Why sequence models

Example of sequence data.

- → Speech recognition.
- → Music generation.
- → Sentiment classification.
- → DNA seq. analysis.
- → Machine translation.
- → Video activity recog.
- → Name entity recog.

Notation:

e.g. Name Entity Recog.

x: Harry Potter and Hermione Granger invented a new spell.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<t>}$ ... $x^{<9>}$

y: 1 1 0 1 1 0 0 0 0

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $y^{<t>}$ ... $y^{<9>}$

$T_x = T_y = 9$. length of x and y.

$x^{(i)<t>}$: for the t^{th} index of the i^{th} training example

$T_x^{(i)}$: the length of the i^{th} training example.

Same for $y^{(i)<t>}$ and $T_y^{(i)}$

Representing Words.

x: Harry Potter and Hermione Granger invented a new spell.

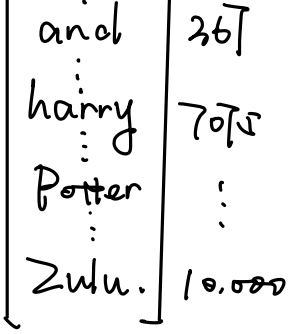
$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<t>}$ ... $x^{<9>}$

e.g. Vocabulary: We have 10,000 words in it.

a	1
aaron	2
:	

One way to represent words: One-hot encoder:

[0 ... ↑



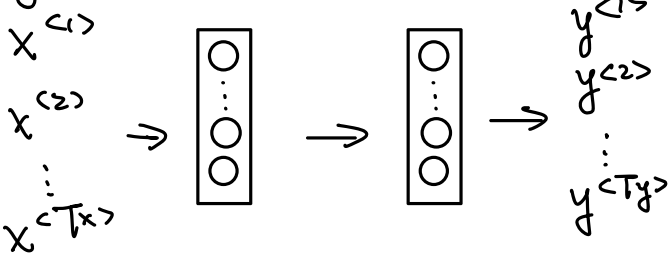
$x^{<1>}$:  ← 40T

10,000 ↓ for all the words.

For words never appear in the vocab. may use a new token <UNK>

Recurrent NN model.

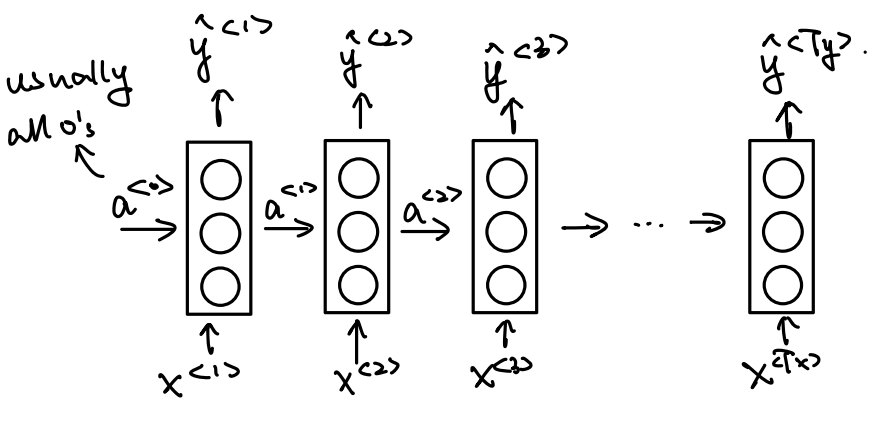
Why not a standard NN.



Problem:

- Inputs outputs can be different length in diff. examples.
- Doesn't share features learned across diff pos. of text.

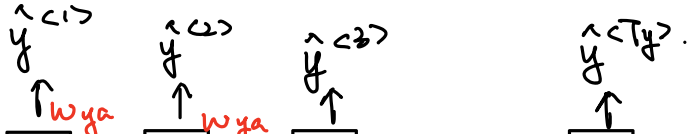
Recurrent NN.

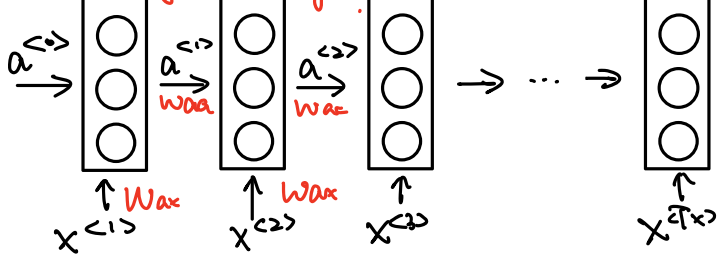


e.g. for the prediction of $y^{<3>}$ it does use information from $y^{<1>}$ and $y^{<2>}$.

(But it doesn't have info later)

Forward Propagation.





Computation Step :

Choice of activation g :

$a^{<0>} = \vec{0}$, $a^{<1>} = g(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a)$ ← tanh. ReLU

$\hat{y}^{<1>} = g(W_{ya}a^{<1>} + b_y)$. ← Sigmoid.

...

$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$

$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$.

Simplified Notation for RNN.

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

$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$.

Suppose W_{aa} is (100, 100) matrix. W_{ax} is (100, 10,000) matrix.

$W_a = [W_{aa} \mid W_{ax}]$ (100 ↓) W_a is (100, 10100) matrix.

(100 → ← 10000 →)

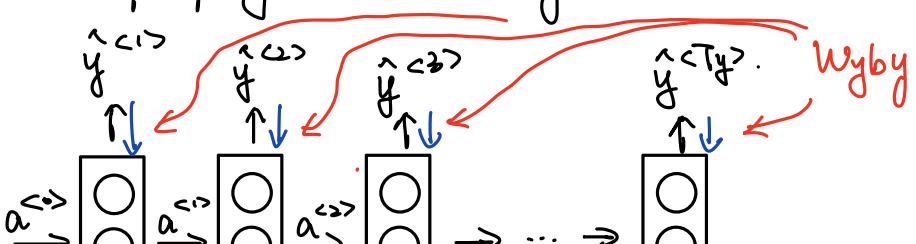
$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ \hline x^{<t>} \end{bmatrix}$ (100 ↓ / 10000 ↓) (10100 ↓)

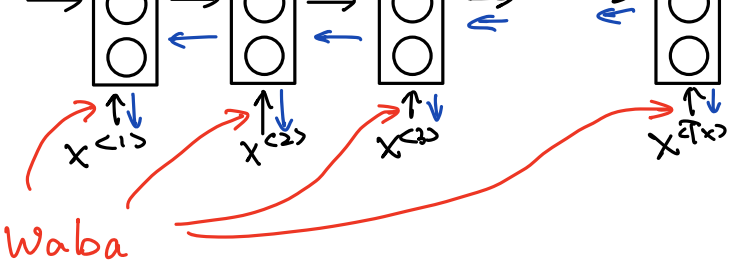
So $W_a[a^{<t-1>}, x^{<t>}] = W_{aa}a^{<t-1>} + W_{ax}x^{<t>}$

Rather than use two weight matrices W_{aa} and W_{ax}. we only use W_a.

Similarly. $\hat{y}^{<t>} = g(W_y a^{<t>} + b_y)$.

Backpropagation Through time.



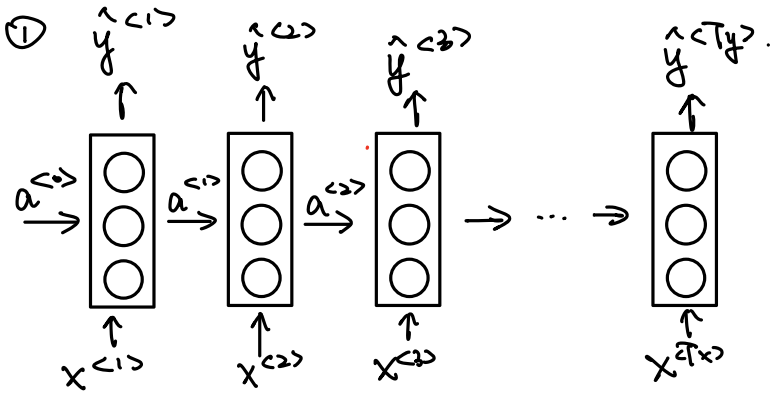


$$\mathcal{L}^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}) = -y^{\langle t \rangle} \log \hat{y}^{\langle t \rangle} - (1 - y^{\langle t \rangle}) \log (1 - \hat{y}^{\langle t \rangle})$$

$$\mathcal{L}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}) = \sum_{t=1}^{T_y} \mathcal{L}^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle})$$ Backprop through time.

