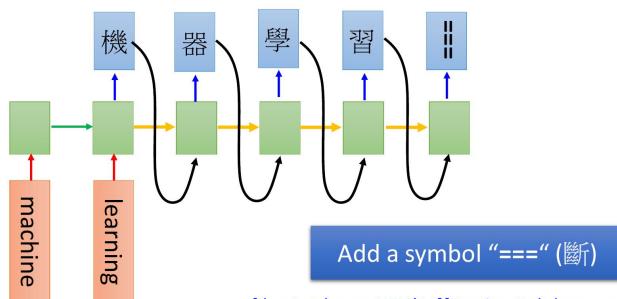
- Both input and output are both sequences with different lengths. → Sequence to sequence learning
  - E.g. Machine Translation (machine learning→機器學習)



- ■ 用 RNN 讀過去，在最後一個時間點呢，memory 存了所有 input 的整個 sequence 的 information (例如：讀進machine learning)
  - 接下來，你就讓 RNN 吐開始吐 character (從機、器、學、習、慣、性……)
- 問題：output停不下來
- 解法：加一個「斷」的token
- 

## Many to Many (No Limitation)

- Both input and output are both sequences with different lengths. → Sequence to sequence learning
  - E.g. Machine Translation (machine learning→機器學習)



[Ilya Sutskever, NIPS'14][Dzmitry Bahdanau, arXiv'15]

- 同樣的做法可以用在語音辨識
  - input: acoustic feature sequence
  - output: character sequence
  - 沒有CTC強，但是意外的工作
- 結合語音辨識跟翻譯，例如：input是一段英文訊號，output中文句子
  - 搜集訓練資料變得比較容易（不需要中間的英文句子）

## Many to Many (No Limitation)

- Both input and output are both sequences with different lengths. → Sequence to sequence learning
- E.g. Machine Translation (machine learning → 機器學習)

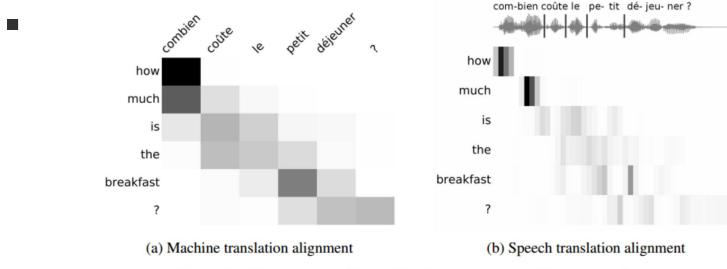


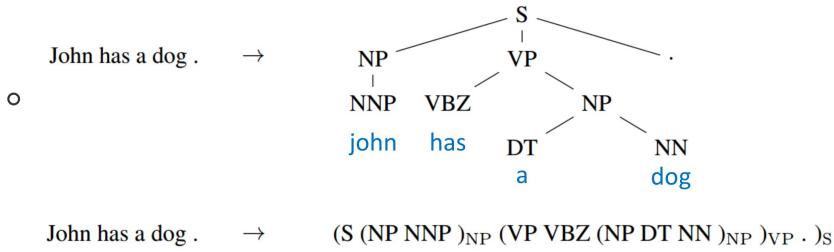
Figure 1: Alignments performed by the attention model during training

## Beyond Sequence

- Syntactic parsing tree

## Beyond Sequence

- Syntactic parsing



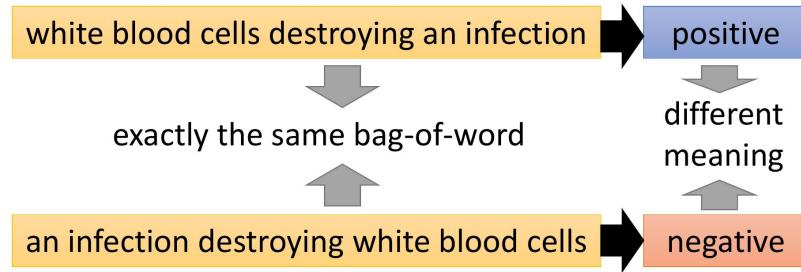
Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, Geoffrey Hinton,  
Grammar as a Foreign Language, NIPS 2015

- Input: 一個句子
- Output: 這個句子的文法的結構樹
- 讓 machine 得到這樣的樹狀的結構
  - 過去: structure learning
  - 現在: 把這個樹狀圖, 描述成一個 sequence
    - root 的地方是 S

