ENCODER-DECODER NETWORKS

① So far we've seen how RNNs can be used for the following tasks:

- classification: given a sequence of letters $\langle a_1, \dots, a_N \rangle$, determine what category the sequence belongs to (e.g. sentiment analysis, language detection)
- language modelling: given a sequence of letters $\langle a_1, \dots, a_n \rangle$, determine the most likely next letter a_{n+1}.

---

② A third common application is:

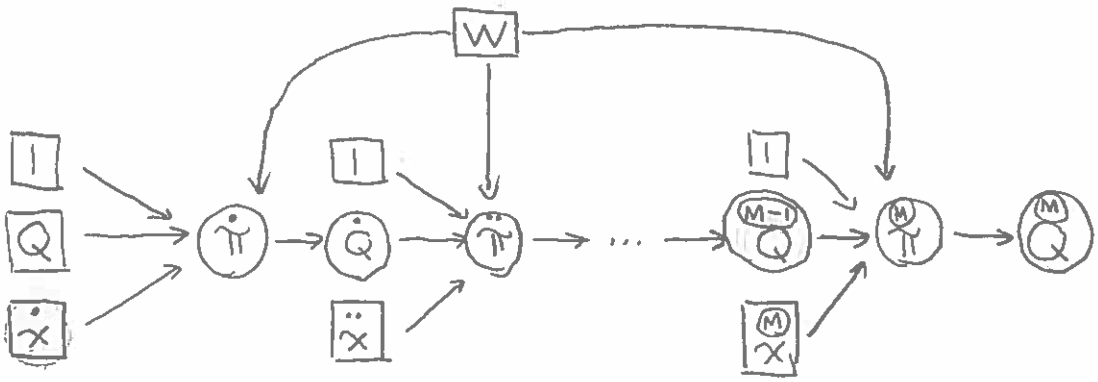
also called transduction

- translation: given a source sequence of letters $\langle a_1, \dots, a_M \rangle$, determine a target sequence of letters $\langle b_1, \dots, b_N \rangle$

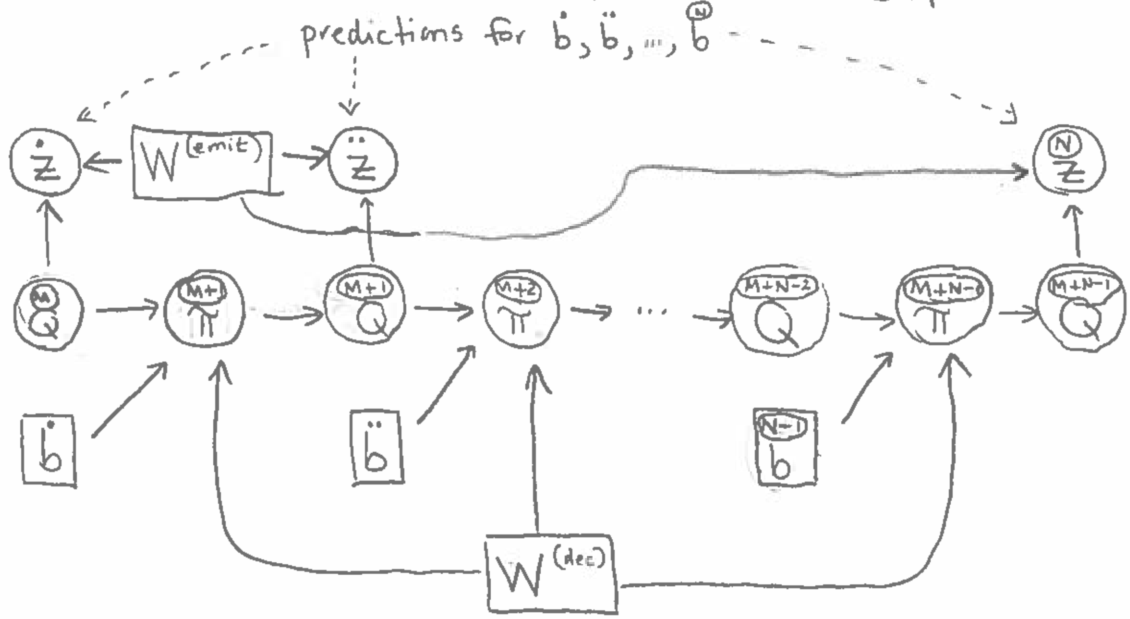
	source seq		target seq	
e.g.	"le chien rouge"	⟶	"the red dog"	(French to english translation)
	"abcdefg"	⟶	"gfedcba"	(string reversal)
	"我的饺子在哪里"	⟶	"1101101"	(Chinese tokenization)

