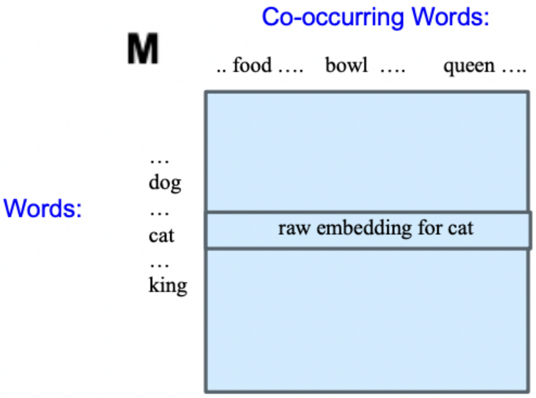
CAS CS 505
Lec 14

Static Embeddings Concluded; Machine Translation, The Attention Mechanism

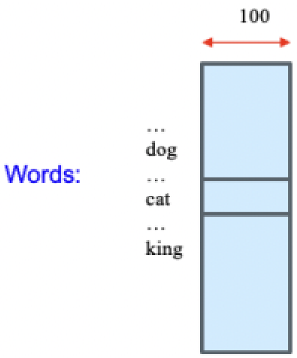
1. Word Embeddings: GloVe
    - a. Another popular word embedding method is GloVe (Global Vectors):
    - b. The first step in creating GloVe embeddings is to create a co-occurrence matrix M
    - c. Each row in this matrix is the raw embedding for a word (they are symmetric):
    - d.



        - i. The original model was trained on 400,000 words from
            1. 2010 Wikipedia dump (1 billion tokens)
            2. 2014 Wikipedia dump + Gigaword 5 (6 billion tokens)
            3. Common Crawl (42 billion tokens)
        - ii. The context window was 21 words (10 to left and 10 to right)
    - e. Next, they adjust the frequency counts to de-emphasize very frequent words and very rare words.
        - i. Very frequent words and very rare words are not highly important
        - ii. Add weighting for how often it occurs in the corpus
    - f. The authors mention in the original paper that they used Adagrad with a learning rate of 0.05 with 100 epochs.
    - g. The third step is to use gradient descent to adjust the frequency counts further so that for two words w1 and w2,

$$\text{embed}(w_1) \cdot \text{embed}(w_2) \approx \mathit{log}(\ \mathit{Prob}(\ M[w_1][w_2]\ )\ )$$

    - h. (Very similar to the gradient descent in Word2Vec). They use a least-squares cost function (same as for linear regression).
    - i. Finally, they use a dimensionality-reduction algorithm (similar to PCA) to reduce the number of columns to 100, 200, 300, etc.



j.
   i. Take co-occurrence matrix and put it into PCA

2. Word Embeddings: fastText
   a. A third popular word embedding algorithm is Meta's fastText:
      i. Uses skip-gram and CBOW approaches, like Word2Vec
      ii. Has pretrained models for 294 languages.
      iii. Unique feature of fastText is that it uses character-level information using N-grams!
   b. You can change many of the parameters, so for example, you can set N = 3 and the word matter becomes:

      `<ma`, `mat`, `att`, `tte`, `ter`, `er>` → used n-grams
   c. Works better for multiple languages → not as useful but it is an alternative
3. Word Embeddings
   a. All of these approaches provide pre-trained models, and also allow you to train on your own corpus.
   b. Which approach is better? As usual, "it depends on the task"!
   c. Word2Vec
      i. Pros: Two simple training algorithms
      ii. Cons: No way of dealing with unknown words, smaller context windows, storage intensive
   d. GloVe (can get a pretrained model and train by adding additional information)
      i. Pro: Scalable, faster to train than Word2Vec (but see next)
      ii. Con: Large preprocessing cost, storage intensive
   e. fastText:
      i. Pros: Because of character-level algorithm, better with unknown words and misspelled words
      ii. Cons: parameters require careful tuning
   f. The next question is: How do we extend this to sequences of words, i.e., how do we create sentence, paragraph, and document embeddings?

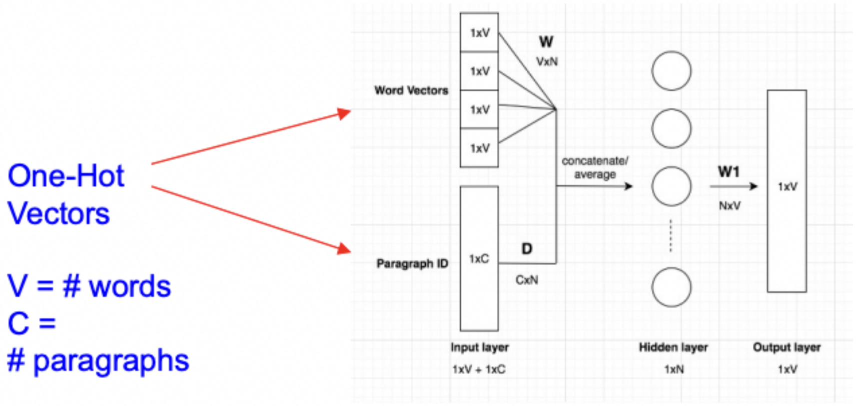
4. Sentence and Document Embeddings
    a. We used the technique of Average Word Embeddings:

$$\text{Embed}(\ w_1, w_2, \ldots, w_N\ ) \ = \ \frac{\text{Embed}(\ w_1\ ) + \text{Embed}(\ w_2\ ) + \cdots \text{Embed}(\ w_n\ )}{N}$$

    b. Many variations of this simple idea have been proposed, including using a weighted average

$$\text{Embed}(\ w_1, w_2, \ldots, w_N\ ) \ = \ \lambda_1 * \text{Embed}(\ w_1\ ) + \lambda_2 * \text{Embed}(\ w_2\ ) + \cdots \lambda_N * \text{Embed}(\ w_n\ )$$

    where the lambda are determined by the relative importance of the word (say, nouns and verbs being more important than conjunctions, etc.)
    c. Another technique is called Doc2Vec (Paragraph Vectors), which extends the basic idea of Word2Vec to paragraphs:
    d. The basic idea is to run Word2Vec but add another vector, called a Paragraph ID. This vector is part of what the model tries to predict during training, connecting it with the words within the document.