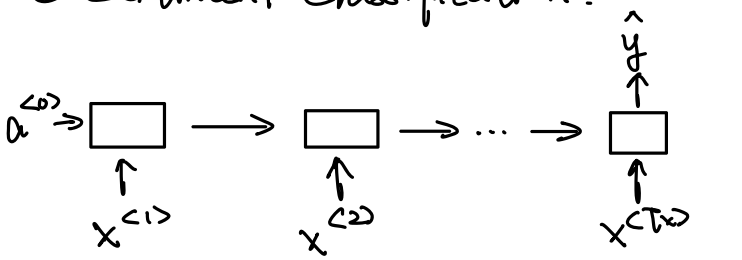
Different Types of RNNs. (Previously $T_x = T_y$, but $T_x \neq T_y$ in general).

e.g. of RNN Architecture.

①  $T_x = T_y$.

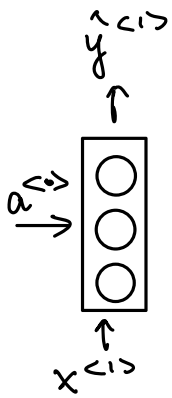
Many to many (Equal).

② Sentiment Classification.

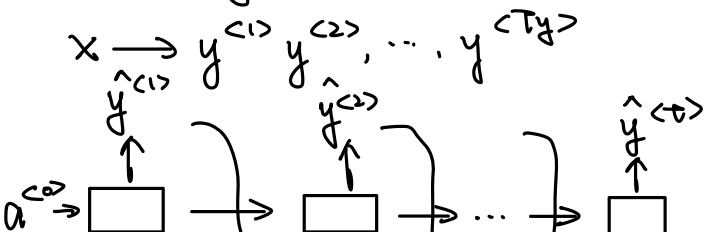


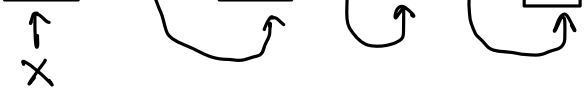
Many to one.

③ of couse you may have one to one.

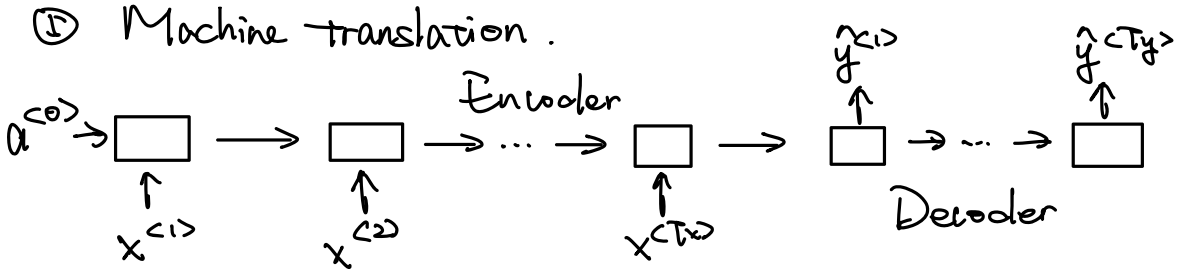


④ Music generation.





③ Machine translation.



Many to many. (where Tx may not equal Ty).

Language Model and Sequence Generation.

What is language modeling?

Speech recog.

① The apple and pair salad.

② The apple and pear salad.

Maybe the second one is much more likely.

$P(\text{Sentence ①}) = 3.2 \times 10^{-13}$. $P(\text{Sentence ②}) = 5.7 \times 10^{-10}$. ✓.

Basically, the language model. take in a sequence $(y^{<1>}, y^{<2>}, \cdots, y^{<Ty>})$ and compute: $P(y^{<1>}, y^{<2>}, \cdots, y^{<Ty>})$

Language modeling with an RNN.

Training set: large corpus of English text. Vocab size 10000.

e.g. Cats average 15 hours of sleep a day. <EOS>.

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $y^{<8>}$ $y^{<9>}$

e.g. The Egyptian Mau is a bread of cat. <EOS>.

<UNK>. if not in the vocab.

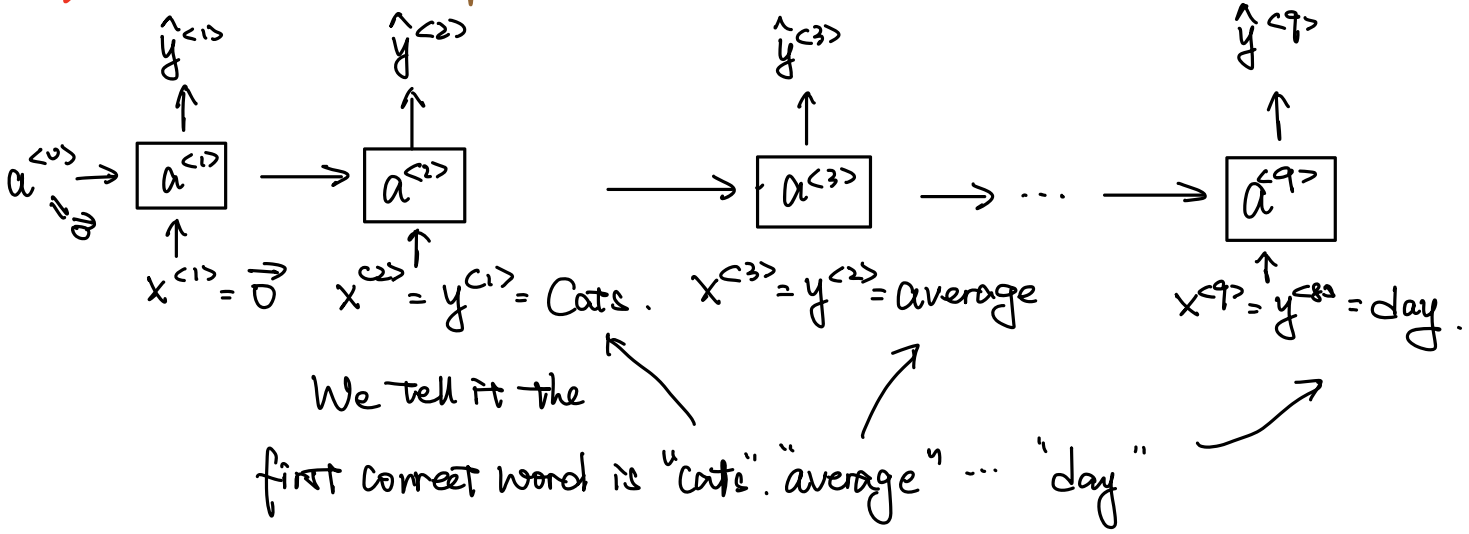
has a softmax to compute:

$P(a)$. $P(\text{aaro})$, ...,

$P(a|\text{cats})$. $P(\text{aaro}|\text{cats})$

$P(\_\,|\,\text{cats, average})$

$P(\_\,|\,\cdots\cdots)$



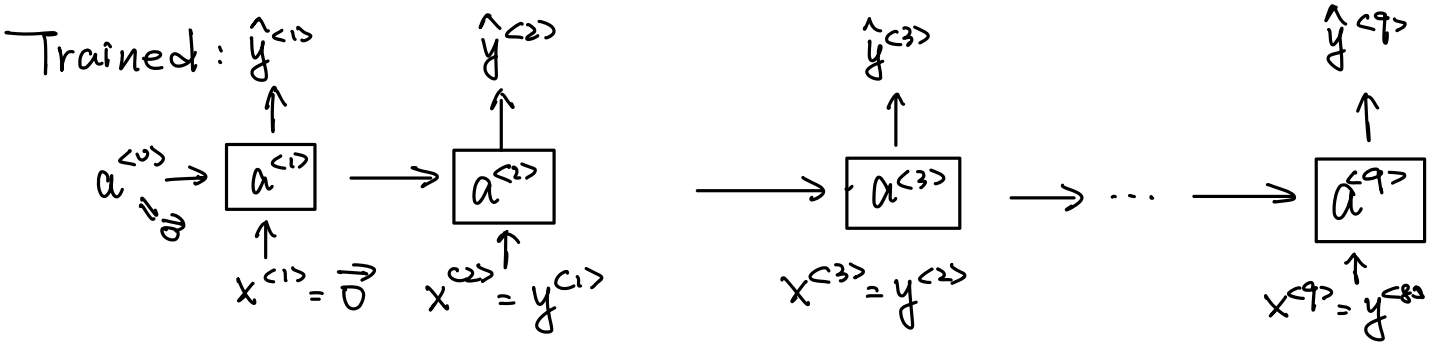
$$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

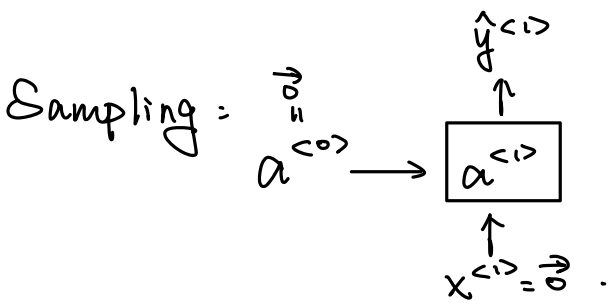
$$\mathcal{L} = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

$$P(y^{<1>}, y^{<2>}, y^{<3>}) = P(y^{<1>}) P(y^{<2>} | y^{<1>}) P(y^{<3>} | y^{<1>}, y^{<2>}).$$

Sampling Novel Seq.

