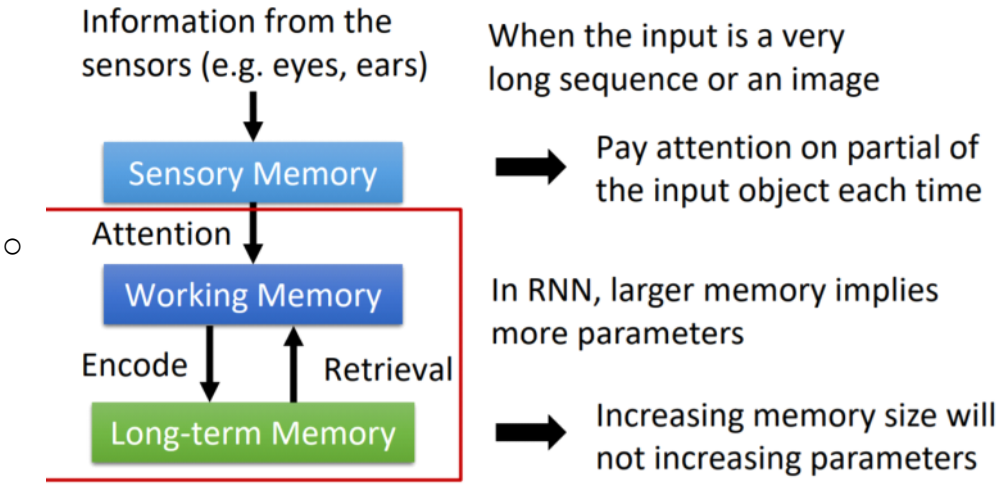
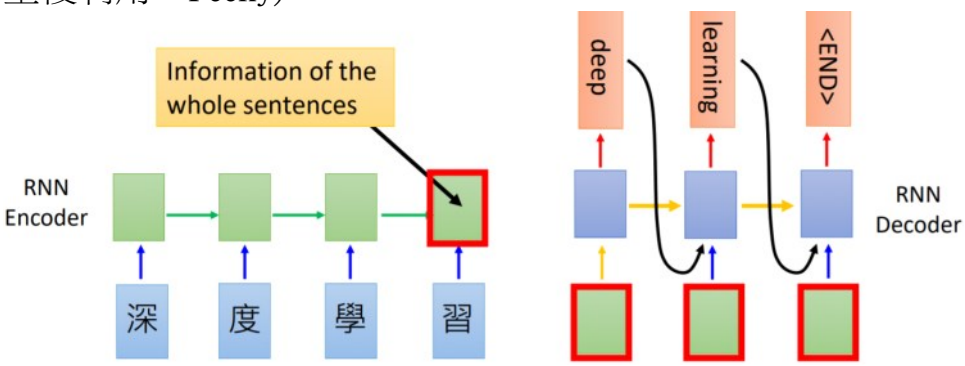
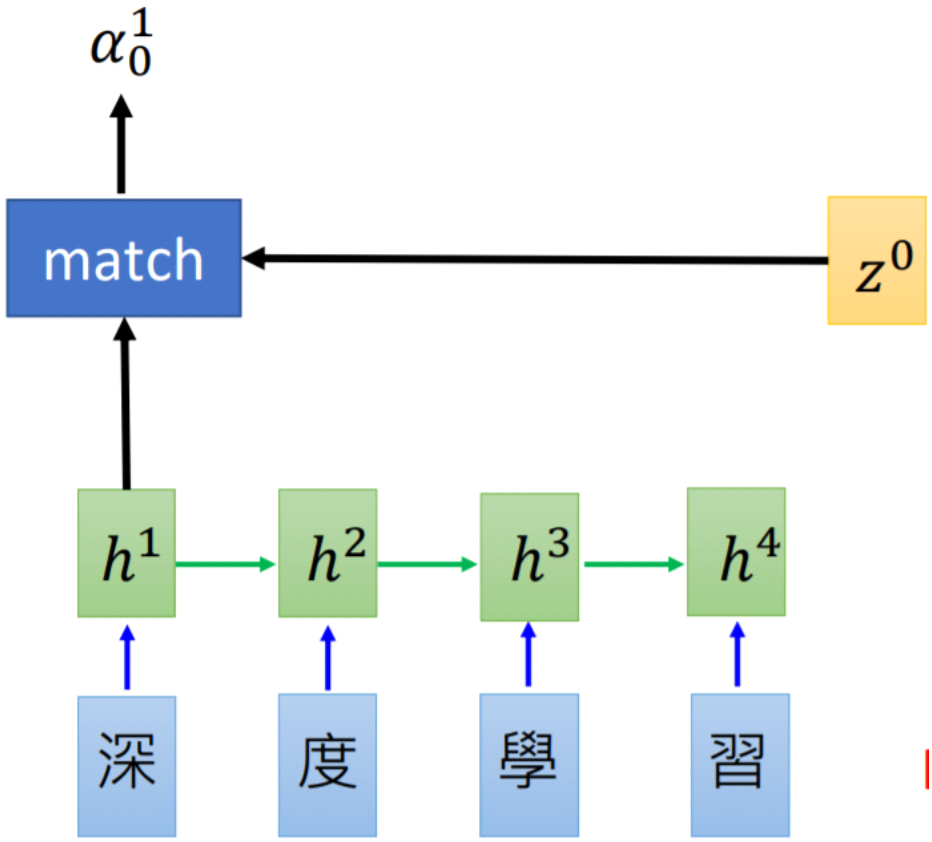
ADL—Attention