ENCODER-DECODER NETWORKS

③ A common architecture for translation is an <u>encoder-decoder</u> network. The "encoder" is just a standard RNN for classification:

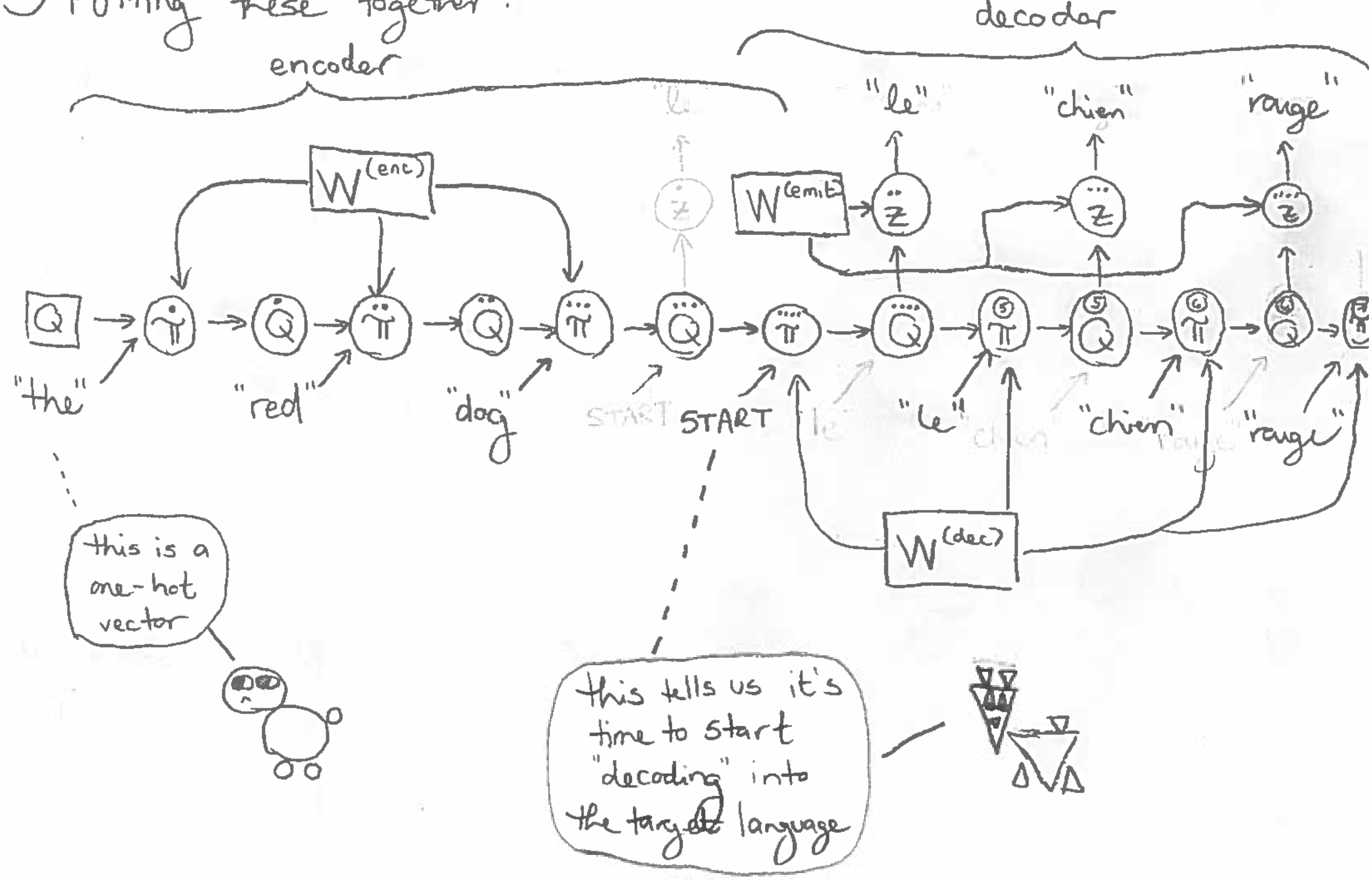


④ But instead of using the final state for classification, we use it to generate a target sequence, using a second RNN that looks like an RNN LM:



ENCODER - DECODER NETWORKS

5) Putting these together:



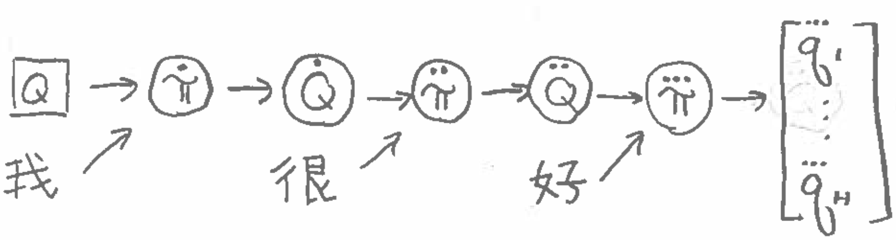
ENCODER-DECODER NETWORKS

⑥ One issue sometimes encountered in practice is that "long" input is translated more poorly than "short" input:

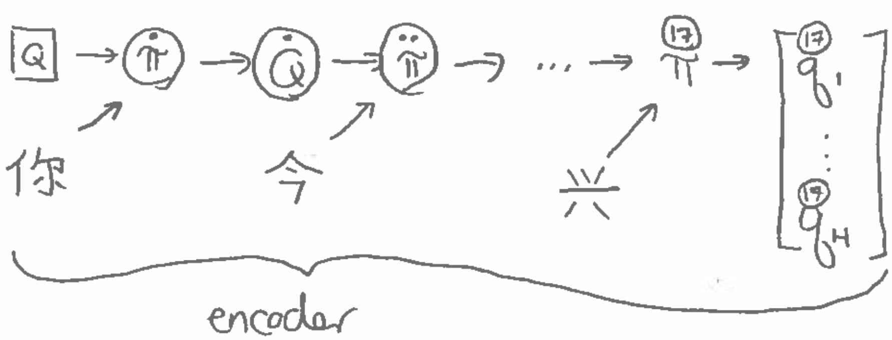
我很好。 → I am fine.

你今天来上课我非常惊讶，但我很高兴。 → Total surprise but joyfulness seeing you in class.

⑦ This is often attributed to the fact that, no matter how long the input sentence, the encoder's final state is always the same size:



this H-length vector needs to encode the sentence 我很好



this H-length vector needs to encode the sentence 你今天来上课我非常惊讶，但我很高兴

Encoder-Decoder Networks

8) That seems a bit unfair. A popular recent strategy to improve the performance of RNNs on long input sequences is to incorporate "attention mechanisms" into the decoder.

---

9) Attention mechanisms are designed to augment the current state $\overset{(k)}{Q}$ with an ability to refer back to one or more input words.

Consider:

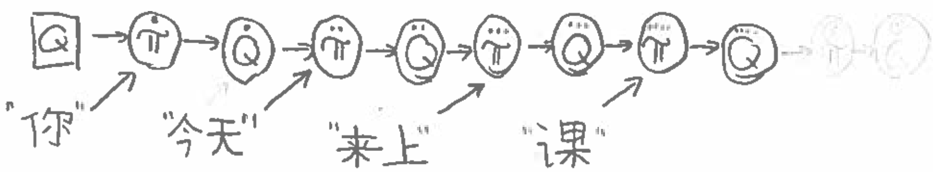
你	今天	来上	课
you	today	came to	class

A good translation could be:

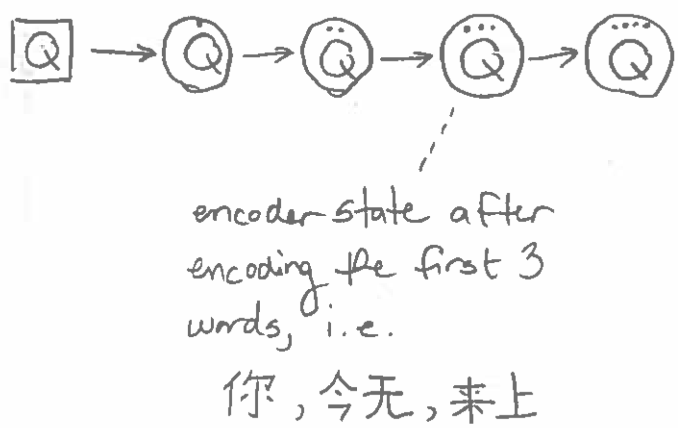
You came to class today

ENCODER-DECODER NETWORKS

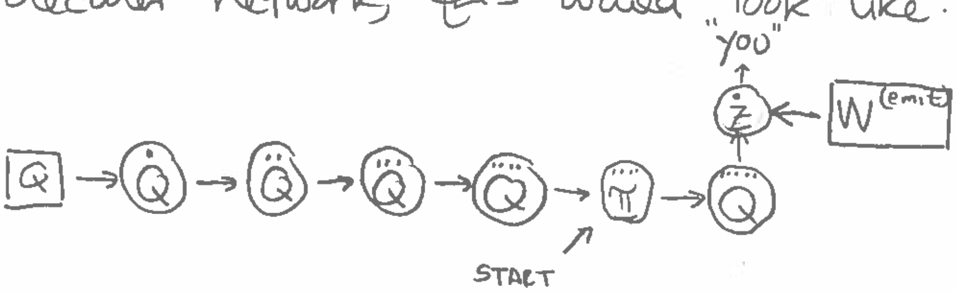
(10) Consider the encoding phase:



For convenience, let's abbreviate this with just the states:

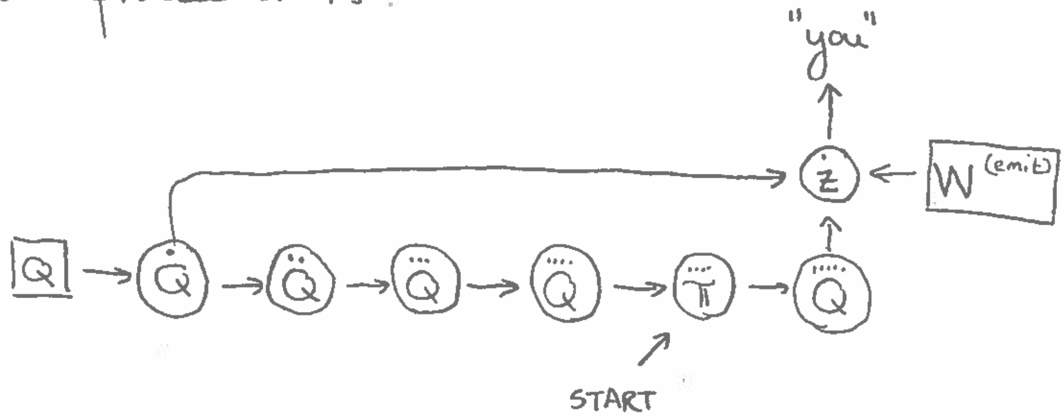


(11) Now we want to decode $\overset{....}{Q}$ into the first word of the English translation, i.e. "you". In a typical encoder-decoder network, this would look like:

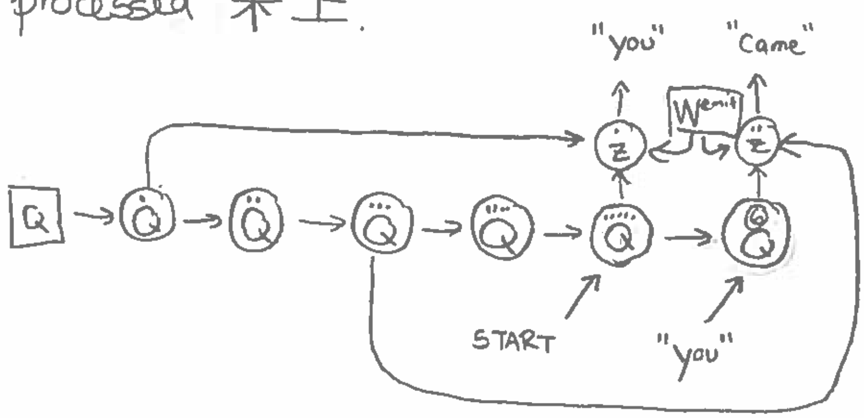


ENCODER-DECODER NETWORKS

(12) But it might help, since we're about to translate "你", if we could review what the encoder state looked like right after we processed 你.



(13) The next English word we'd like to produce is "came", which translates from the third token 来上, so it might help if we could review the encoder state after we processed 来上.



But we really don't know what this network structure should look like in advance; it depends on the input sentence.

(14) Could we learn it?