Sampling a seq. from a trained RNN.

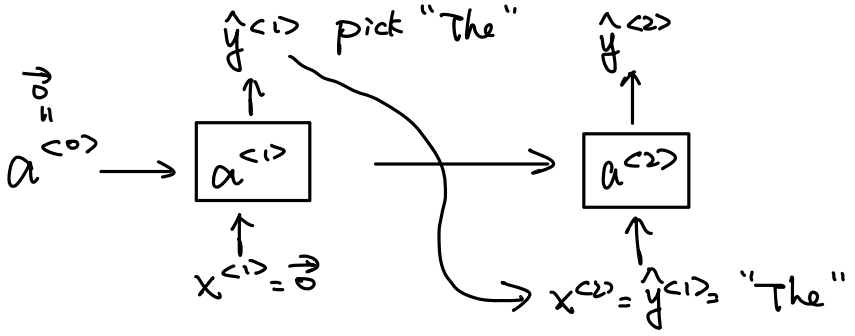




$\hat{y}^{<1>}$ will give you all the probability of all the words in the vocab.

Then we just use np.random.choice to pick one.

Step ③. After we picked one. like we pick "The".



And $\hat{y}^{<2>}$ will give you all the probabilities of words given "The".

We keep continue this process. until:

① Sample certain amount of words.

② We sampled the <EOS> token.

Character level language model.

Vocab = [a. b. c. ···, x. y. z. ␣. . . , . 0. ··· .9. A. B. ···, Z].

Then your $\hat{y}^{<1>}$. $\hat{y}^{<2>}$. will be characters.

e.g. C a t ␣ a v e r a g e

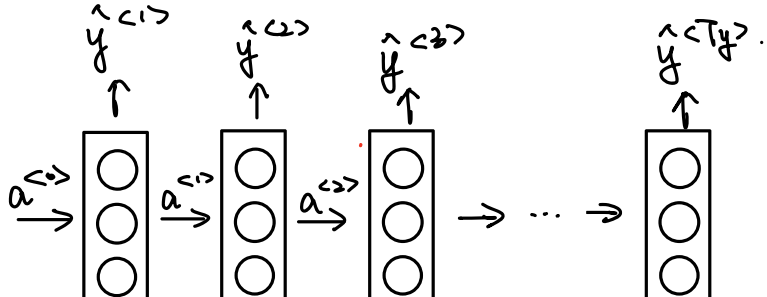
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ...

Disadvantage: Too long. Can't catch long range dependencies.

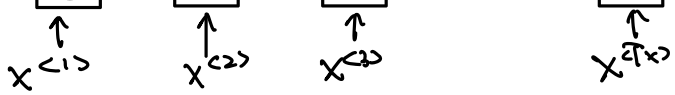
Vanishing gradients with RNNs. Using RNN, hard to catch this

The cat, which already ate ····· , was full.

The cats. which ····· , were full.



it's hard for the diff. between $\hat{y}^{<Ty>}$. and $y^{<Ty>}$ to affect early units like $x^{<2>}$.

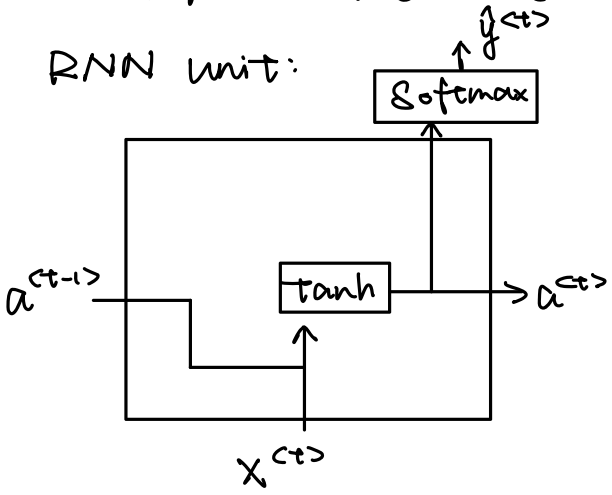


By using RNN. $y^{<t>}$ is mainly affected by close items like $x^{<t-2>}$. $x^{<t-1>}$. $x^{<t>}$

* There might also exist exploding gradient problem. we may saw NaN values. And we can apply gradient clipping. e.g. if your gradient is greater than some threshold. we just rescale the gradient vec. This is easy to solve. Vanishing gradient hard to solve.

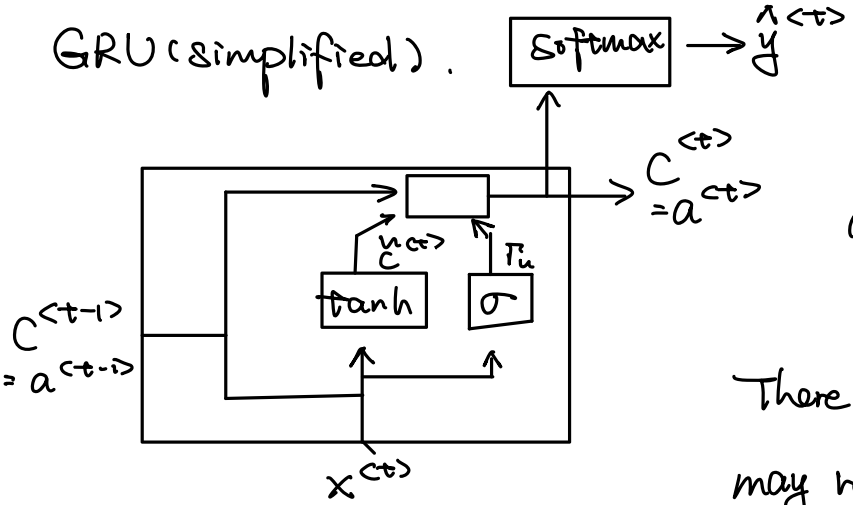
Gated Recurrent Unit. (GRU)

RNN unit:



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

GRU (simplified).



c = memory cell.

at time t. c will have some value $c^{<t>}$.

here $c^{<t>} = a^{<t>}$.

There will be a candidate $\tilde{c}^{<t>}$ at time t

may replace $c^{<t>}$.

$$\tilde{c}^{<t>} = \tanh(W_c[c^{<t-1>}, x^{<t>}] + b_c).$$

There's a gate. $\Gamma_u = \text{Sigmoid}(W_u[c^{<t-1>}, x^{<t>}] + b_u)$

Γ_u decide when to change $c^{<t>}$

$\Gamma_u = 1$ $\Gamma_u = 0.20. = 0. =$

$c^{<t>} = 1$ ..... $= 1$

e.g. The cat, which ....., was full..

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

① if $\Gamma_u = 1$. Set $c^{<t>} = \tilde{c}^{<t>}$. Go ahead and update.

② if $\Gamma_u = 0$. $c^{<t>} = c^{<t-1>}$. No update.

Of course: $c^{<t>}$. $\tilde{c}^{<t>}$. Γ_u can be vector. Thus * represent elementwise product. So that we can update some bits without affect others.

And ~~dff~~. bits for ~~dff~~. use.

Full GRU. r stands for relavence. So Γ_r tells you how relavent is $c^{<t-1>}$

$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$ to compute next candidate for $c^{<t>}$

$\Gamma_u = \text{Sigmoid}(W_u[c^{<t-1>}, x^{<t>}] + b_u)$.

$\Gamma_r = \text{Sigmoid}(W_r[c^{<t-1>}, x^{<t>}] + b_r)$.

$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$

LSTM. (long short term memory) unit.

