# Weekly Report of NLP

**Haoyu Wang**
School of Software Engineering
Shanghai Jiao Tong University
gogowhy@sjtu.edu.cn

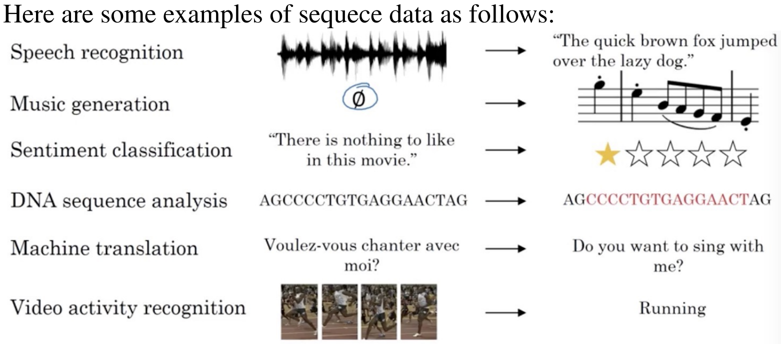
## Abstract

In this section, i will give a brief summary of the technique used for NLP(natural language processing).

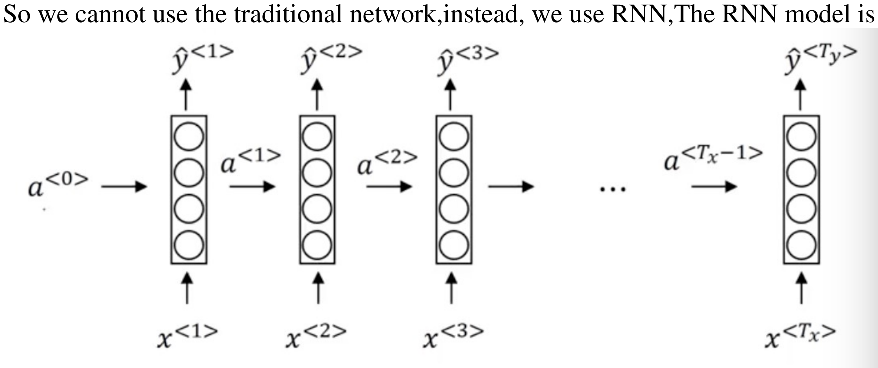
## 1 RNN(recurrent neural network)

### 1.1 model

Here are some examples of sequece data as follows:




For example, we can use NLP to recognize people's name, at first we should create a vocabulary list, and mark the words in a sentence in the vector with "1" where the word is and with "0" where the word is not.
Beacause:1. the output and input can be different lengths in different examples
2.it doesn't share features learned across different positions of text.
So we cannot use the traditional network,instead, we use RNN,The RNN model is as follows:

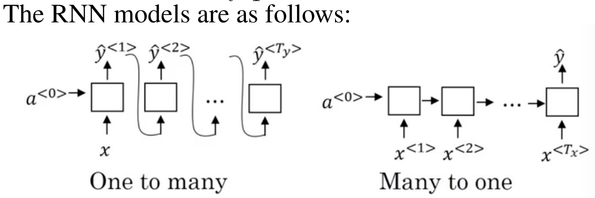



$a[0] = 0$
$a[1] = g(Waaa[0] + Waxx[1] + ba)$(with tanh or reLu )
$y(train)[1] = g(Wyaa[1] + by)$ (with sigmoid)
$a[t] = g(Waaa[t-1] + Waax[t] + ba)$