    e.
        i. Divide predicting word into paragraphs
        ii. Second paragraph is one hot → All 0s except one 1 (which paragraph it came from)
        iii. Simply telling which paragraph the word embedding came from
    f. The model is trained on a corpus of paragraphs, with the Paragraph ID shared between training samples from the same paragraph. The embedding extracted from the hidden layer now contains the meaning of words in one paragraph, in some sense storing the topics discussed in that paragraph.
    g. To this point, we have seen a variety of ways to encode sequences of words into a vector space:
        i. BOWs / TF
        ii. TF-IDF
        iii. TF(-IDF) plus PCA for dimensionality reduction
        iv. Average Word Embeddings based on Word2Vec, GloVe, fastText
        v. Doc2Vec (Paragraph Vectors)

h. We can also tweak these by

i. Tokenizing;

ii. Eliminating stop words;

iii. Using only the K most common words;

i. But all of these have a huge disadvantage: They ignore word order!

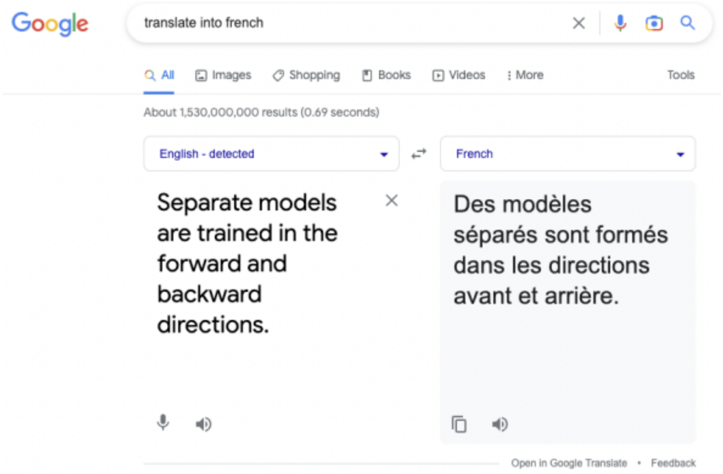
i. I hate cats and love dogs! = I hate dogs and love cats!

j. The solution to this is the technique of Attention, which will be used in the SOTA Transformer Algorithm. But in order to motivate this carefully, we shall consider one of the classic hard problems in NLP, machine translation….

5. The Attention Mechanism

a. The Attention mechanism is a way of encoding relationships between words in sequence.

b. To see the motivation for the Attention Mechanism, consider translating “Separate models are trained in the forward and backward directions.” into various languages....



translate into french

About 1,530,000,000 results (0.69 seconds)

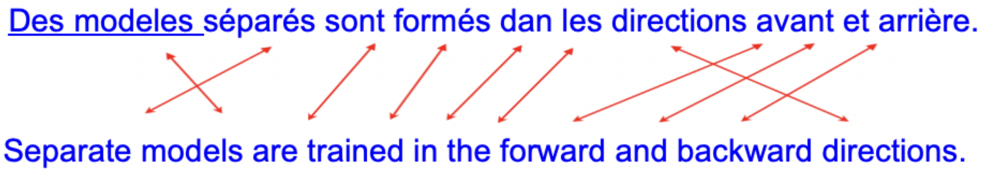
English - detected ⇄ French

Separate models are trained in the forward and backward directions.

Des modèles séparés sont formés dans les directions avant et arrière.

c. Notice that, as with most modern languages, the words have a similar sequential order, but there is some variation in position:

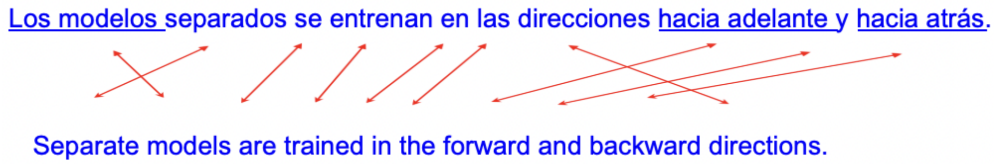
d. French:



Des modeles séparés sont formés dan les directions avant et arrière.

Separate models are trained in the forward and backward directions.

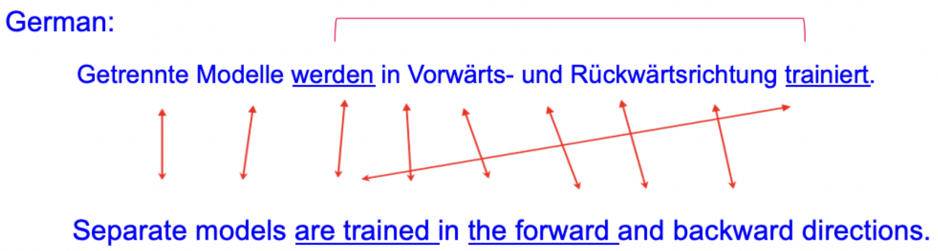
e. Spanish:



Los modelos separados se entrenan en las direcciones hacia adelante y hacia atrás.

Separate models are trained in the forward and backward directions.

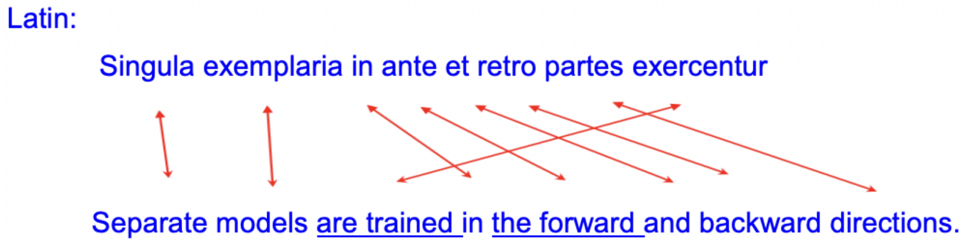
f. Some languages have a longer-range but predictable change in order (e.g., verbs go to the end of the sentence).

g. 