Previous equations:

$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$

$\Gamma_u = \text{Sigmoid}(W_u[c^{<t-1>}, x^{<t>}] + b_u)$.

$\Gamma_r = \text{Sigmoid}(W_r[c^{<t-1>}, x^{<t>}] + b_r)$

$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$

$a^{<t>} = c^{<t>}$

LSTM. No $c^{<t>}$ and Γ_r.

$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$

$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$.

$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$

$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$

$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$

Gives the memory cell the choice to keep the original value and add new value

Γ_f: forget gate. Γ_o: output gate.

$a^{<t>} = \Gamma_o * \tanh c^{<t>}$

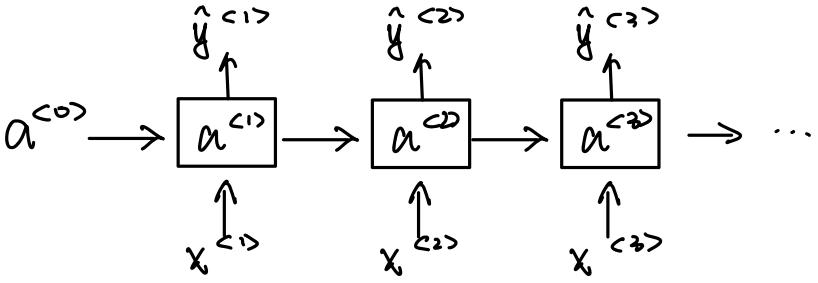
Bidirectional RNN.

Getting information from the future.

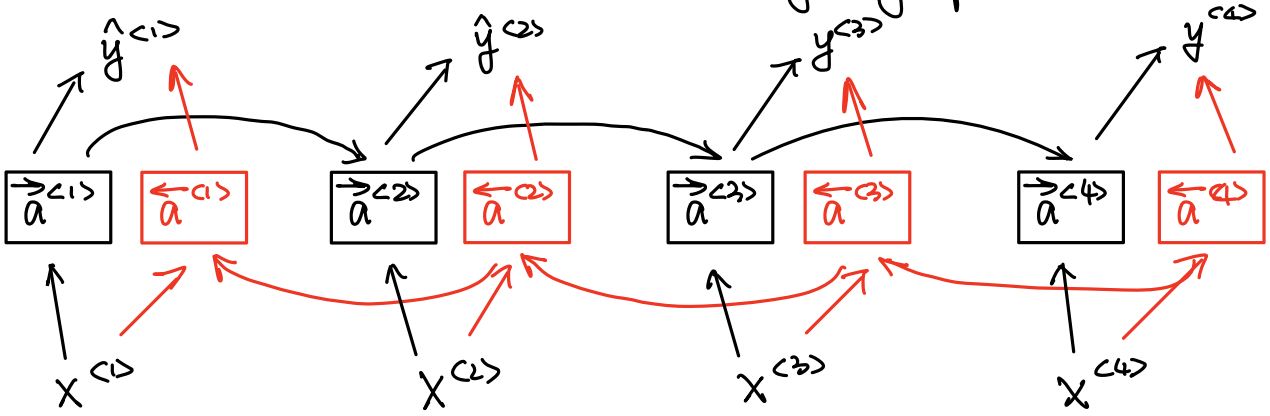
e.g. ① He said: "Teddy bears are on sale!"

② He said: "Teddy Roosevelt was a great president".

We see the word "Teddy". we can't decide it from previous two words: "He said". First two word didn't tell us "Roosevelt" of "bear"



Bidirectional RNN (BRNN). Acyclic graph.

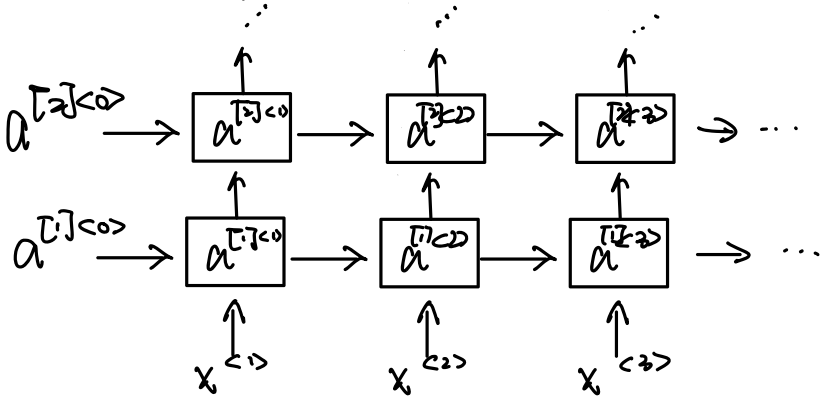


Now we have: $\hat{y}^{<t>} = g(W_y[\overrightarrow{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$

These blocks can also be GRU and LSTM blocks.

Deep RNNs

Deep RNN example.

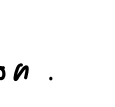


Also we can replace blocks use GRU/LSTM blocks.

$$a^{[3]<3>} = g(W_a^{[3]}[a^{[3]<2>}, a^{[1]<3>}] + b_a^{[3]})$$

Word Representation.

$V = [a, aaron, \cdots, zulu, <UNK>]$. $|V| = 10,000$.

1 hot representation. $a =$ 

Problem. Taking any two representation. The inner product is always 0.
So talking about similarity. There's no difference between "apple" & "orange" and "queen & orange".

Featured representation: word embedding.

e.g.	Man	Woman	King	Queen	Apple	Orange.
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	0.7	0.69	0.03	-0.02
Food.	0.04	0.01	0.02	0.01	0.95.	0.97.

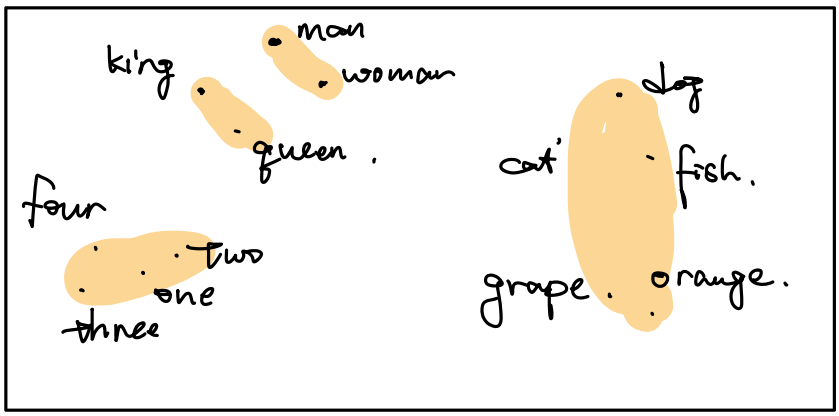
And there are many other types. Allow people to use number to evaluate this thing.

e.g. Maybe there're 300 number of features. And we can use this 300 dim vector to represent words.

So. now it is better to generalise things better.

e.g. Model can learn how to fill words after "apple" by learning from orange.

Visualising word Embeddings:

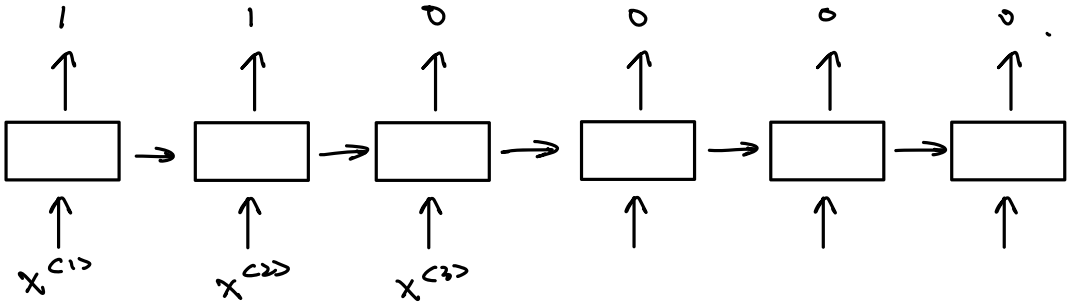


→ 300D → 2D.

t-SNE.

Using word embeddings.

Name entity recognition example: Recog. people's name.



Sally Johnson is an orange farmer.

New input. Test.

Robert Lin is an apple farmer.

Using feature representation. it will be easier for the model to recog. the similarity for "apple" and "orange".

In reality. we have tons of text. 1B to 100B words.

You can use them to train. And then apply to your own model.

Transfer the task to you maybe 100k words model.

Transfer learning and word embeddings.

① Learn word embeddings from large text corpus. (1B-100B words).

(Or download pre-trained models).