## Sequence to sequence auto-encoder

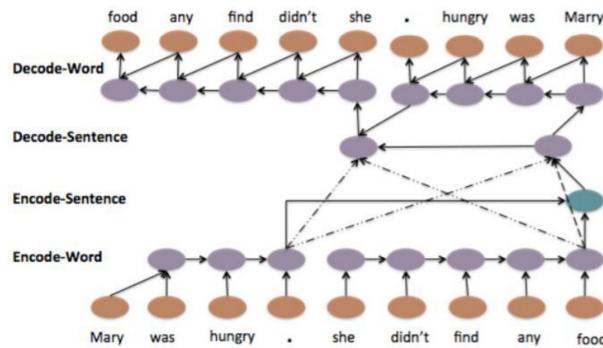
## Sequence-to-sequence Auto-encoder - Text

- To understand the meaning of a word sequence, the order of the words can not be ignored.



- Document to vector
  - bag-of-word
    - 忽略 word order 的 information
    - 字的排列可能造成句子正命或是負面的影響
  - Sequence to sequence auto-encoder
    - input 一個 word sequence
    - 通過 RNN 把它變成一個 embedded 的 vector (潛藏重要資訊)
    - 把這個 embedded vector 當成 decoder 的輸入，讓這個 decoder 長回一個一模一樣的句子
    - 好處：
      - 不需要 label data，只需要收集到大量的文章
  - Hierarchical neural auto-encoder

## Sequence-to-sequence Auto-encoder - Text



Li, Jiwei, Minh-Thang Luong, and Dan Jurafsky. "A hierarchical neural autoencoder for paragraphs and documents." arXiv preprint arXiv:1506.01057(2015).

- output target 會是下一個句子
  - 比較好得到 語意 的意思

那如果我們要把一個 document 表示成一個 vector 的話

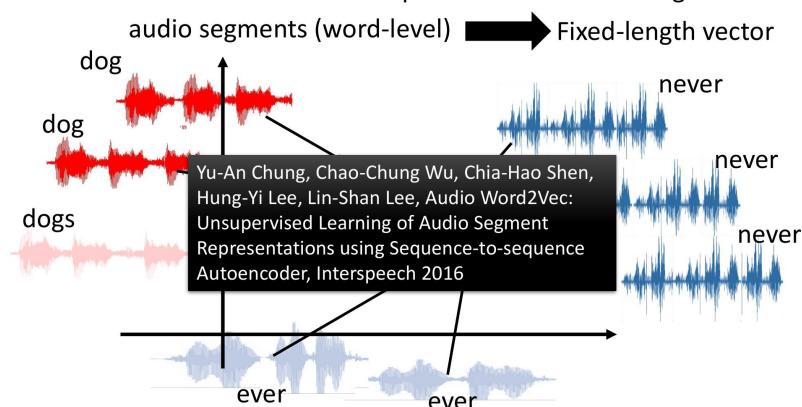
往往會用 bag-of-word 的方法

## 語音上的應用

- 

### Sequence-to-sequence Auto-encoder - Speech

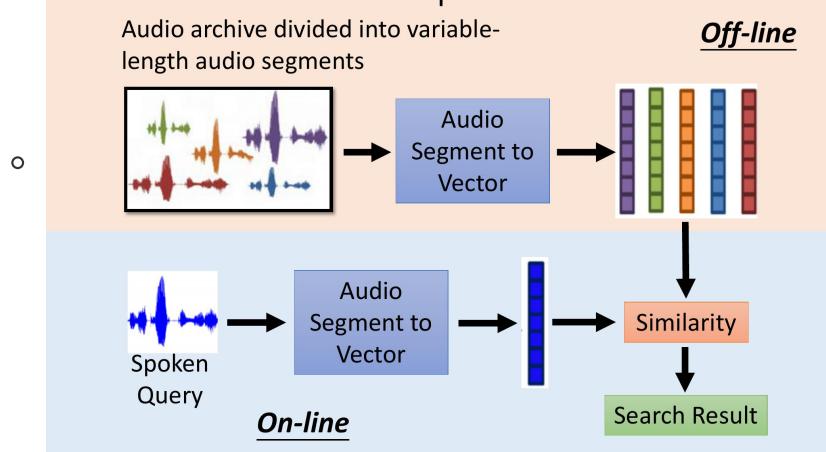
- Dimension reduction for a sequence with variable length