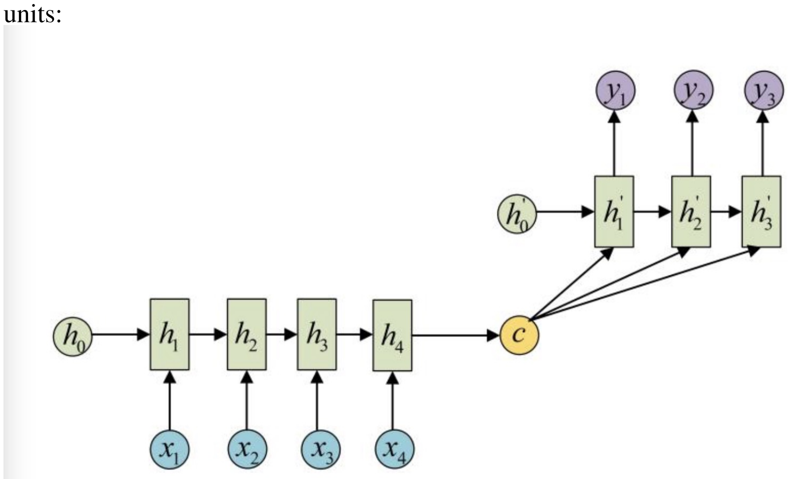
$y(train)[t] = g(Wyaa[t] + by)$

### 1.2 back prog

The loss function of an RNN is as follows:
$L[t](y(train)[t], y[t]) = -y[t]logy(train)[t] - (1 - y[t])log(1 - y(train)[t])$
$L(y, y(train)) = \sum_{i=1}^{n} L(y[i], y(train)[i])$
The RNN models are as follows:

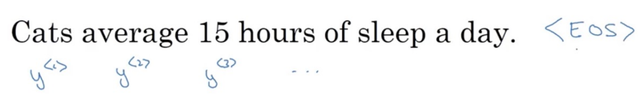



The moset widly used model is as follows, which does not limit the number of decoder or encoder units:

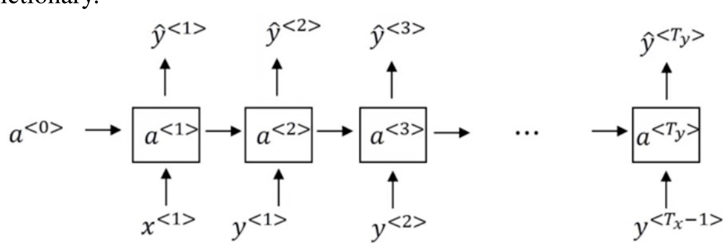


### 1.3 language modeling

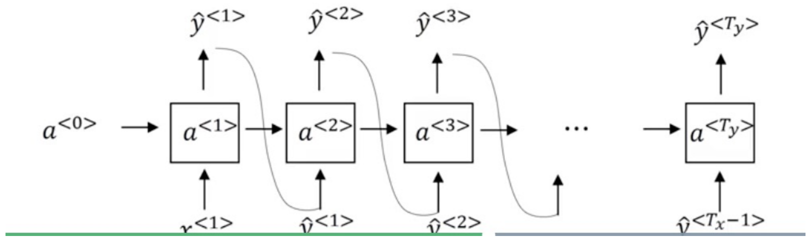
Language model helps us to check out what is the probability of a sentence.We first use large corpus of english texts, and calculate the probability of each word in a sentence.
For example: we first tokenize the following sentence by marking it with y[1],y[2],y[3], and a [EOS] to mark that it reaches the end of the sentence and a [UNK] to mark that the word is not in




the dictionary.

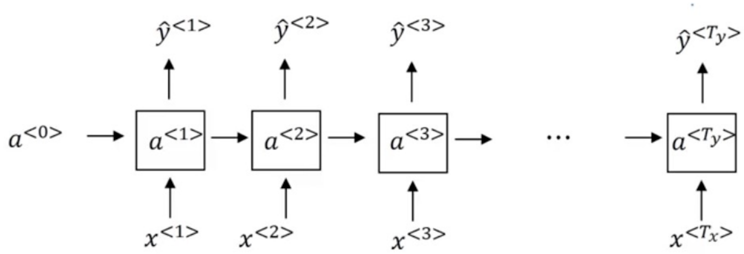


$L(y(train)[t]) = -\sum_{i=1} yi[t]logy(train)i[t]$
$L = -\sum_{L}[t](y(train)[t], y[t])$
$p(y[1], y[2], y[3]) = p(y[1])(y[2]|y[1])p(y[3]|y[1]y[2])$
The softmax tells the chance of a token.