② Transfer .....

③ Optional: Continue to finetune the word embeddings with new data

Properties of word embedding.

Analogies:

know man → woman. for king → queen. How to know this?

e.g.	Man	Woman	King	Queen	Apple	Orange.
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	0.7	0.69	0.03	-0.02
Food.	0.04	0.01	0.02	0.01	0.95.	0.97.

Recall this:

$$e_{man} = \begin{bmatrix} -1 \\ 0.01 \\ 0.03 \\ 0.04 \end{bmatrix} \quad e_{woman} = \begin{bmatrix} 1 \\ 0.02 \\ 0.02 \\ 0.01 \end{bmatrix}$$

$$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

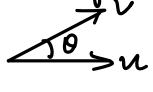
So the diff. are about the same. $e_{man} - e_{woman} \approx e_{king} - e_{queen}$.

Find word w: $\arg\max_w \text{sim}(e_w, e_{king} - e_{man} + e_{woman})$.

Cosine similarity.

$$\text{sim}(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$$

This is actually the cosine of the angle between u and v.



Embedding Matrix: How to learn word embedding.

a	aaron	...	orange (6257)	...	zulu	<UNK>

10000 words in the vocab.

300 E =

O_{6257}: the one-hot encoding vector of "Orange".

Consider $E \cdot O_{6257}$. This will select the feature column of "orange" in E.

Embedding for "orange" in the feature space.

Actually this is not efficient. In practice. use specialized function to look up an embedding in E instead of using matrix multiplication.

Learning word embeddings.

Neural Language Model.

O_j: One hot vector of word at index j.

e_j: embeding vector of word at index j

I want a glass of orange ——

index: 4343 9665 1 3852 6163 6257.

I O_{4343} → E → e_{4343}.

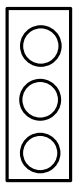
want O_{9665} → E → e_{9665}

a ... e_1

glass ... e_{3852}

of ... e_{6163}

orange. ... e_{6257}

→  → Softmax

Over entire vocab size. Here we have 10,000 possible outputs.

Of course. since there are too many weights. we may only have a fixed history. E.G: 4 words. We only consider "a glass of orange".

Other text/target pairs.

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

Context. → target

Context: Last 4 words.

4 words on left & right. ↓

"a glass of orange ___ to go alone with"

Last 1 word.

"orange ___".

Nearby 1 word. skip gram model.

Word2Vec.

Skip-grams. (Suppose you have following sentense in your training)

I want a glass of orange juice to go alone with my cereal

Context	Target.
orange	juice
orange	glass.
orange	my.

Model:

Vocab size = 10,000k

x → y.

Content c ("orange") → Target t ("juice").

6257 → 4834

To represent "orange". We can start from onehot O_{6257}. or $e_{6257} = E O_{6257}$.

Then we feed e_{6257} into a softmax model. And try to output $\hat{y}$.

$$\text{Softmax} = p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

θ_t: param associate with output t.

$$L(\hat{y}, y) = -\sum_{i=1}^{10,000} y_i \log \hat{y}_i \quad \text{(Usual loss for softmax)}.$$

$y = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ ← index 4834.

So the matrix E will have a lot of parameters.

And we keep training it.

Problem with softmax classification.

If |vocab| = 10,000k. Computation too large for $p(t|c)$.

Other method: Hierachical softmax. e.g. 10000 size.

So that don't need to sum over all the 10,000 examples.

Once we choose the context c. we can sample target t in certain window

How to sample the context c?

e.g. Sample less usual words. (Not words like "a". "the". "of".)

Negative Sampling. Deliberately sample negative examples to train.

Defining a new learning problem. (New supervised learning).

I want a glass of orange juice to go alone with my cereal

x: context, word — y: Target

	context	word	Target
Pair:	orange	juice.	1
↑	orange	king ← pick from random	0
k	orange	book	0
	orange	the	0
↓	orange	of ← also "of" appear next to "orange"	0
	↑ c	↑ t	↑ y.

New learning task:

input: a pair of word.

output: will they appear together or not.

Normally: K = 5-20 for small dataset.

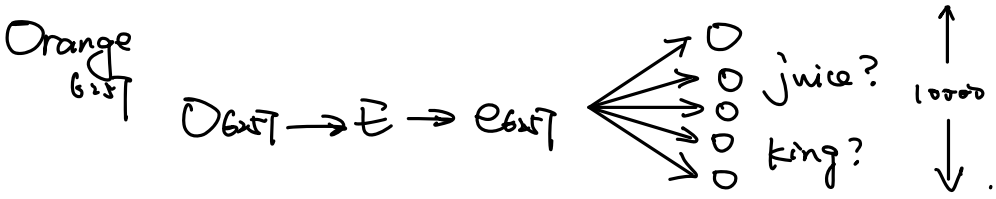
K = 2-5 for large dataset.

we can think of we have $\frac{1}{k}$ pos&neg sample.

Model:

Softmax = $p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$ (from previous slide).

Now: $p(y=1|c,t) = \sigma(\theta_t^T e_c)$



Turn a 10,000 softmax problem into 10000 binary classification problem

And we only deal k+1 of them.

Selecting Negative example. How.

① Sample according to appear freqeuence. (Dont sample a lot of "a" "the")

② Equal probability. (None representative).

* ③ $p(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10,000} f(w_j)^{3/4}}$. The freqeuncy of words appear but take power of $\frac{3}{4}$.

GloVe word vectors. (GloVe: global vectors for word representation).

I want a glass of orange juice to go alone with my cereal

X_{ij}: # times i appear in content of j

t c

This captures how often does word i and j appear together.

Model:

minimize $\sum_{i=1}^{10,000} \sum_{j=1}^{10000} f(x_{ij})(\theta_i^T e_j + b_i + b_j - \log x_{ij})^2$

t c

weighting term

o?

① $f(x_{ij}) \geq 0$ if $x_{ij} = 0$

Convention: $0 \log 0 = 0$

② Don't give too much weight for words like "a", "the", "of" ...

Don't give too less weight for infreq. words like "durian"

θ_i and e_j are symmetric. Need to initialize θ and e at random.

$$e_w^{final} = \frac{e_w + \theta_w}{2}$$

A note on the featurization view of word embedding.

Sometimes. the feature representation might be hard for human understanding.

Sentiment Classification

e.g Problem.

x	→	y.
The dessert is excellent.		★★★★★.
Service was quite slow.		★★
...		...

Question: You might won't have a large enough training set.

Simple sentiment classification e.g.

The dessert is excellent ★★★★★.

8928 2468 4694 3180.

The $O_{8928} \rightarrow E \rightarrow e_{8928}$

dessert e_{2468}

is ... e_{4694} → [Avg.] → ○ → $\hat{y}$

excellent. e_{3180}

Avg.: 300 Dim

softmax.: 1-5 stars

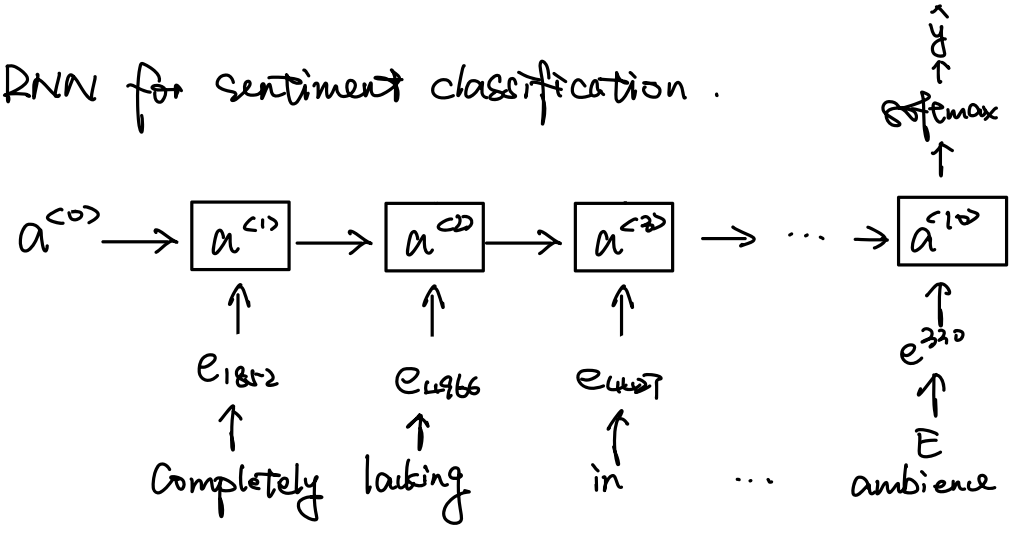
If E was trained on a large data set. E.G. 100B words.