- audio 的 word to vector

- 應用：語音搜尋

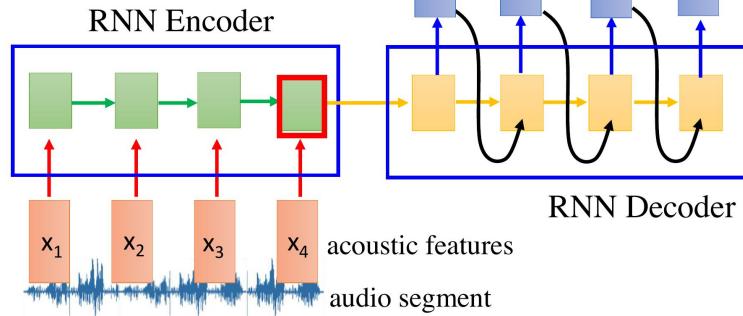
### Sequence-to-sequence Auto-encoder - Speech



- 使用者輸入一段話（語音），轉成vector後可以跟database裡的資料對比相似度。
- 先把audio segment 抽成 acoustic feature sequence當作input，丟入RNN (encoder)
  -

## Sequence-to-sequence Auto-encoder

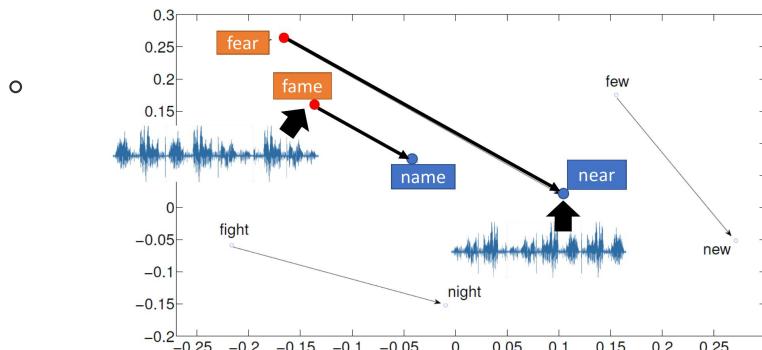
The RNN encoder and  
decoder are jointly trained.