German:

Getrennte Modelle werden in Vorwärts- und Rückwärtsrichtung trainiert.

Separate models are trained in the forward and backward directions.

h. 

Latin:

Singula exemplaria in ante et retro partes exercentur

Separate models are trained in the forward and backward directions.

6. The Attention Mechanism: BRNNs

a. A BRNN can help with this problem, because when it creates the activation vector at one point in the sequence, it has access to the backward and forward context:

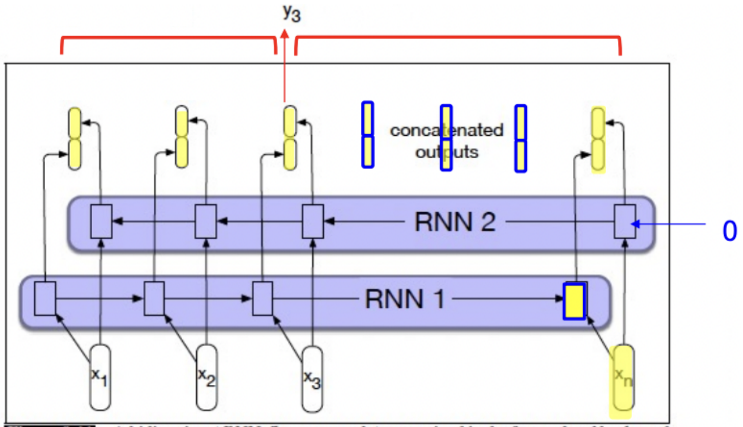
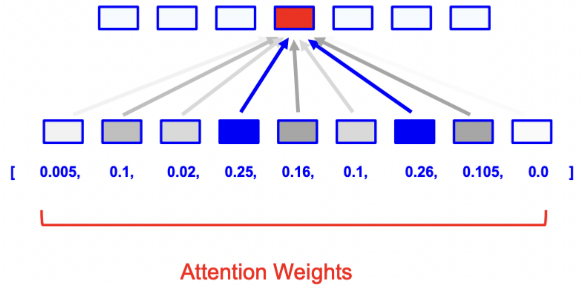
b. 

Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

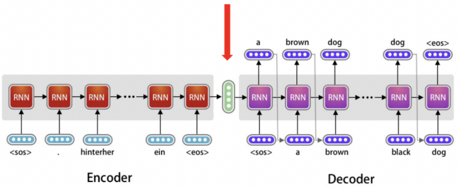
c. However, the ability to remember context ”fades” the farther you are from the current activation, and it would be useful to have more control of specific words in the forward and backward context.

7. The Attention Mechanism
    a. Attention refers the ability to focus on particular words in the backward and forward context; the pattern of what words matter in which context can be learned by the network. The pattern can be represented by a probability distribution over the sequence of input tokens:

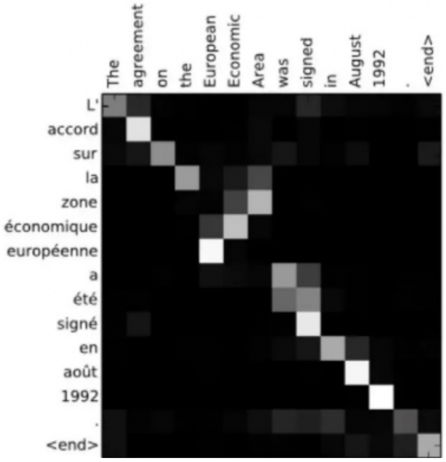


    b.

8. The Attention Mechanism: BRNNs
    a. The fundamental problem to be solved is how to increase the communication between the Encoder and Decoder, to improve the ability of the context vector to hold all the information necessary to do the translation:



        i. The connection between the encoder and decoder carries a lot of information → it is a bottleneck
        ii. Make the transition more subtle and allow more action between the two in a controlled way
    b. The basic idea of attention is to train a FFNN to produce these attention weights for each output token; the resulting 2D matrix of attention weights shows the relationship between the words in the input sentence and those in the output sentence.



c. We add a FFNN between the encoder and the decoder to calculate the attention vector a<i> for each output token.

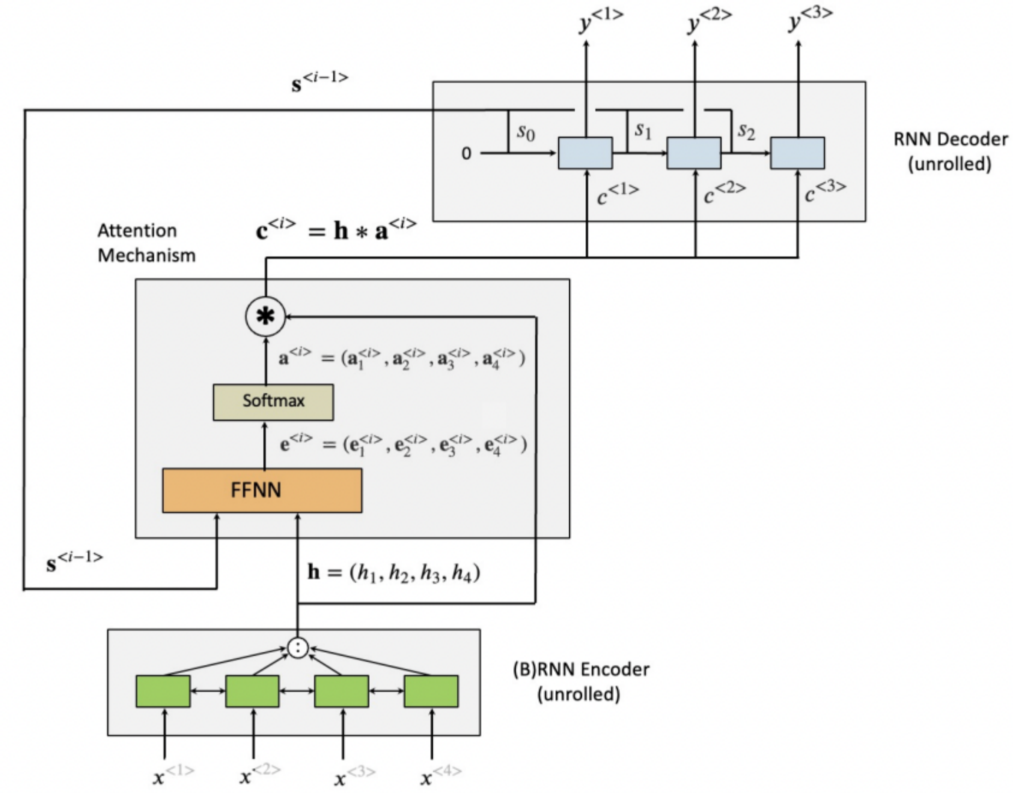
d. Input to FFNN is output vector h from Encoder and activation vector s<i-1> from Decoder.