Notice: word sequence doesn't matter.

Consider: "Complete lacking in good taste, good service. good ambience".

Word "good" appear multiple time. Model might think this sentence is a good review. (if we take "Avg" or "Sum").

RNN for sentiment classification.



So this might consider the Completely negative effect of word "lacking" performed towards the sentiment of the sentence.

Debiasing word embeddings.

The problem of bias in word embedding.

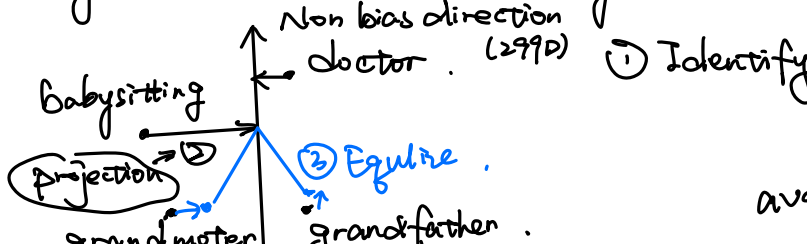
Man: Woman as King: Queen.

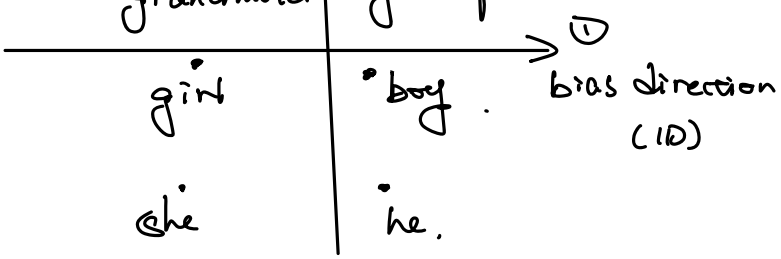
Man: Computer_Programer as Woman: Homemaker X.

Father: doctor as Mother: Nurse. X.

Word embedding can reflect gender, ethnicity, age. sexual orientation. and other biases of the text used to train the model.

Addressing bias in word embeddings.





② Neutralize. For every word that is not definitional, project to get rid of bias

Most words are not definitional. And shall be neutralized

Not like word "grandmother" "father." which have specific gender info. Like word "doctor" or "nurse" which should not include gender info.

↓

Train a linear classifier to decide.

③ Equalize pairs: e.g. Dist(grandmother, babysit) = Dist(grandfather, babysit)

This is relative small. Can just hand pick them.

Basic Models.

Seq to Seq model.

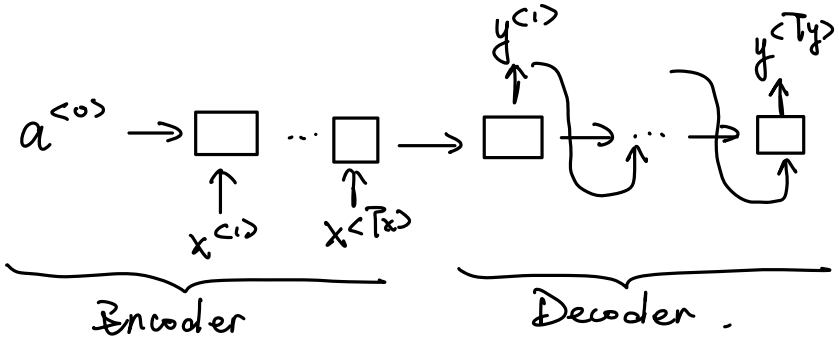
$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$

Jane visite l'Afrique en septembre

→ Jane is visiting Africa in september.

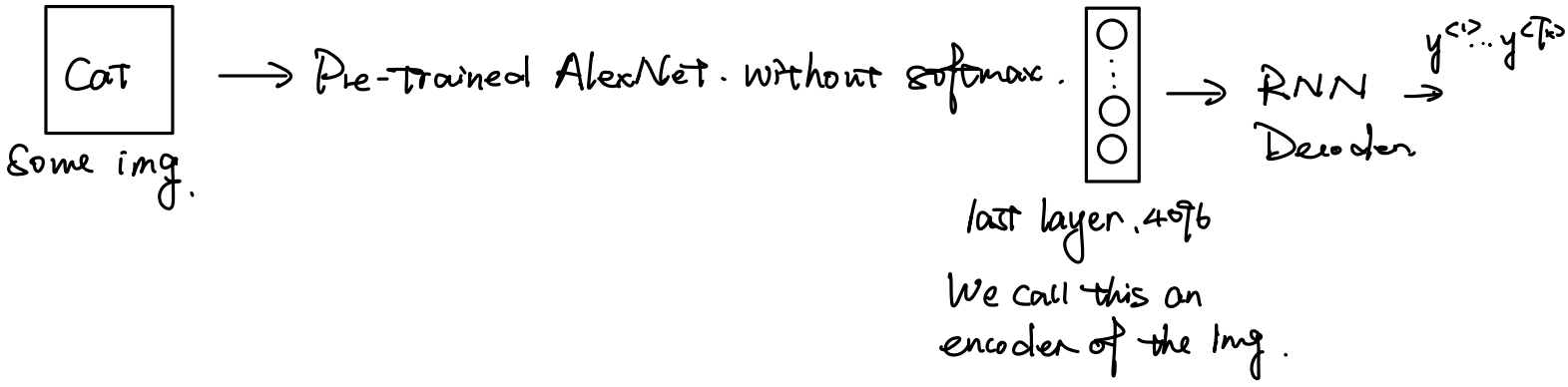
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$

How to train a neural net can do this?



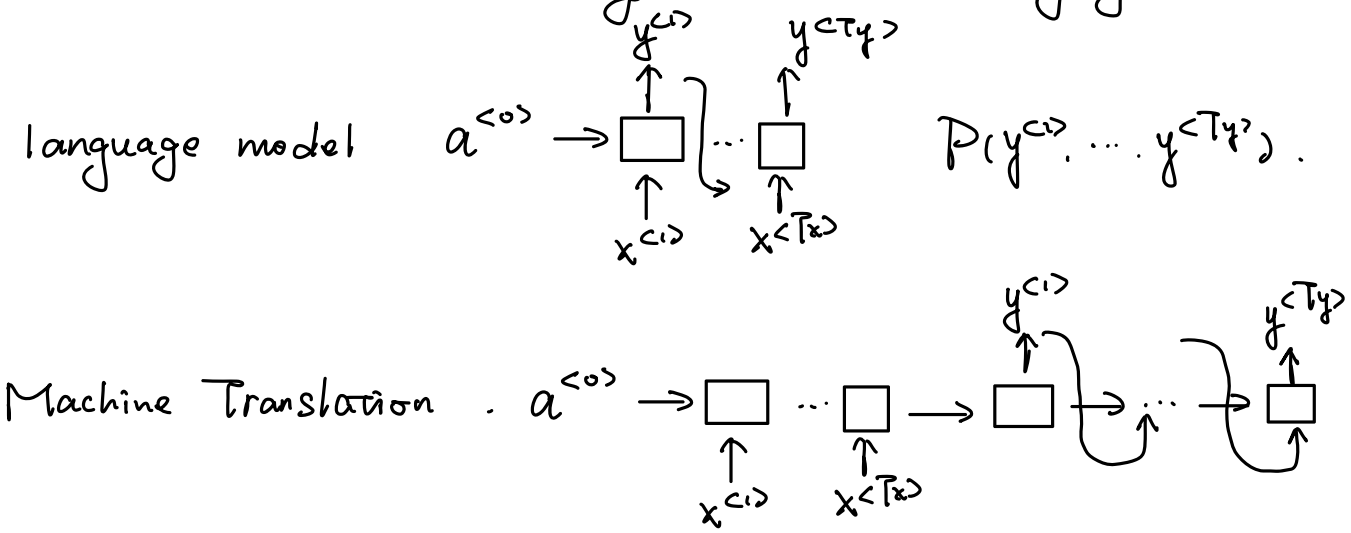
Need enough pair of English - French sentences.

Image Captioning.



Picking the Most Likely Seq.

Machine translation as building a conditional language model.



The decoder part really looks like the language model.

But "conditional" on the encoder part. $P(y^{<1>}, \dots, y^{<T_y>} | x^{<1>} \dots x^{<T_x>})$.

Finding the most likely translation.

$P(y^{<1>}, \dots, y^{<T_y>} | x)$

English. French

Jane visite l'Afrique en septembre

→ Jane is visiting Africa in september.