- 然後再透過另外一個RNN decode, output要跟input越像越好
- 有趣的結果：

## Sequence-to-sequence Auto-encoder - Speech

- Visualizing embedding vectors of the words

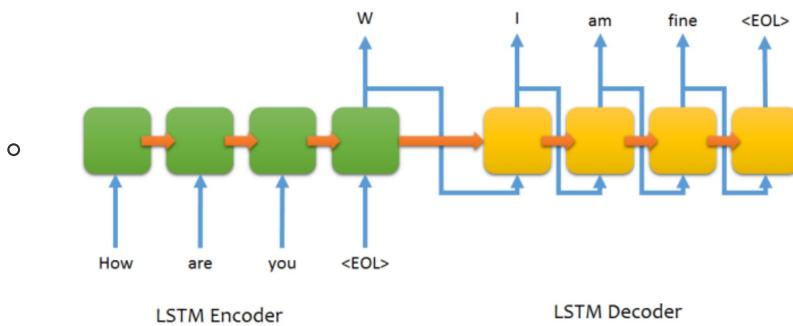


- 各個字詞具有「聲音」上的意義，但沒有「語意」上的意義（把f換成n的，向量變化方向差不多）

## 實際應用Demo

- Chatbot (聊天機器人)
  - Sequence to sequence auto encoder

## Demo: Chat-bot

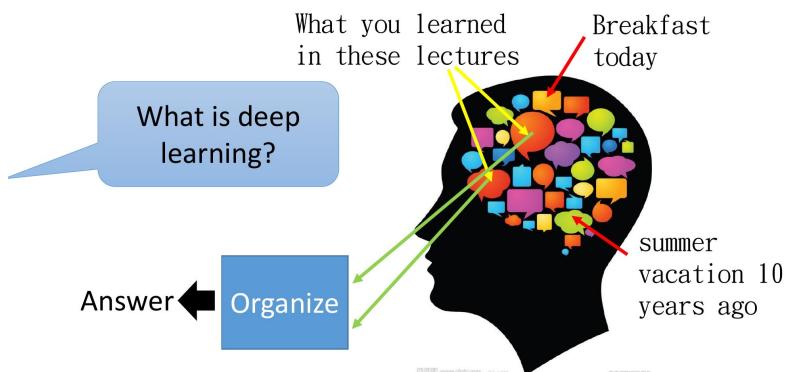


電視影集 (~40,000 sentences)、美國總統大選辯論

- Attention-base model

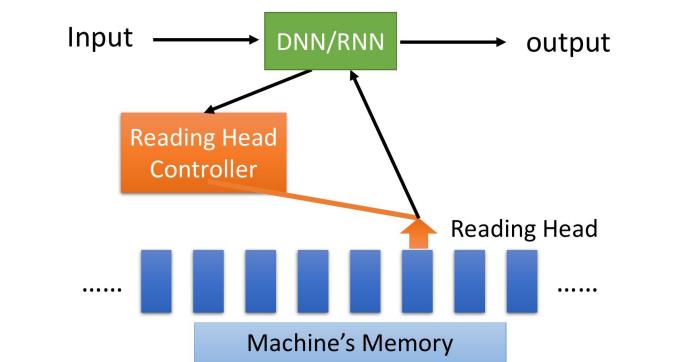
◦

### Attention-based Model



[http://henrylo1605.blogspot.tw/2015/05/blog-post\\_56.html](http://henrylo1605.blogspot.tw/2015/05/blog-post_56.html)

### Attention-based Model



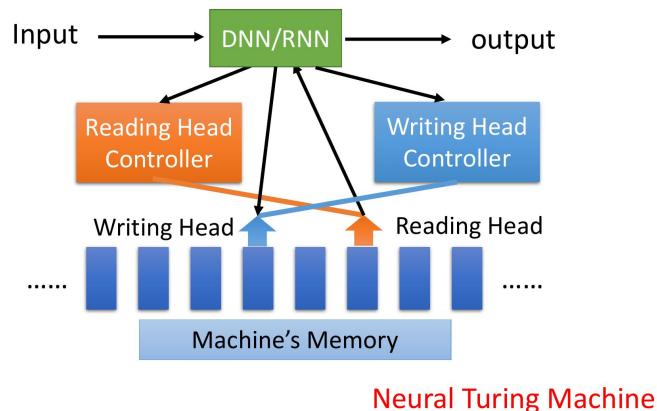
Ref:  
[http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS\\_2015\\_2/Lecture/Attain%20\(v3\).ecm.mp4/index.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Attain%20(v3).ecm.mp4/index.html)

- - 當你輸入一個 input 的時候，這個 input 會被丟進一個中央處理器
  - 這個中央處理器，可能是一個 DNN/RNN，用來操控一個讀寫頭 (reading head)

controller)

- 這個 reading head controller 會決定這個 reading head 放的位置，然後 machine 再從這個 reading head 放的位置，去讀取 information 出來，產生最後的 output
- Neural Turing Machine (2.0 版本)

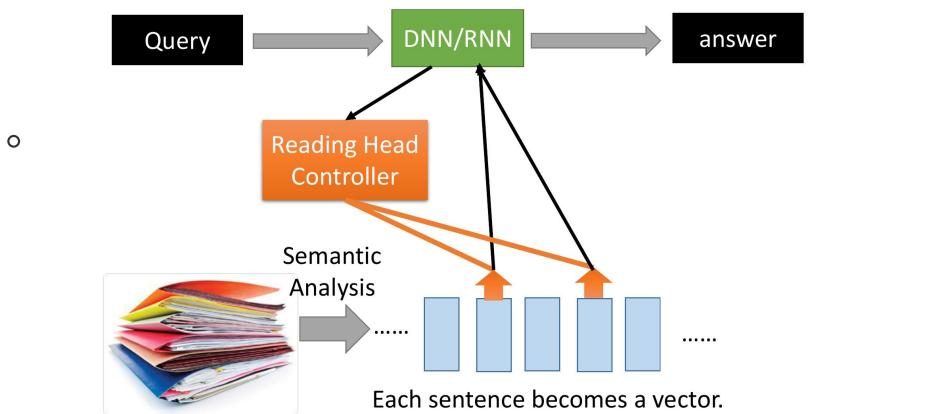
## Attention-based Model v2



Neural Turing Machine

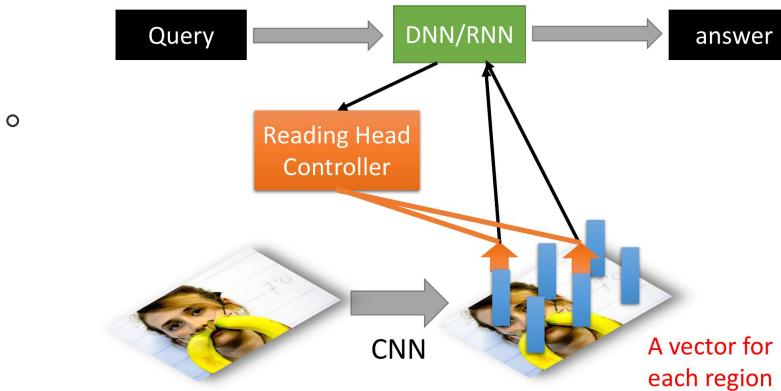
- ■ 會多去操控一個 writing head controller
- 不只有讀的功能，還可以把資訊 discover 出來的東西，寫到它的 memory 裡面去
- Reading comprehension

