Course 5

Sequence Model

Sequence Model

Sequence Model.

X: "Harry Potter" and "Hermione Granger" invented a spell

$x^{<1>}$ $x^{<2>}$ ... $x^{<9>}$ $T_x = 9$

y: 1 1 0 1 1 0 0 0

$y^{<1>}$ $y^{<2>}$ ... $y^{<9>}$ $T_y = 9$

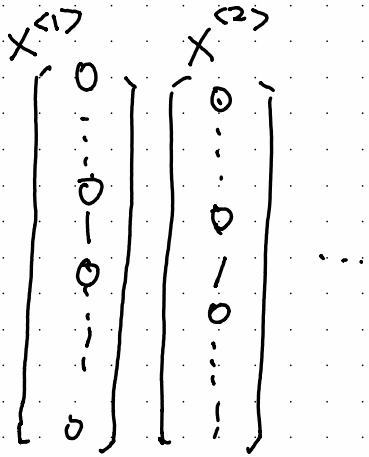
ith training example

$X^{(i)<t>}$ $T_x^{(i)}$

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

Representation

a
aaron 1
and 2
harry 4075
potter 6830
zulu
10,000



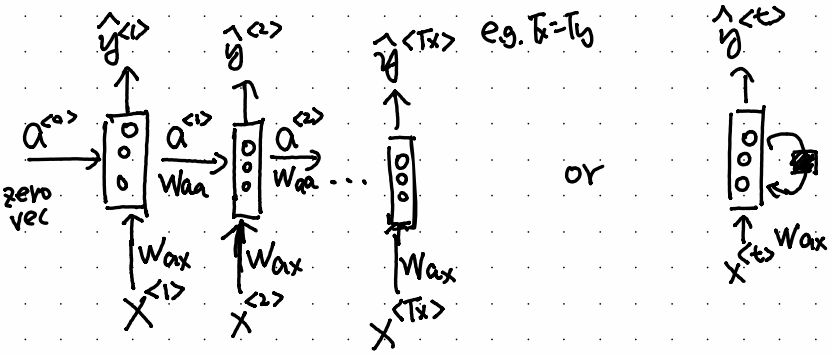
one-hot.

~30,000 for commercial software

Recurrent NN

why not standard network

- input / output could be of different length
- doesn't share features across different pos.



limitation: only use info "before" t △ B-RNN

$\vec{a}^{<0>} = \vec{0}$

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

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

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

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

$$a^{\langle t \rangle} = g(W_{aa} a^{\langle t-1 \rangle} + W_{ax} x^{\langle t \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = g(W_{ya} a^{\langle t \rangle} + b_y)$$

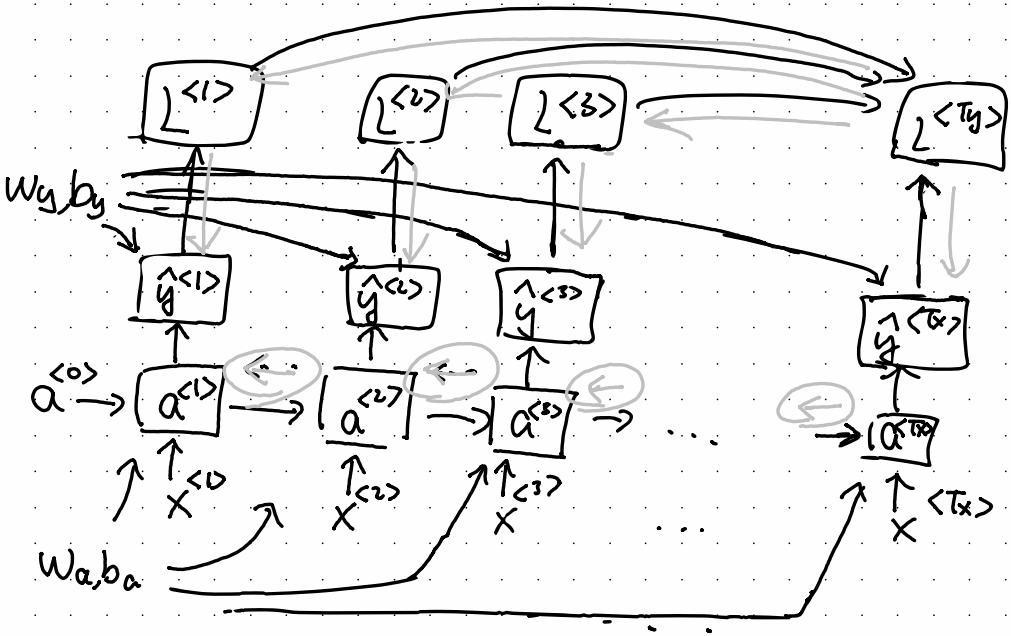
⇓ simplify

$$a^{\langle t \rangle} = g(W_a [a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_a)$$

$$\underset{100}{\left[\underset{100}{W_{aa}} \vdots \underset{10000}{W_{ax}} \right]} = \underset{(100,\,10100)}{W_a}$$

$$\begin{bmatrix} a^{\langle t-1 \rangle} \\ \hline x^{\langle t \rangle} \end{bmatrix} \begin{matrix} \updownarrow 100 \\ \updownarrow 10{,}000 \end{matrix} \quad \updownarrow 10{,}100$$