2019年3月19日　　上午 10:36

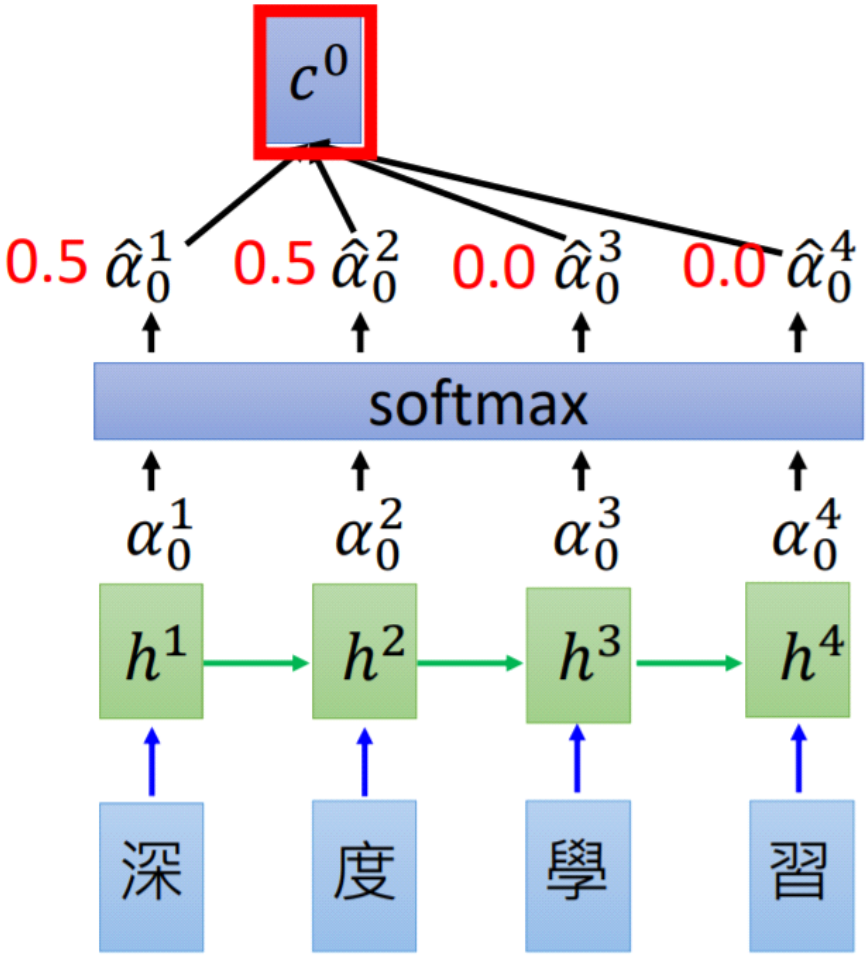
- 只關注有需要拿進去處理的資訊
  - EX: ignore雜訊，只會關注想要的working訊號，只有部分資訊拿進去處理
  - long-term memory是已經存進去了。經過encode後就會放進去腦袋，需要的時候再拿進去拿出來，把所有資訊濃縮
  - Attention: input很大很長，不會一次把全部資訊都消化掉，只會從input object的部分抓出資訊
  - 想要保留越多的資訊，所需要的參數就越多。但如果先儲存資訊到working memory，要的時候才拿。RNN是存下來
  - 