e. Attention weight vector a<i> is softmax of output e<i> from FFNN. (learns the attention weight)

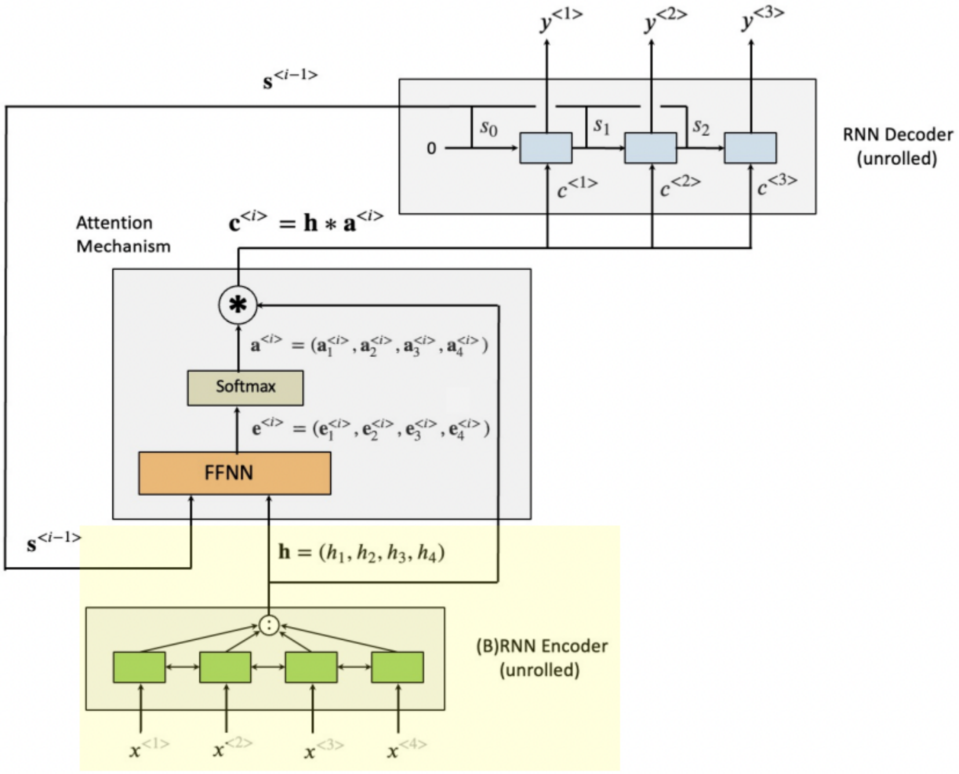
    i. The as are now probability distributions

f. Element-wise product of a<i> and h produces the context vector c<i>.

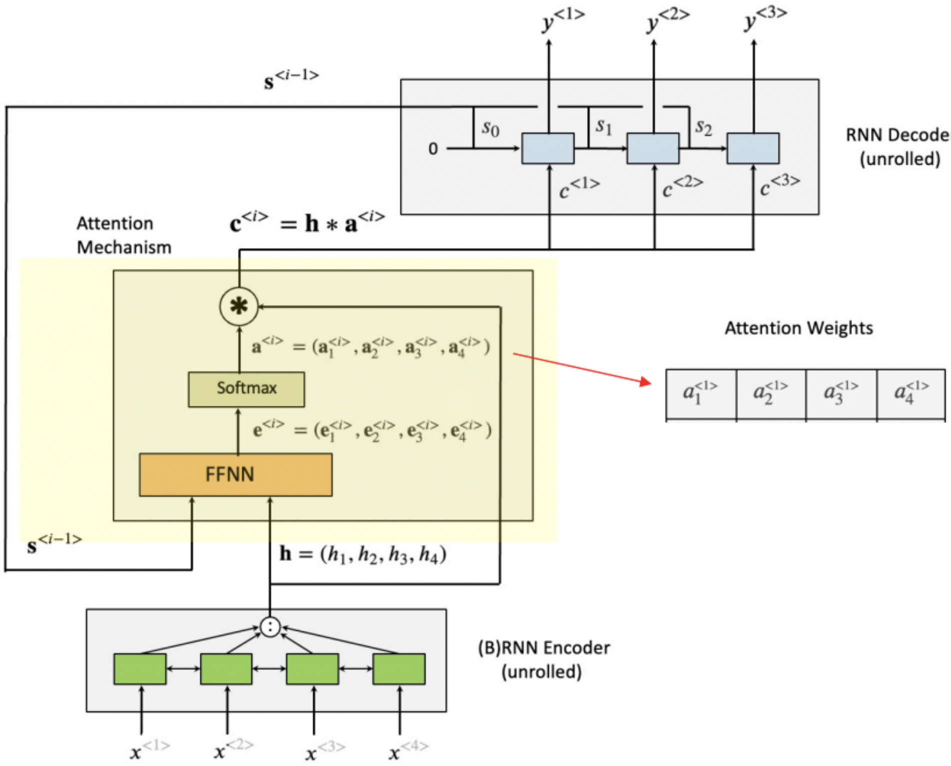
g.