Suppose we do something similar to the bitmasking we did in LSTMs. Consider the following strategy:

- compute a bitmask m of size K where K is the number of encoder states, e.g. a bitmask for $K=3$ could be:

$$m = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

- construct the $H \times K$ matrix M whose k^{th} column is the k^{th} encoder state $\overset{(k)}{Q}$
- let $c = Mm$, e.g.

$$c = \begin{bmatrix} \dot{q}_1 & \ddot{q}_1 & \dddot{q}_1 \\ \vdots & \vdots & \vdots \\ \dot{q}_H & \ddot{q}_H & \dddot{q}_H \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \ddot{q}_1 \\ \vdots \\ \ddot{q}_H \end{bmatrix} = \ddot{Q}$$

This allows the network to "select" an encoder state from all of the encoder states.

Encoder-Decoder Networks

(15) One way to implement the bitmask is to take the dot product of each encoder state with the current decoder state; e.g. if our current decoder state is $\overset{(6)}{Q}$:

$$\begin{bmatrix} \dot{Q}^T \overset{(6)}{Q} \\ \ddot{Q}^T \overset{(6)}{Q} \\ \dddot{Q}^T \overset{(6)}{Q} \\ \ddddot{Q}^T \overset{(6)}{Q} \end{bmatrix} \overset{\text{(e.g.)}}{=} \begin{bmatrix} -3.8 \\ -3.2 \\ 2.7 \\ 1.2 \end{bmatrix}$$

larger dot products indicate greater correspondence between the two states

(16) Running this vector through a softmax function gives us a probability distribution:

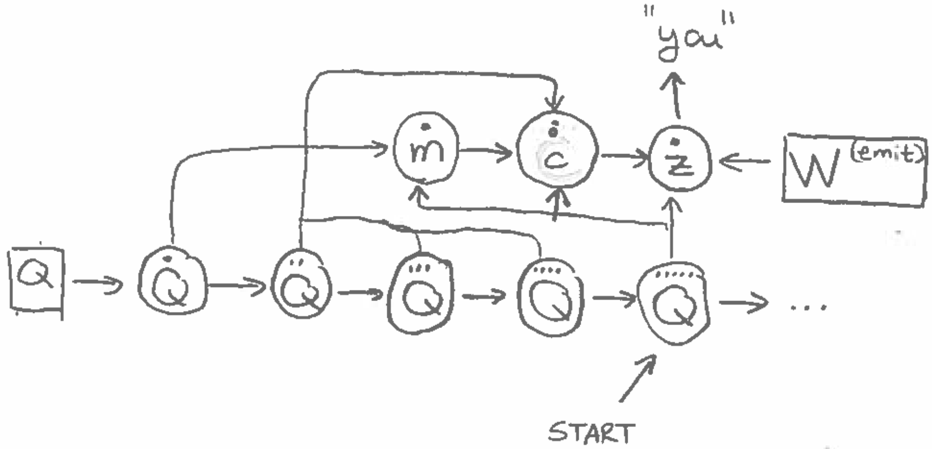
$$\text{softmax}\left(\begin{bmatrix} -3.8 \\ -3.2 \\ 2.7 \\ 1.2 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0.82 \\ 0.18 \end{bmatrix}$$

(17) This becomes our "soft" bitmask:

$$\begin{bmatrix} \dot{q}_1 & \ddot{q}_1 & \dddot{q}_1 & \ddddot{q}_1 \\ \vdots & \vdots & \vdots & \vdots \\ \dot{q}_H & \ddot{q}_H & \dddot{q}_H & \ddddot{q}_H \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ .82 \\ .18 \end{bmatrix} = .82 \cdot \dddot{Q} + .18 \cdot \ddddot{Q}$$

ENCODER-DECODER NETWORK

(18) Putting it all together:



where:

$$\dot{m} = \text{softmax}\left(\begin{bmatrix} \dot{Q}^T \overset{\cdot\cdot\cdot\cdot\cdot}{Q} \\ \ddot{Q}^T \overset{\cdot\cdot\cdot\cdot\cdot}{Q} \\ \dddot{Q}^T \overset{\cdot\cdot\cdot\cdot\cdot}{Q} \\ \ddddot{Q}^T \overset{\cdot\cdot\cdot\cdot\cdot}{Q} \end{bmatrix} \right)$$

$$\dot{c} = \begin{bmatrix} \dot{Q} & \ddot{Q} & \dddot{Q} & \ddddot{Q} \end{bmatrix} \dot{m}$$

(19) This describes just one of many available attention mechanisms (called "Luong global attention"). Generally, attention mechanisms improve the performance of RNNs on long input sequences:

~~Total surprise but joyfulness seeing you in class~~

I'm surprised but happy you came to class today.