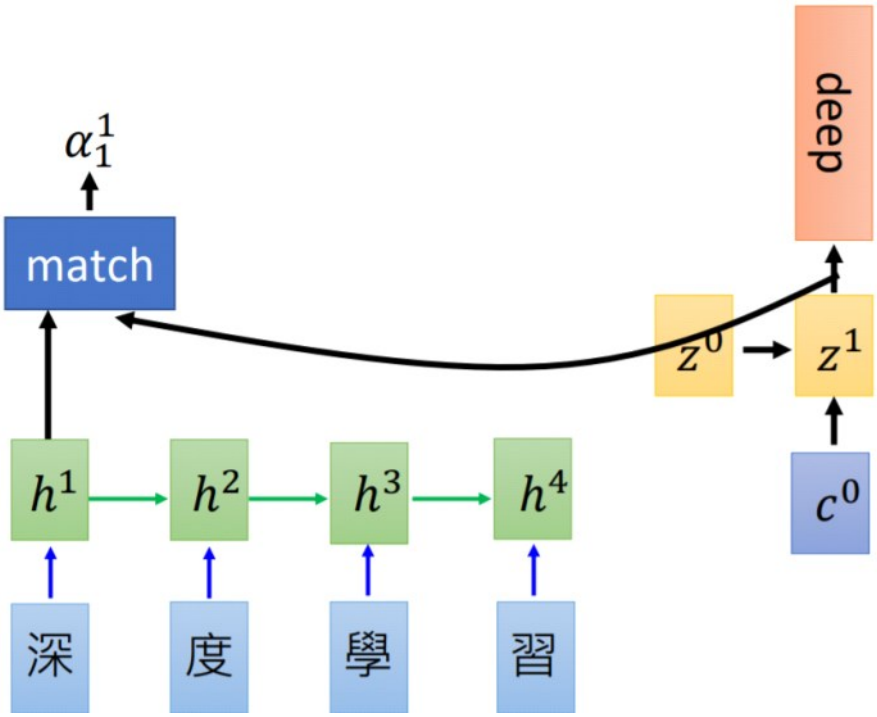
- 產生的字會影響下一個字的產生=>auto regressive
- 希望把原本產生出來的vector也可以被encode重新考量起來 (紅色框框重複利用，Peeky)
- 

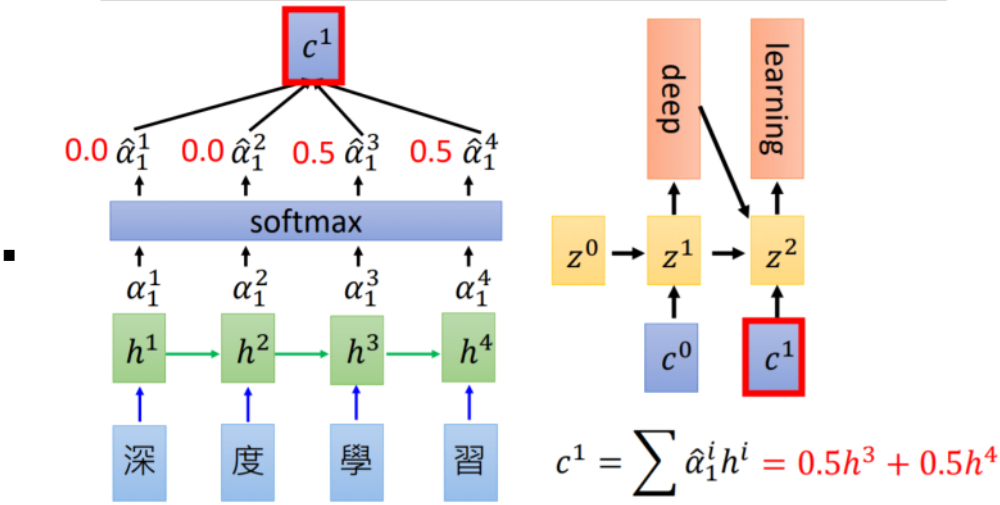
- Attention:
  - 選擇要注意的地方z，可以說要focus在哪裡=>alpha (EX:0~1之間的小數)
  - match可以看z和h的cosine similarity，如果高的話就高
    - 放NN，會輸出一個scalar輸出weight，同時一起訓練(提高模型參數)
      - input : h跟z輸出alpha