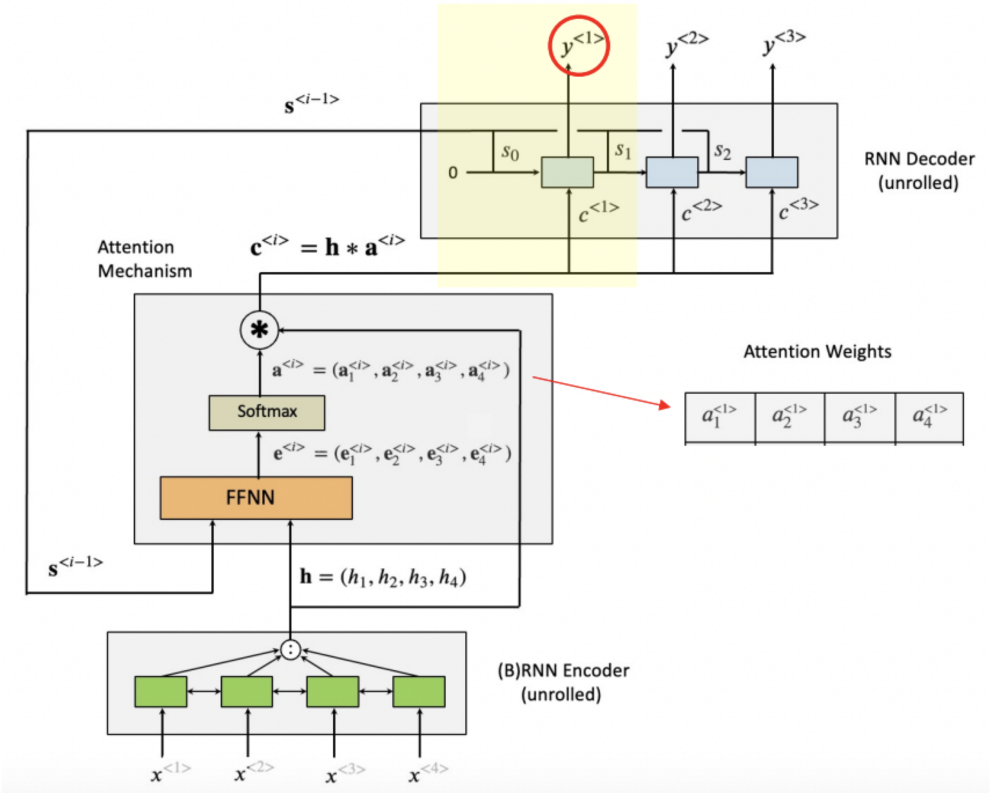
$y^{<1>}$
$y^{<2>}$
$y^{<3>}$
$\mathbf{s}^{<i-1>}$
s_0
s_1
s_2
0
RNN Decoder (unrolled)
$c^{<1>}$
$c^{<2>}$
$c^{<3>}$
Attention Mechanism
$\mathbf{c}^{<i>} = \mathbf{h} * \mathbf{a}^{<i>}$
$\mathbf{a}^{<i>} = (\mathbf{a}_1^{<i>}, \mathbf{a}_2^{<i>}, \mathbf{a}_3^{<i>}, \mathbf{a}_4^{<i>})$
Softmax
$\mathbf{e}^{<i>} = (\mathbf{e}_1^{<i>}, \mathbf{e}_2^{<i>}, \mathbf{e}_3^{<i>}, \mathbf{e}_4^{<i>})$
FFNN
$\mathbf{s}^{<i-1>}$
$\mathbf{h} = (h_1, h_2, h_3, h_4)$
(B)RNN Encoder (unrolled)
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
$x^{<4>}$


$y^{<1>}$
$y^{<2>}$
$y^{<3>}$
$\mathbf{s}^{<i-1>}$
0
s_0
s_1
s_2
RNN Decoder
(unrolled)
$c^{<1>}$
$c^{<2>}$
$c^{<3>}$
Attention
Mechanism
$\mathbf{c}^{<i>} = \mathbf{h} * \mathbf{a}^{<i>}$
$\mathbf{a}^{<i>} = (\mathbf{a}_1^{<i>}, \mathbf{a}_2^{<i>}, \mathbf{a}_3^{<i>}, \mathbf{a}_4^{<i>})$
Softmax
$\mathbf{e}^{<i>} = (\mathbf{e}_1^{<i>}, \mathbf{e}_2^{<i>}, \mathbf{e}_3^{<i>}, \mathbf{e}_4^{<i>})$
FFNN
$\mathbf{s}^{<i-1>}$
$\mathbf{h} = (h_1, h_2, h_3, h_4)$
(B)RNN Encoder
(unrolled)
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
$x^{<4>}$

h.