### 1.4 vanishing gradients of RNNs and solutions

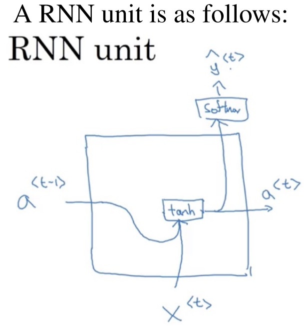
A traditional RNN is as follows




If a sentence says, ”The cat,which already ate food, was full”,the remembering of singular proform would be possibly missing, the machine usually fails to recognize whether it’s singular of plural proform.”Was” or ”were”, which one is correct? Here are some ways to solve the problem.

## 2 Gated Recuurent Unit(GRU)

A RNN unit is as follows:




$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

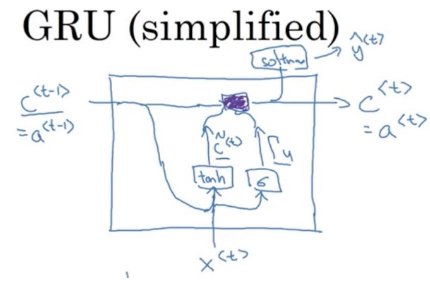
The GRU is going to have a new variable C(memory cell) to remember it’s proform. For GRU, c[t]=a[t]!!!

$$c(new)[t] = tanh(Wc[c[t-1], x[t] + bc])$$

$\Gamma u = \sigma(Wu[c[t-1], x[t] + bu])$(u=update gate)

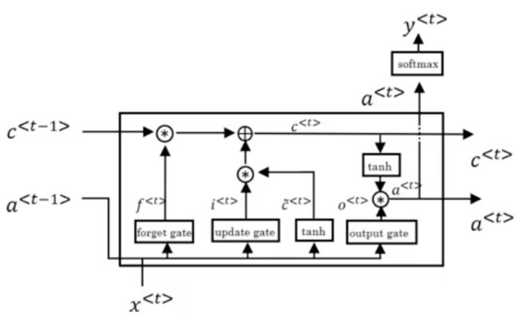
c would be set either 1 or 0 to memorize the proform of the word or so.And the job of the gate, of gamma u, is to decide when do you update these values. In particular, when you see the phrase, the cat, you know they you’re talking about a new concept the especially subject of the sentence cat. So that would be a good time to update this bit

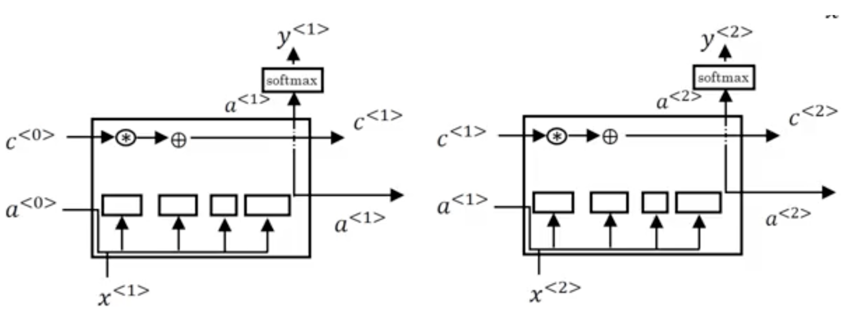
$$c[t] = \Gamma u * c(new)[t] + (1 - \Gamma u) * c[t-1]$$




## 3 Long Short Term Memory (LSTM)

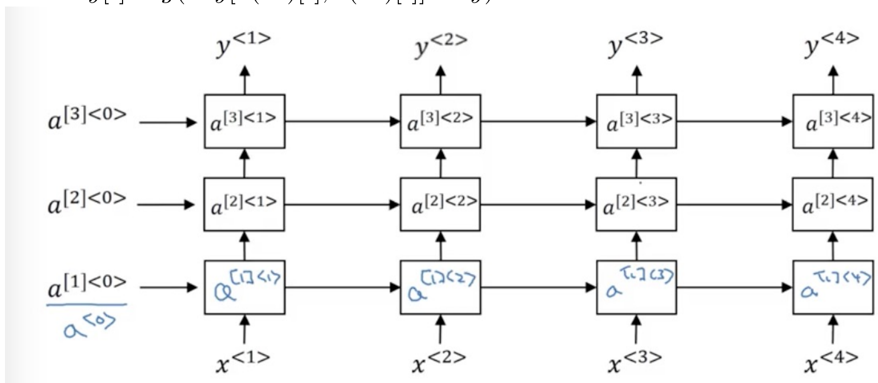
In the LSTM:
$c(new)[t] = tanh(Wc[s[t-1], x[t]] + bc)$
$\Gamma u = (Wu[a[t-1], x[t] + bu])$
$\Gamma f = \sigma(Wf[s[t-1], x[t]] + bf)$
$c[t] = \Gamma u * c(new)[t] + \Gamma f * c[t-1]$
$a[t] = \Gamma output * c[t]$