## Reading Comprehension

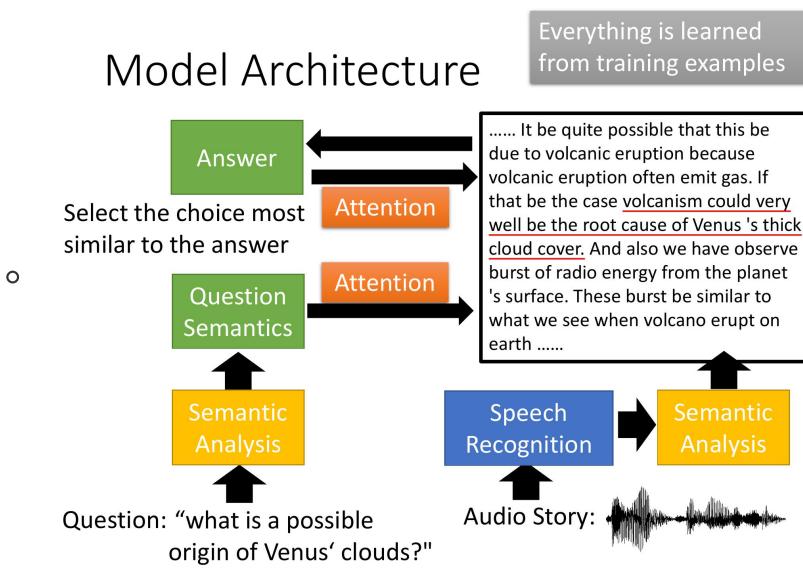


- attention-based model
- 先讓model看一堆document，再問問題，透過attention model找出答案
- Visual Question Answering

# Visual Question Answering



- 讓 machine 看一張圖，然後問它一個問題
- 透過 CNN 可以把這個圖的每一小塊 region 用一個 vector 來表示
- Speech Question Answering
  - 讓 machine 聽一段聲音，然後問他問題，讓他從四個選項裡面選出正確選項 (TOFEL)
  - 同樣使用 attention-based model



## RNN v.s. Structured learning

# RNN v.s. Structured Learning

- RNN, LSTM
  - Unidirectional RNN does NOT consider the whole sequence
  - Cost and error not always related
  - Deep



- HMM, CRF, Structured Perceptron/SVM
  - Using Viterbi, so consider the whole sequence
  - How about Bidirectional RNN?
  - Can explicitly consider the label dependency
  - Cost is the upper bound of error

我這邊其實有一個問題

我們講了 Deep learning  
也講了 Structured learning

- 不同之處：
  - 考慮整個句子：
    - uni-directional 的 RNN 或 LSTM 只看一半
    - 透過 Viterbi 的 algorithm, 可以考慮的是整個句子
    - 但 RNN/LSTM 等等, 也可以做 Bi-directional
  - 能否直接限制 label 和 label 之間的關係
    - 用 Viterbi algorithm 求解的時候, 可以直接把你要的 constrain 下到 Viterbi algorithm 裡面
  - RNN 和 LSTM 的 cost function 跟實際上最後要考慮的 error 往往是沒有關係的
    - 如果是用 structured learning 的話, 它的 cost 會是你 error 的 upper bound
  - Deep
    - RNN/LSTM 可以是 deep
    - HMM, CRF 拿來做 deep learning 是比較困難的
  - 整體說起來 RNN/LSTM 在 sequence labeling task 上面表現是比較好的

## Deep learning 和 structured learning 的結合

- 舉例：
  - input 的 feature 先通過 RNN/LSTM
  - 通過 RNN/LSTM 的 output 再做為 HMM, CRF... 的 input
  - 用 RNN/LSTM 的 output 來定義 HMM, CRF... 的 evaluation function
- 同時又享有 deep 的好處, 又享有 structured learning 的好處
- 語音上常見的組合

- deep learning 的 model: CNN/LSTM/DNN , 加上 HMM
- HMM 往往都還在
- Slot filling
  - 很流行用 Bi-directional LSTM, 再加上 CRF 或是 structured SVM
  - 先用 Bi-directional LSTM 抽出 feature, 再拿這些 feature 來定義 CRF 或者是 structured SVM 裡面需要用到的 feature
- 也許 deep and structured 是未來一個研究的重點的方向