→ Jane is going to be visiting Africa in sept.

→ In sept. Jane will visit Africa.

} argmax $P(y^{<1>} \dots y^{<T_y>} | x)$.

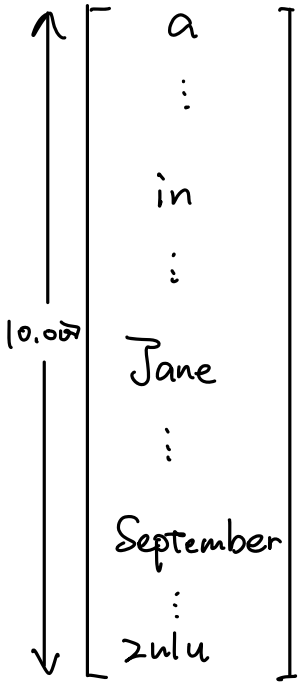
Why not greedy search. $P(y^{<1>}|x)$

Pick the most likely first word. Then most likely second, and ...

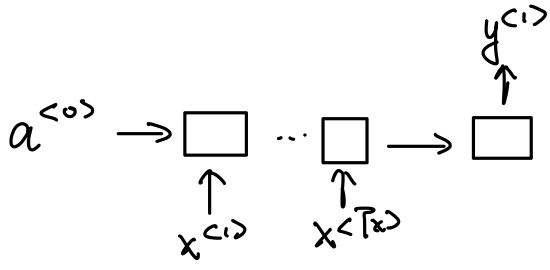
We need a search algorithm instead of listing all the sentences.

Beam Search.

Step 1.



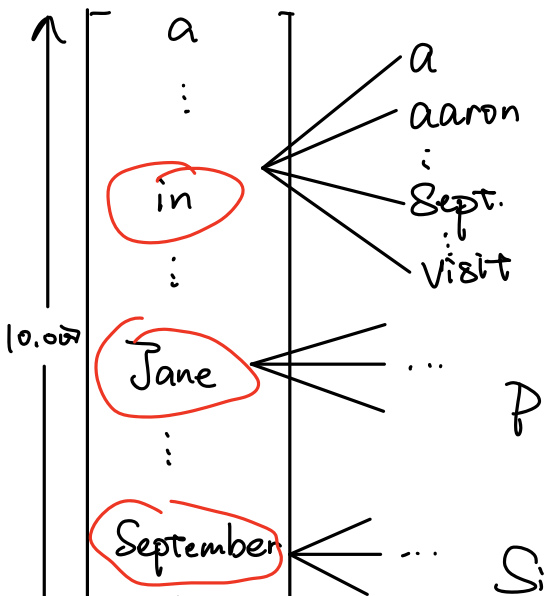
e.g. B = 3 (beam width).



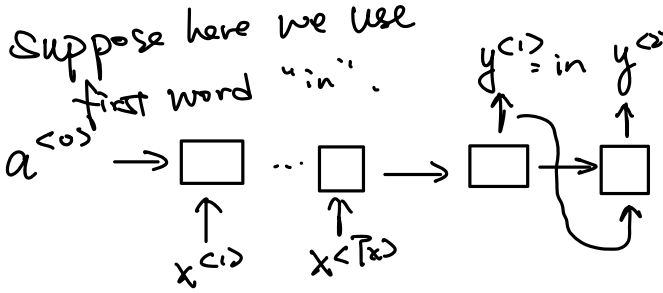
First thing. generate the most likely first words.

Instead of greedy pick the most likely. but have a param B to get the number Bs of most likely here. suppose we choose "Jane" "in" "September"

Step 1. Step 2.



Suppose here we use first word "in".



$P(y^{<2>}|x, \text{"in"})$.

Actually we want the most likely pair.

$$P(y^{<1>}, y^{<2>}|x) = P(y^{<1>}|x)P(y^{<2>}|x, y^{<1>}).$$

Since we use B=3. Step 2 we need to consider all

zuiu

30,000 possibilities. And we pick the most likely 3.

e.g. Say: "in sept.". "Jane is". "Jane visits".

Also. we reject "Sept." as the first word.

Everytime we have number of B of copies of the network. And we do through search over all vocab. And extend network. Cont.

e.g. Step three: "in sept. Jane". "jane is visiting". "Jane visits Africa"

Finally we may end up with "Jane visits Africa in Sept.".

Refinement to Beam Search.

Length Normalization.

$$\underset{y}{\text{argmax}} \prod_{t=1}^{T_y} \underbrace{P(y^{<t>} | x, y^{<1>} \dots y^{<t-1>})}_{\text{This number always smaller than 1.}}.$$

So we consider: (first optimization).

$$\underset{y}{\text{argmax}} \sum_{y=1}^{T_y} \log P(y^{<t>} | x, y^{<1>} \dots y^{<t-1>}).$$

Notice: This equation tends to prefer short sentences!

So we add a normalize term.

$$\underset{y}{\text{argmax}} \frac{1}{T_y^{\alpha}} \sum_{y=1}^{T_y} \log P(y^{<t>} | x, y^{<1>} \dots y^{<t-1>}.)$$

Where α is a hyperparam you can tune. ① if $\alpha=1$. normalizing over length. ② $\alpha=0$. No normalization.

Beam Search discussion.

Beam width B:

① large B. Better result, slower

② Small B. worse result. faster.

in practice. we use 10. production 100. Not common for 1000↑

Remember. Beam search is not guaranteed to find the exact max!

Error Analysis on Beam Search.

Example:

Jane visite l'Afrique en septembre

Human: Jane is visiting Africa in september (y^*)

Alg: Jane visited Africa last Sept. ($\hat{y}$).

Increasing the Beam Width barely hurts.

RNN computes $P(y|x)$.

What you can do. is use the RNN model. to compute $P(y^*|x)$ and $P(\hat{y}|x)$. This help you to identify the error.

Case ① $P(y^*|x) > P(\hat{y}|x)$. (human perform better).

Beam Search choose $\hat{y}$. But y^* attains higher $P(y|x)$.

Conclusion: Beam Search is at fault.

Case ②. $P(y^*|x) < P(\hat{y}|x)$.

y^* is better translation than $\hat{y}$. But RNN predicted $P(y^*|x) < P(\hat{y}|x)$

Conclusion: RNN model is at fault.

Error Analysis Process.

Human.	Alg (Give sth. wrong)	$P(y^* \mid x)$	$P(\hat{y} \mid x)$.	Attention.
Jane visits Africa in Sept.	Jane visited Africa last Sept.	2×10^{-10}	1×10^{-10}	B.

Continue go through all the bad example the Alg. returned.
We can figure out what fraction of errors are "due to" beam search V.S. RNN model.

Bleu Score. How to evalute when there are multiple good results?

French: Le chat est sur le tapis.

Reference ①: The cat is on the mat.
Reference ②. There is a cat on the mat.
} Both correct.

Bleu: Bilingual evaluation. understudy.

extreme e.g. MT output: the the the the the the. the. (7 words)

Precision: $\frac{7}{7}$. whether the words in the output appear in reference.

Which is not a good evaluation.

Modified precision: "the" appear in ① two time. in ② one time.

So the upper bound is 2.

$\frac{2}{7}$ ← Count clip("the") in reference.
← Count ("the") in output

Bleu score on bigram.

Reference ①: The cat is on the mat.

Reference ②. There is a cat on the mat

MT output: The cat the cat on the mat. ($\hat{y}$)

Possible bigram.	Count	Count Clip.
"the cat"	2	1

"cat the"	1	0
"cat on"	1	1
"on the"	1	1
"the mat"	1	1

Modified precision: $\frac{\sum \text{Count Clip}}{\sum \text{Count}} = \frac{4}{6}$.

Bleu score on unigram.

$$P_1 = \frac{\sum_{\text{unigram} \in \hat{y}} \text{Count clip}(\text{unigram})}{\sum_{\text{unigram} \in \hat{y}} \text{Count}(\text{unigram})}$$

Bleu score on n-gram.

$$P_n = \frac{\sum_{\text{n-gram} \in \hat{y}} \text{Count clip}(\text{n-gram})}{\sum_{\text{n-gram} \in \hat{y}} \text{Count}(\text{n-gram})}.$$

This allows you to evaluate the "overlap" of references and the predicted output.

Bleu Details.

P_n = Bleu score on n-gram only. e.g. P_1, P_2, P_3, P_4.

Combine Bleu score: $BP\left(\exp\left(\frac{1}{4}\sum_{n=1}^{4} P_n\right)\right)$

BP = brevity penalty. Very short output, easy get high score.