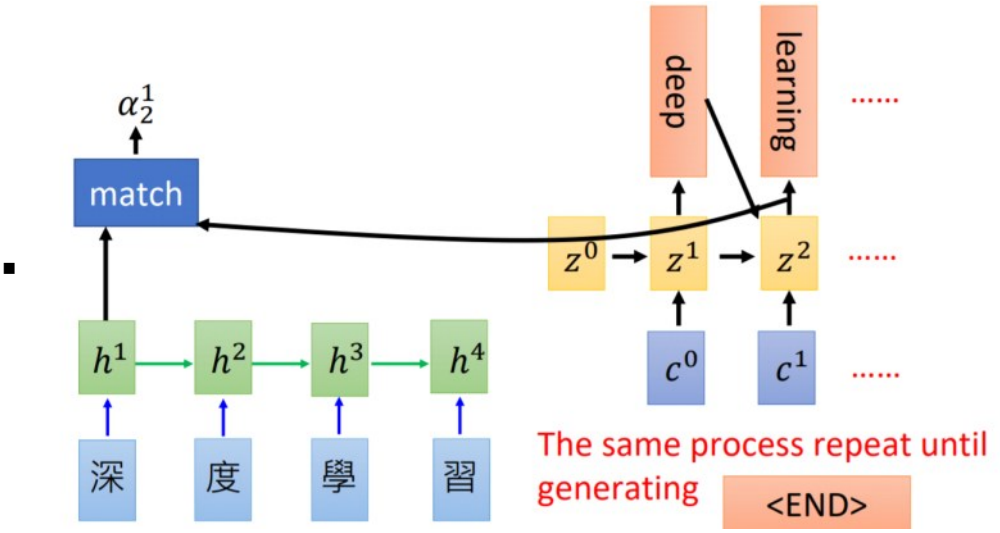


- z0對應過不同h會輸出不同alpha再經過softmax成為機率分布
  - 對於h1~h4應該要放多少注意力在上面，算出weigted sum得到c0
  - 原本是只看h4(重複看)，但怕h1h2的資訊被削弱了，如果覺得"深度"比較重要，c0就會變成h1跟h2的weighted sum結果
  - 把c0當成RNN的input