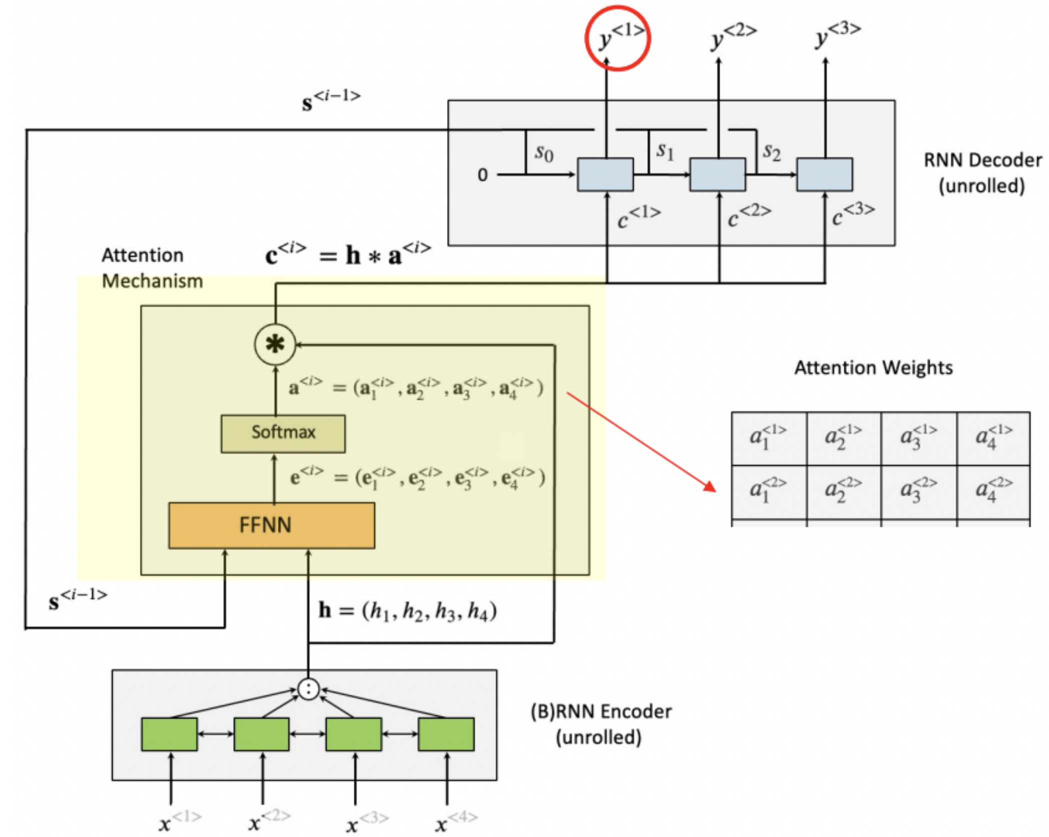

$y^{<1>}$
$y^{<2>}$
$y^{<3>}$
$\mathbf{s}^{<i-1>}$
0
s_0
s_1
s_2
RNN Decoder
(unrolled)
$c^{<1>}$
$c^{<2>}$
$c^{<3>}$
Attention
Mechanism
$\mathbf{c}^{<i>} = \mathbf{h} * \mathbf{a}^{<i>}$
$\mathbf{a}^{<i>} = (\mathbf{a}_1^{<i>}, \mathbf{a}_2^{<i>}, \mathbf{a}_3^{<i>}, \mathbf{a}_4^{<i>})$
Softmax
$\mathbf{e}^{<i>} = (\mathbf{e}_1^{<i>}, \mathbf{e}_2^{<i>}, \mathbf{e}_3^{<i>}, \mathbf{e}_4^{<i>})$
FFNN
Attention Weights
$a_1^{<1>}$
$a_2^{<1>}$
$a_3^{<1>}$
$a_4^{<1>}$
$\mathbf{s}^{<i-1>}$
$\mathbf{h} = (h_1, h_2, h_3, h_4)$
(B)RNN Encoder
(unrolled)
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
$x^{<4>}$

i.


$y^{<1>}$
$y^{<2>}$
$y^{<3>}$
$\mathbf{s}^{<i-1>}$
s_0
s_1
s_2
0
RNN Decoder (unrolled)
$c^{<1>}$
$c^{<2>}$
$c^{<3>}$
Attention Mechanism
$\mathbf{c}^{<i>} = \mathbf{h} * \mathbf{a}^{<i>}$
$\mathbf{a}^{<i>} = (\mathbf{a}_1^{<i>}, \mathbf{a}_2^{<i>}, \mathbf{a}_3^{<i>}, \mathbf{a}_4^{<i>})$
Softmax
$\mathbf{e}^{<i>} = (\mathbf{e}_1^{<i>}, \mathbf{e}_2^{<i>}, \mathbf{e}_3^{<i>}, \mathbf{e}_4^{<i>})$
FFNN
$\mathbf{s}^{<i-1>}$
$\mathbf{h} = (h_1, h_2, h_3, h_4)$
(B)RNN Encoder (unrolled)
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
$x^{<4>}$
Attention Weights

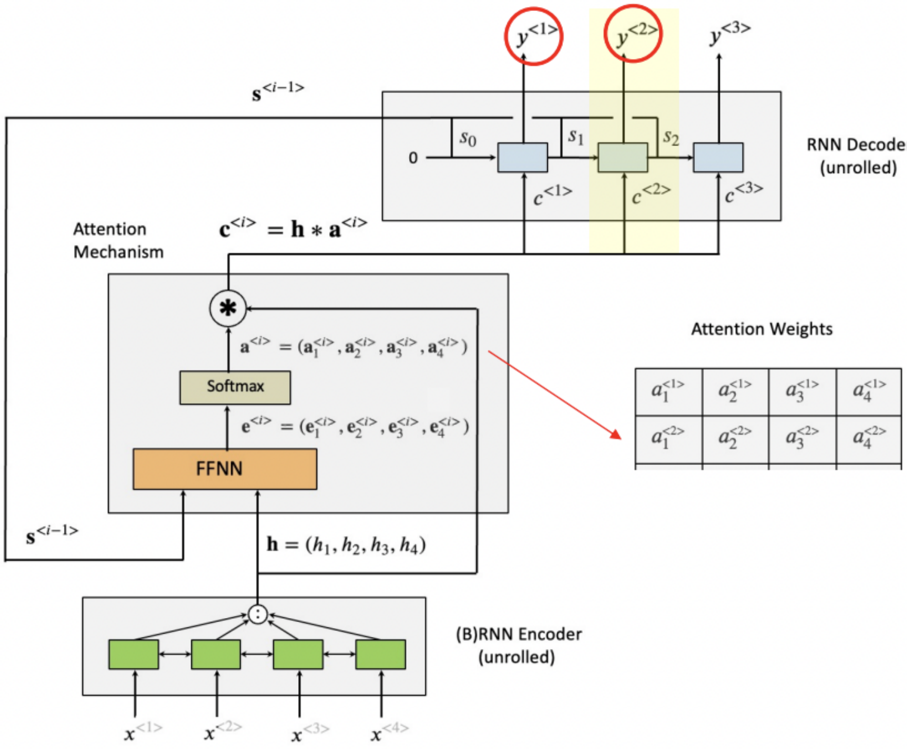
$a_1^{<1>}$	$a_2^{<1>}$	$a_3^{<1>}$	$a_4^{<1>}$

j.