This penalize short output.

$$BP = \begin{cases} 1 & \text{if MT output length} > \text{reference length}. \\ \exp(1 - \text{MT output length}/\text{reference length}) & \text{otherwise}. \end{cases}$$

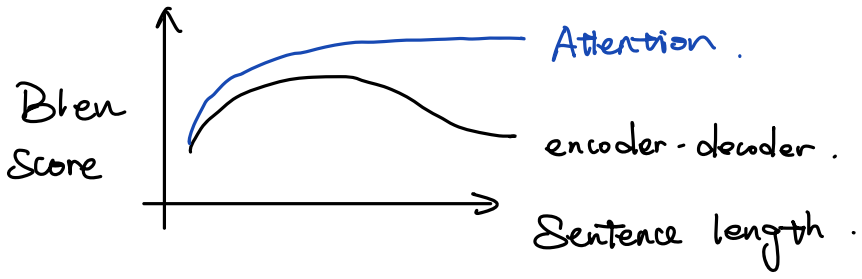
Attention model intuition.

The problem of long seq.

Suppose we have a long French sentence. and want to translate to English.

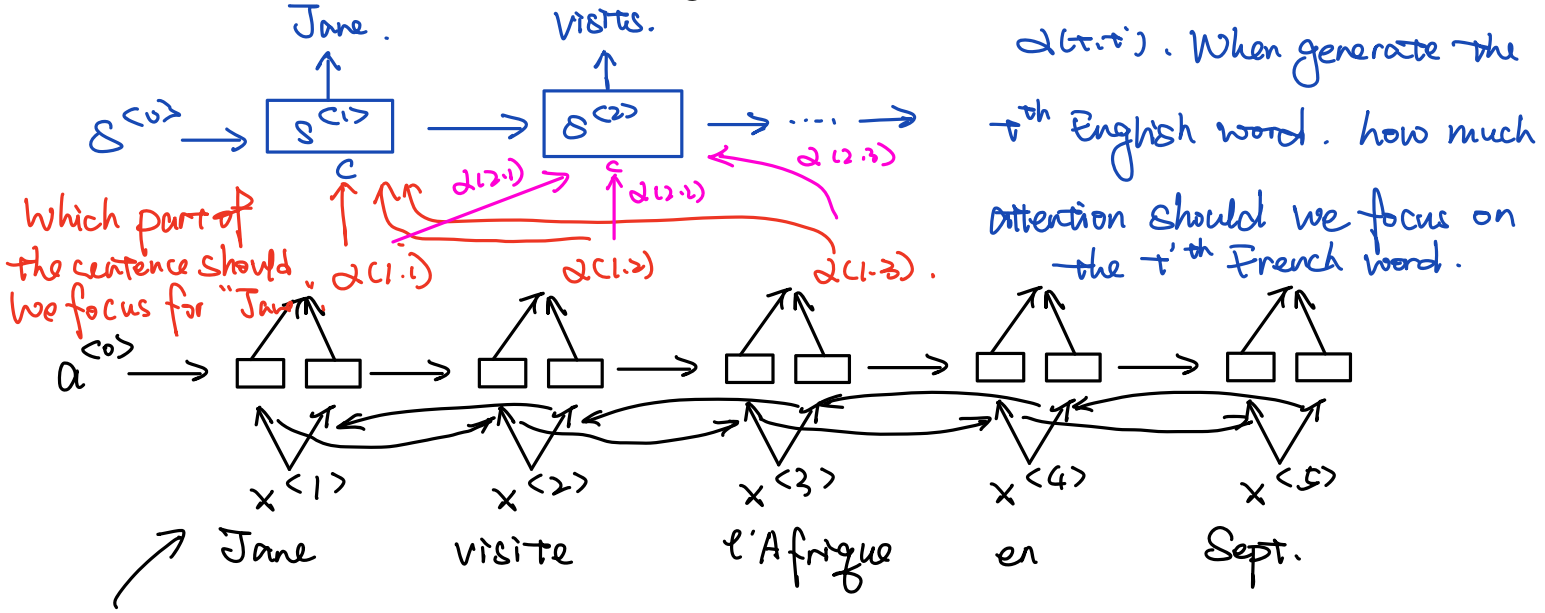
What human translation did is translate part by part. But current

encoder - decoder model need to memorize it all.



Attention model intuition. Cillustrate with short sentence).

And we use another RNN to generate translation.



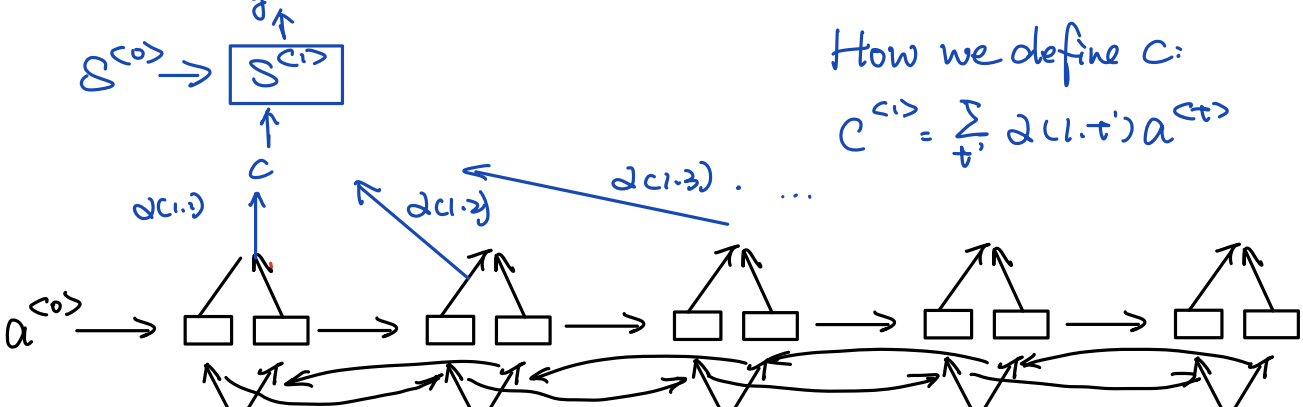
Suppose we use a BRNN (May be BiLSTM. BiGRU).

Attention model gives a set of weight. how much you should focus.

e.g. first word "Jane". Combine all the attention weights. $\alpha(1,1) \sim \alpha(1,5)$

Also there're weights for $\alpha(2,1)$. $\alpha(3,3)$ ....

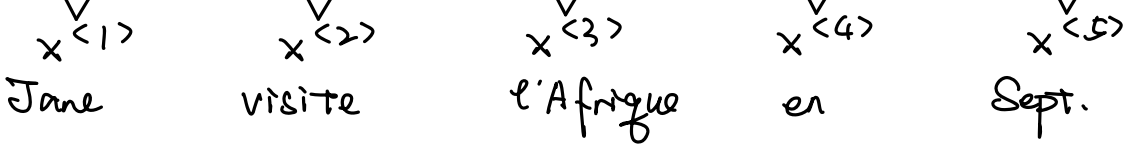
Attention Model.



we notice: $\sum_{t'} \alpha(1,t') = 1$.

How we define c:

$c^{<1>} = \sum_{t'} \alpha(1,t') a^{<t>}$



At every timestamp, use $a^{<t>} = (\vec{a}^{<t>}, \overleftarrow{a}^{<t>})$

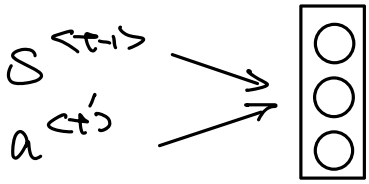
forward, Backward.

Computing attention $\alpha^{<t,t'>}$. (Same as $\alpha(t,t')$ above).

$\alpha^{<t,t'>}$ = amount of attention $y^{<t>}$ should pay to $a^{<t'>}$

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$

how to get e? Use a small network.



it seems like $e^{<t,t'>}$ should depend on $s^{<t-1>}$, $a^{<t'>}$. How? Don't know

So we use GD.

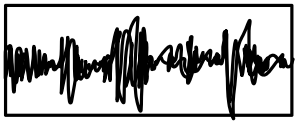
Downside of this Alg: Runtime: $(T_x \cdot T_y)$. Quadratic cost.

Speech recog.

Problem.

x → y

audio clip → transcript.



→ "The quick brown fox".

Airpressure across time.

we used to use phonemes. de kwik broun.

basic sound units:

We can use Attention model for speech recog.

CTC cost for speech recog.

(Connectionist temporal classification)