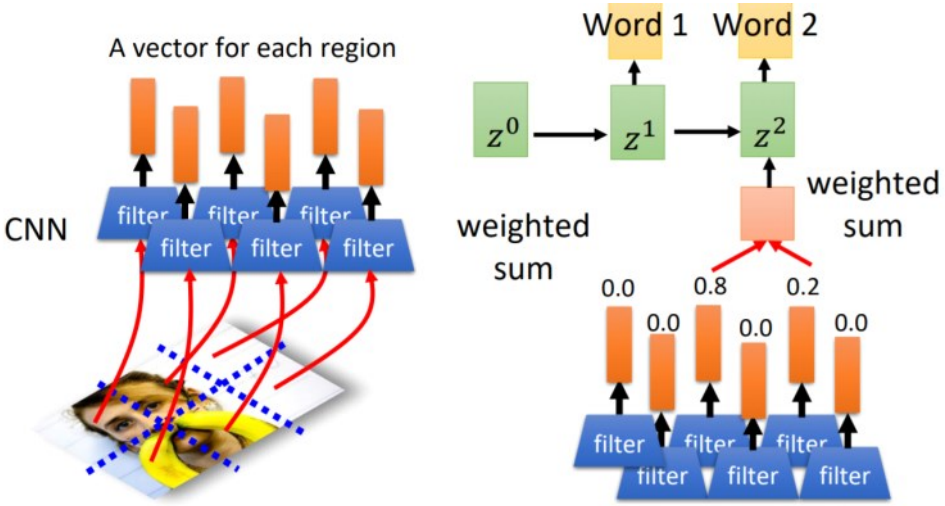
- 

- z1是輸出的hidden state，接下來再把"deep"拿進去看要focus在哪裡
- 得到c1的attention結果再去predict
- 

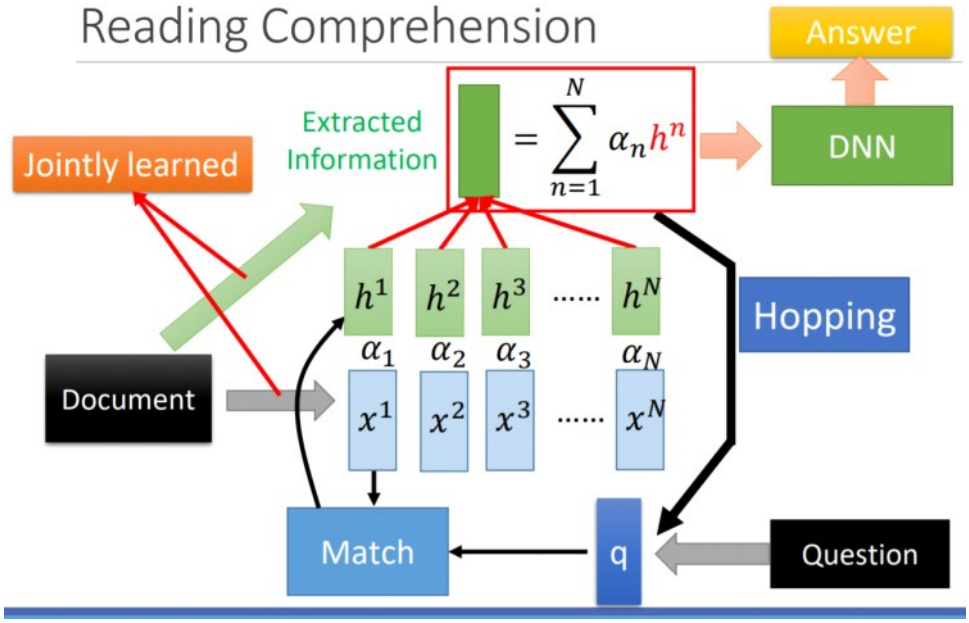
- 

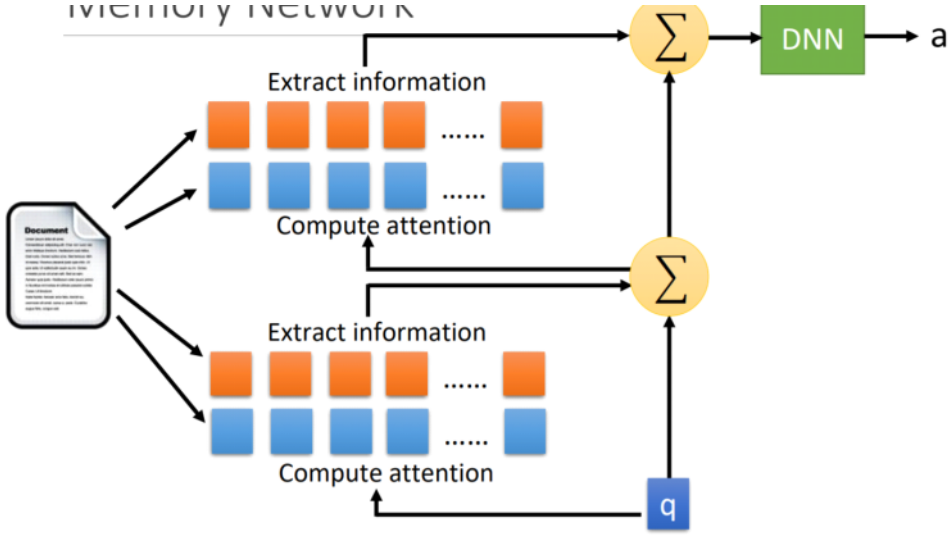
  - 不斷輸出值到end signal
- 

  - 並不會增加額外參數，頂多match機制，但如果是使用cos-similarity就會不需要了

- Image Captioning
  - 把圖片切成多個小region，利用attention跟小region做weighted sum，看哪一個小塊比較重要，當成RNN要decoder的輸入，得到第一個字再去做下一個
  - 

  - 藉由attention機制，去看機器學了什麼，但是performance不一定會進步。可以知道為什麼會有這樣的錯誤
  - 不同的字focus的點不一樣 (video captioning也可以做)

- Reading Comprehension
  - 有一篇文章，希望回答相關問題
  - 只要選擇部分才可以回答問題
  - 把N個句子分開，encode成N個向量
  - 問題也是個向量
  - 所以對N個句子看attention答案，去看答案在哪一個句子
  - 把對應的句子進行weigted sum
  - 參數: encdoe句子們、attention、問題向量怎麼得到答案
  - 可以jointly的train
  - 但是summarize出來的資訊跟原本句子不一定會一模一樣，所以要更好的encode
  - Match: x跟h不一樣，用q跟x決定h再去算attention
  - 但是可能橫跨好幾個地方，所以attention要做好多次，得到的結果可能還需要去看一次attention的分布，要做好幾層才可以得到真實答案
  - 

- Memory network
  - questuion近來會去計算attention，才去算出結果
  - 然後再把抽出來的資訊做一次attention，拿不同的機率分布
  - 

    - 第一次attention是從問題來的，第二次attention是從那句supporting fact在去找下一個部分重要的句子出來
    - 把句子都存成memory
- Conversational QA
    - dataset; CoQA、QuAC
    - 要可慮前面問題的答案才可以知道下一題怎麼辦

Attention on Memory

- Neural Turing Machine
    - 模擬真實電腦得讀取狀況，把資訊ecode到memory裡面