## 4 Bidirectional RNNs and deep RNNs

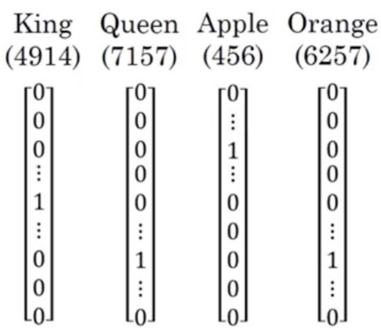
Bidirectional: $y[t] = g(Wy[a(\rightarrow)[t], a(\leftarrow)[t]] + by)$




Deep:
For example: $a[2][3] = g(Wa[2][a[2][2], a[1][3]] + ba[2])$

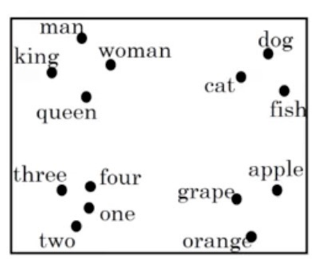
## 5 Word Embeddings

in the dictionary V, V=[a,aarom,......zulu,¡UNK¿]




King is represented as O(4914), Queen as O(7157).

| | Man (5391) | Woman (9853) | King (4914) | Queen (7157) | Apple (456) | Orange (6257) |
|---|---|---|---|---|---|---|
| Gender | −1 | 1 | -0.95 | 0.97 | 0.00 | 0.01 |
| Royal | 0.01 | 0.02 | 0.93 | 0.95 | -0.01 | 0.00 |
| Age | 0.03 | 0.02 | 0.70 | 0.69 | 0.03 | -0.02 |
| Food | 0.09 | 0.01 | 0.02 | 0.01 | 0.95 | 0.97 |




$e(man) - e(women) = \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ e(man)-e(women)=e(king)- e?

WE use the Cosine similarity to calculate the similarity:

$Sim(e(w), e(king) - e(man) + e(women)) \leftarrow Sim(u, v) = \frac{u^T v}{\|u\| \|v\|}$ The whole Embedding matrix is called E

$E * O(6257) = e(6257) \rightarrow E * O(j) = e(j)$

In the example ”I want a glass of orange ( ) ”, the words are combined as (in vector form):

| | | | | | | |
|---|---|---|---|---|---|---|
| I | $o_{4343}$ | $\longrightarrow$ | $E$ | $\longrightarrow$ | $e_{4343}$ |
| want | $o_{9665}$ | $\longrightarrow$ | $E$ | $\longrightarrow$ | $e_{9665}$ |
| a | $o_1$ | $\longrightarrow$ | $E$ | $\longrightarrow$ | $e_1$ |
| glass | $o_{3852}$ | $\longrightarrow$ | $E$ | $\longrightarrow$ | $e_{3852}$ |
| of | $o_{6163}$ | $\longrightarrow$ | $E$ | $\longrightarrow$ | $e_{6163}$ |
| orange | $o_{6257}$ | $\longrightarrow$ | $E$ | $\longrightarrow$ | $e_{6257}$ |

Context c (”orange 6257”) $\rightarrow$ Target t (”juice 4834 ”)

Oc $\rightarrow$ E $\rightarrow$ ec $\rightarrow$ O(softmax) $\rightarrow$ y(train)

Softmax: $p(t|c) = \frac{e^{(}\theta t^T e^c)}{\sum_{j=1}^{N} e^{(}\theta j^T ec)}$