$y^{<1>}$
$y^{<2>}$
$y^{<3>}$
$\mathbf{s}^{<i-1>}$
s_0
s_1
s_2
0
RNN Decoder (unrolled)
$c^{<1>}$
$c^{<2>}$
$c^{<3>}$
Attention Mechanism
$\mathbf{c}^{<i>} = \mathbf{h} * \mathbf{a}^{<i>}$
$\mathbf{a}^{<i>} = (\mathbf{a}_1^{<i>}, \mathbf{a}_2^{<i>}, \mathbf{a}_3^{<i>}, \mathbf{a}_4^{<i>})$
Softmax
$\mathbf{e}^{<i>} = (\mathbf{e}_1^{<i>}, \mathbf{e}_2^{<i>}, \mathbf{e}_3^{<i>}, \mathbf{e}_4^{<i>})$
FFNN
$\mathbf{s}^{<i-1>}$
$\mathbf{h} = (h_1, h_2, h_3, h_4)$
(B)RNN Encoder (unrolled)
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
$x^{<4>}$
Attention Weights

$a_1^{<1>}$	$a_2^{<1>}$	$a_3^{<1>}$	$a_4^{<1>}$
$a_1^{<2>}$	$a_2^{<2>}$	$a_3^{<2>}$	$a_4^{<2>}$

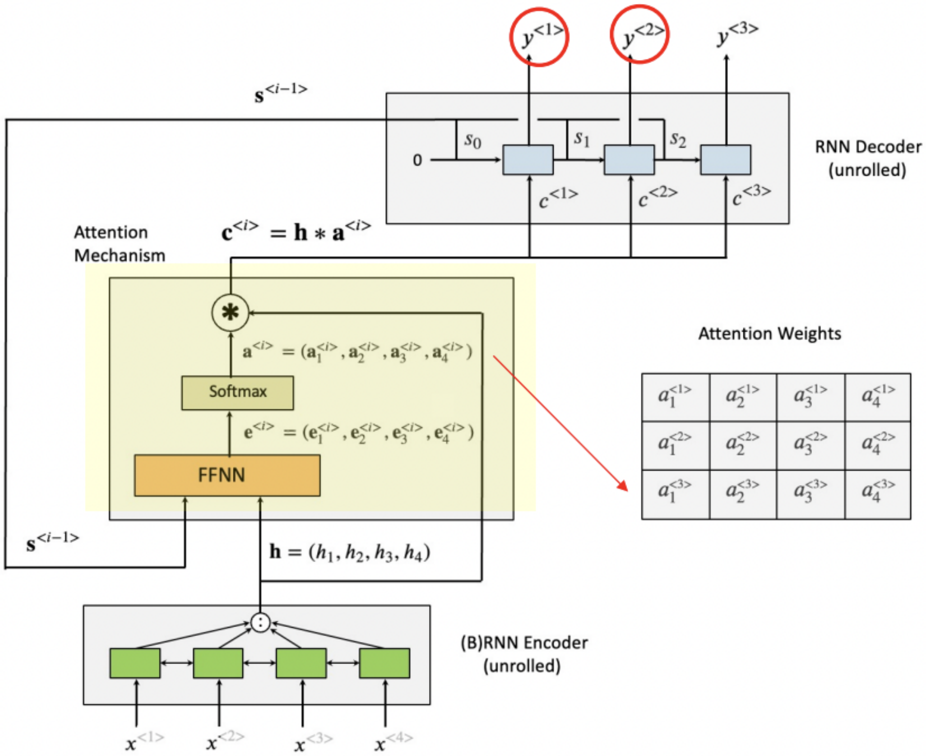
k.



$y^{<1>}$
$y^{<2>}$
$y^{<3>}$
$\mathbf{s}^{<i-1>}$
0
s_0
s_1
s_2
$c^{<1>}$
$c^{<2>}$
$c^{<3>}$
RNN Decoder
(unrolled)
Attention
Mechanism
$\mathbf{c}^{<i>} = \mathbf{h} * \mathbf{a}^{<i>}$
$\mathbf{a}^{<i>} = (\mathbf{a}_1^{<i>}, \mathbf{a}_2^{<i>}, \mathbf{a}_3^{<i>}, \mathbf{a}_4^{<i>})$
Softmax
$\mathbf{e}^{<i>} = (\mathbf{e}_1^{<i>}, \mathbf{e}_2^{<i>}, \mathbf{e}_3^{<i>}, \mathbf{e}_4^{<i>})$
FFNN
Attention Weights
$\mathbf{s}^{<i-1>}$
$\mathbf{h} = (h_1, h_2, h_3, h_4)$
(B)RNN Encoder
(unrolled)
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
$x^{<4>}$

$a_1^{<1>}$	$a_2^{<1>}$	$a_3^{<1>}$	$a_4^{<1>}$
$a_1^{<2>}$	$a_2^{<2>}$	$a_3^{<2>}$	$a_4^{<2>}$