$L(y(train), y) = -\sum_{i=1}^{N} yilogy(train)i$

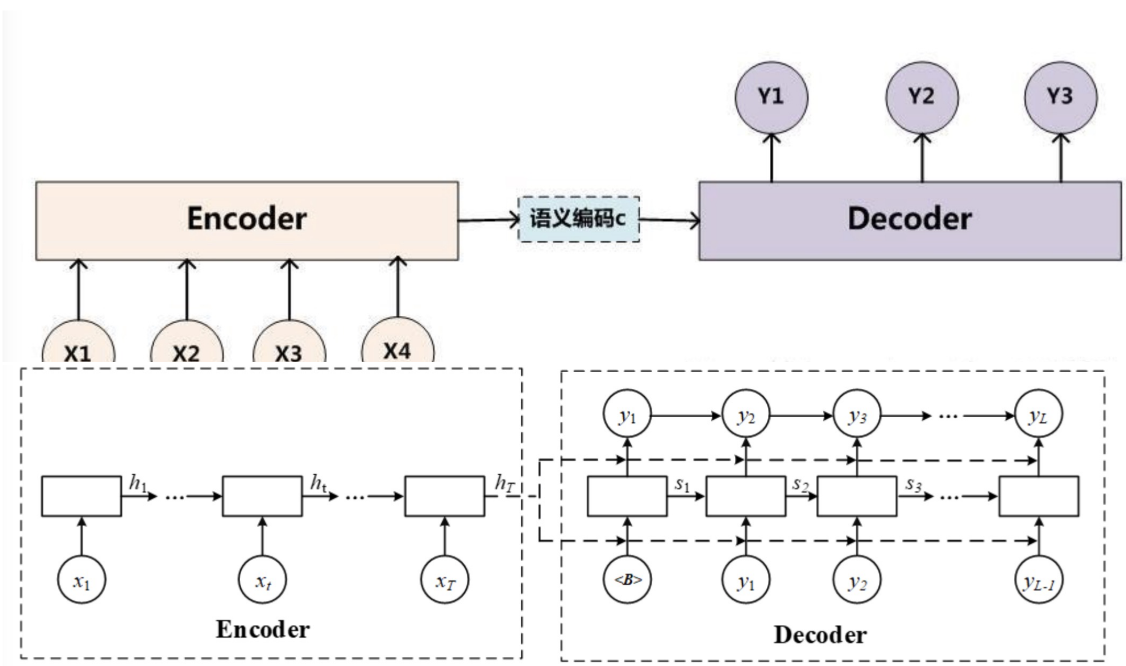
## 6 GloVe word vectors

”I want a glass of orange juice to go along with my cereal.”

It makes Xij= number of times that i appears in the context of j (i as t and j as c)

minimize $\sum_{i=1}^{N} \sum_{j=1}^{N} f(Xij)(\theta^T \theta j + bi + bj - logX^2 ij)$

## 7 traditional encoder-decoder model




$s(t) = f(s(t-1), y(t-1), hT)$ $y(t) = g(s(t), y(t-1), hT)$

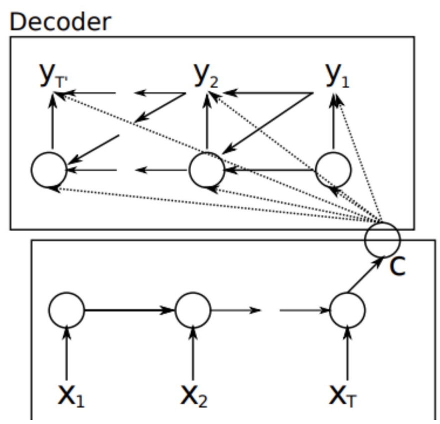
We can use different encoders and decoders to finish the work ,whatever, such as

CNN,RNN,BiRNN,GRU,LSTM or so.

### 7.1 encoder

For an input sequence x=(x1,x2,......xTx), we first change it into a context vector c, the present hidden condition is determined by the last condition and the present input: $h(t) = f(Xt, h(t-1))$ gethering all of the hidden units, we would get $c = q(ht, .....hTx)$ (the h and q functions are not linear)

### 7.2 decoder




According to the context vector and the expected wordsy1,y2......y(t-1), decoder is used to predict y(t) , so the probability is as follows:

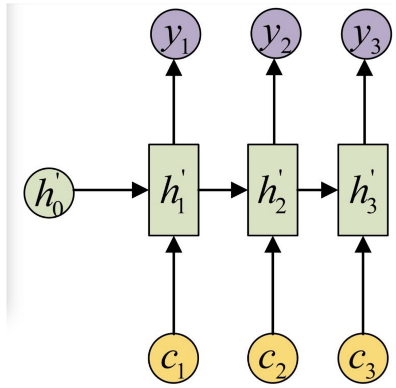
$$p(y) = \prod_{t=1}^{T} p(y_t | \{y_1, \ldots, y_t\}, c) \qquad s_t = f(s_{t-1}, y_{t-1}, c) \qquad p(y) = \prod_{t=1}^{T} p(y_t | \{y_1, \ldots, y_t\}, c) = q(y_{t-1}, s_t, c)$$

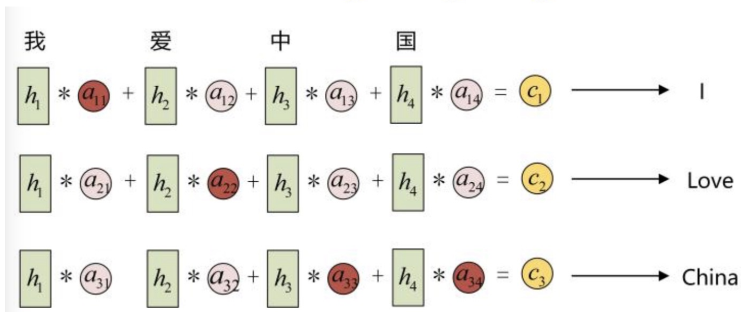
### 7.3 problems

No matter how long we input , the encoder would change it into a fixed-length vector, decoder is limited by the length of vector c.So we take in the Attention machanism.

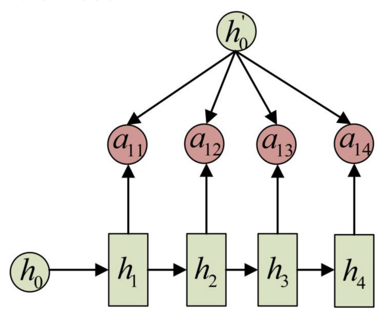
## 8 Attention

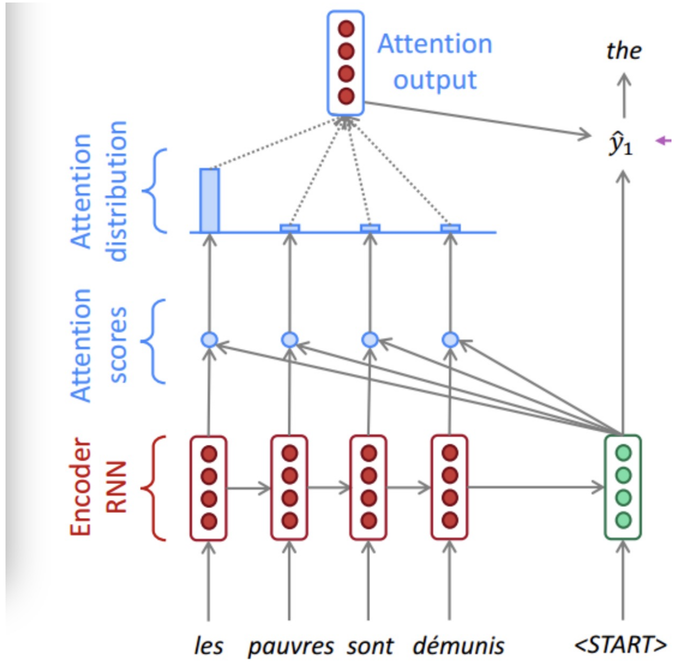
Attention is a machanism that differs the weight of different parts of a sequence to make the transfer more accurately. It does not need encode into a vector, instead, it gives different sequeces ,and relys on the decoder to choose which sequences to use in the next step.






the h1,h2,h3,h4 knows the words ,and estimate the probability and relativity betweec1 and the word. So when getting the first word in this example , a11 is definitely the biggest ”I”; c3 relates with h3,h4 so a33 and a34 are the biggest value.Then where do we get the value a? a is learned from the model .






## 9 plan

My plan is to learn the cs224n by stanford to know more about NLP.