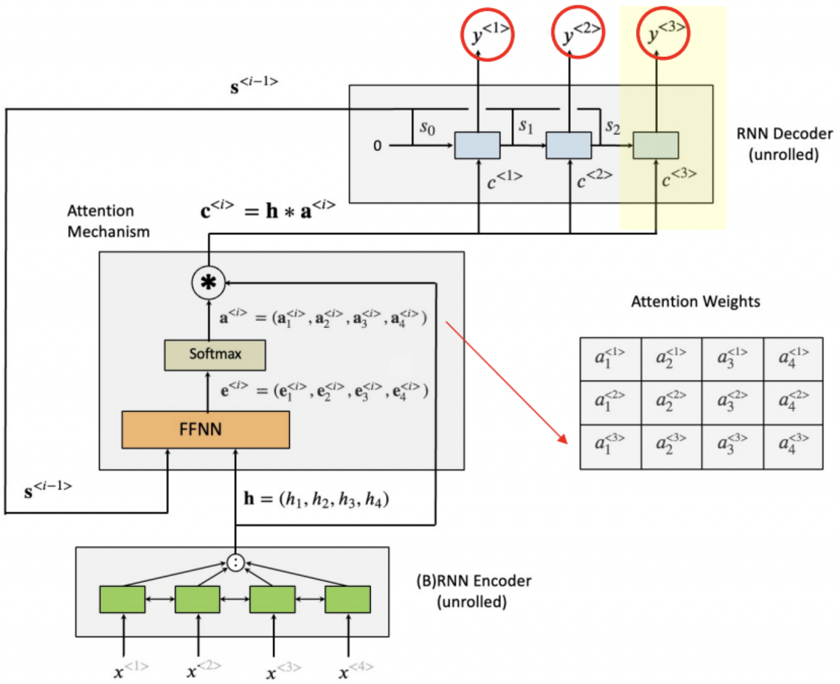
l.



$y^{<1>}$
$y^{<2>}$
$y^{<3>}$
$\mathbf{s}^{<i-1>}$
0
s_0
s_1
s_2
$c^{<1>}$
$c^{<2>}$
$c^{<3>}$
RNN Decoder
(unrolled)
Attention
Mechanism
$\mathbf{c}^{<i>} = \mathbf{h} * \mathbf{a}^{<i>}$
$\mathbf{a}^{<i>} = (\mathbf{a}_1^{<i>}, \mathbf{a}_2^{<i>}, \mathbf{a}_3^{<i>}, \mathbf{a}_4^{<i>})$
Softmax
$\mathbf{e}^{<i>} = (\mathbf{e}_1^{<i>}, \mathbf{e}_2^{<i>}, \mathbf{e}_3^{<i>}, \mathbf{e}_4^{<i>})$
FFNN
Attention Weights
$\mathbf{s}^{<i-1>}$
$\mathbf{h} = (h_1, h_2, h_3, h_4)$
(B)RNN Encoder
(unrolled)
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
$x^{<4>}$

$a_1^{<1>}$	$a_2^{<1>}$	$a_3^{<1>}$	$a_4^{<1>}$
$a_1^{<2>}$	$a_2^{<2>}$	$a_3^{<2>}$	$a_4^{<2>}$
$a_1^{<3>}$	$a_2^{<3>}$	$a_3^{<3>}$	$a_4^{<3>}$

m.



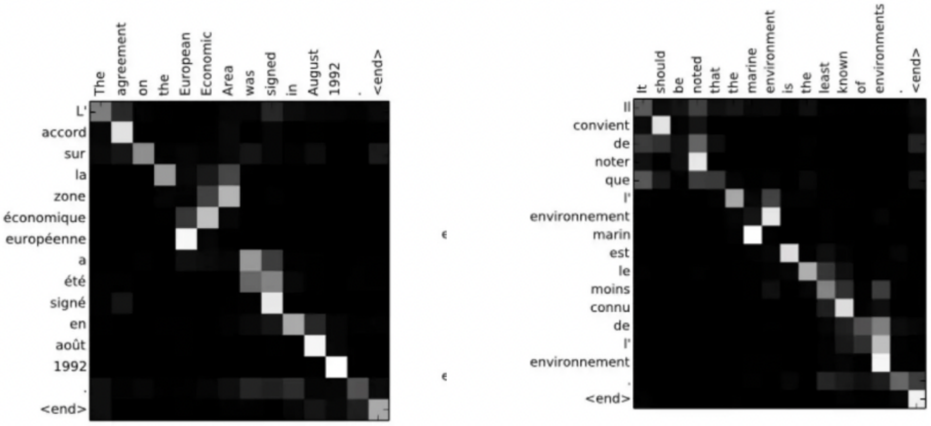
$y^{<1>}$
$y^{<2>}$
$y^{<3>}$
$\mathbf{s}^{<i-1>}$
0
s_0
s_1
s_2
RNN Decoder (unrolled)
$c^{<1>}$
$c^{<2>}$
$c^{<3>}$
Attention Mechanism
$\mathbf{c}^{<i>} = \mathbf{h} * \mathbf{a}^{<i>}$
$\mathbf{a}^{<i>} = (\mathbf{a}_1^{<i>}, \mathbf{a}_2^{<i>}, \mathbf{a}_3^{<i>}, \mathbf{a}_4^{<i>})$
Softmax
$\mathbf{e}^{<i>} = (\mathbf{e}_1^{<i>}, \mathbf{e}_2^{<i>}, \mathbf{e}_3^{<i>}, \mathbf{e}_4^{<i>})$
FFNN
$\mathbf{s}^{<i-1>}$
$\mathbf{h} = (h_1, h_2, h_3, h_4)$
(B)RNN Encoder (unrolled)
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
$x^{<4>}$
Attention Weights

$a_1^{<1>}$	$a_2^{<1>}$	$a_3^{<1>}$	$a_4^{<1>}$
$a_1^{<2>}$	$a_2^{<2>}$	$a_3^{<2>}$	$a_4^{<2>}$
$a_1^{<3>}$	$a_2^{<3>}$	$a_3^{<3>}$	$a_4^{<3>}$

n.

o. This requires a lot of data

p. Displaying the activation matrix shows how attention was applied to the translation:



q.

r. How to train
   i. Need to backpropagate all the weights (takes a long time to develop)

s. Attention can be used in many other contexts such as Automatic Speech Recognition and Image Captioning