Backprop. through time

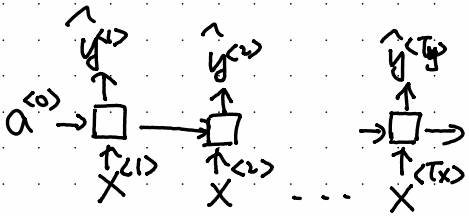


$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>}\log\hat{y}^{<t>} - (1-y^{<t>})\log(1-\hat{y}^{<t>})$$

$$L(\hat{y}, y) = \sum_{t=1}^{T_x} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Different RNN types

$T_x = T_y$

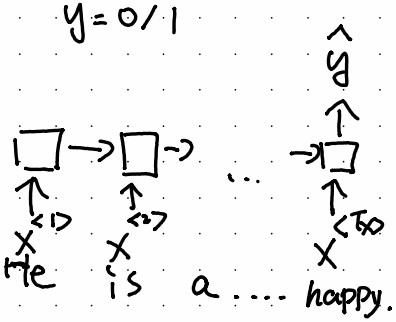


many-to-many

e.g. sentiment detection

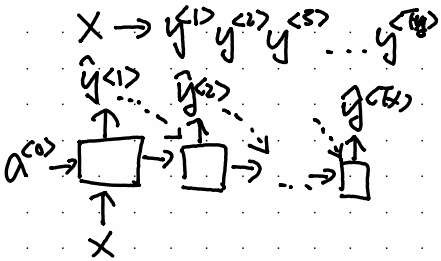
X = text

y = 0/1



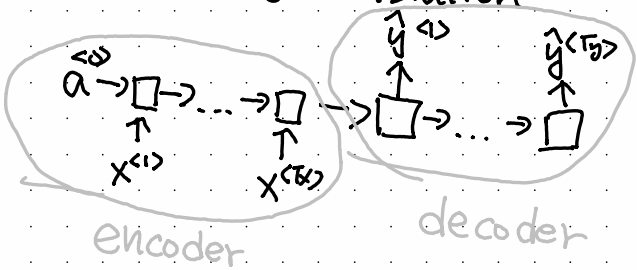
many-to-one

music generation



one-to-many

machine translation



many-to-many

Language Model

Speech Recognition

the apple and pair salad

the apple and pear salad ← higher p, although both sound the same

language model: $P(y^{<1>} \dots y^{<T_y>})$?

Training Set: large corpus of text.

- An apple a day ... doctor away. <EOS>

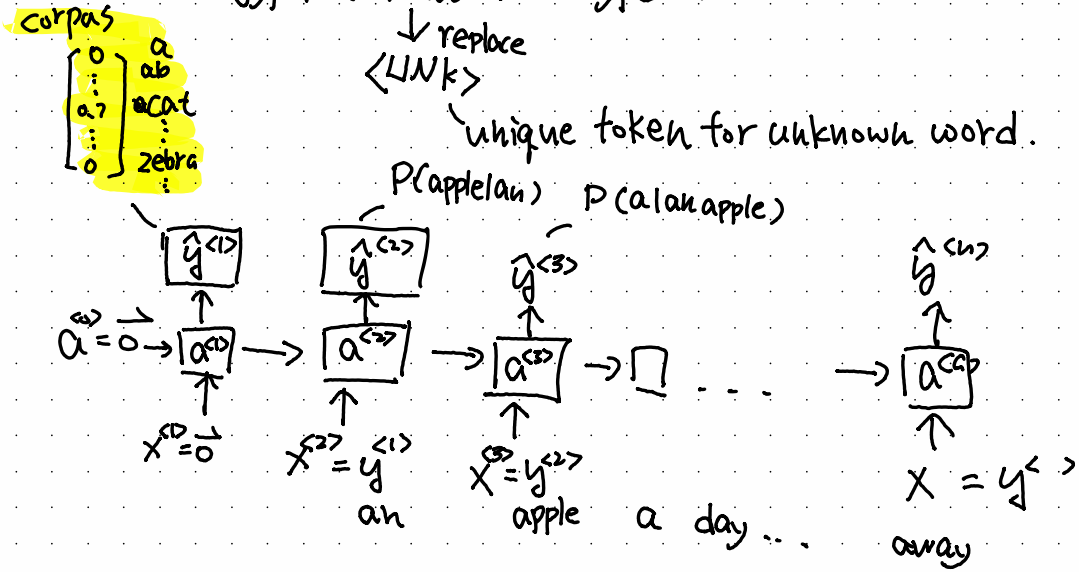
  $y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $y^{<n>}$ $y^{<>}$ Tokenize

- The Egyptian Mau is a type of cat

  ↓ replace

  <UNK>

  unique token for unknown word.



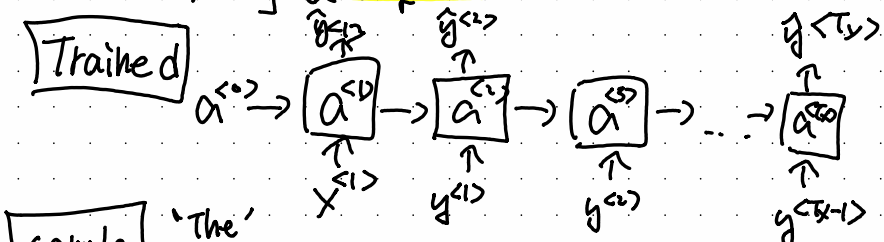
$P(y^{<1>}, y^{<2>}, y^{<3>})$

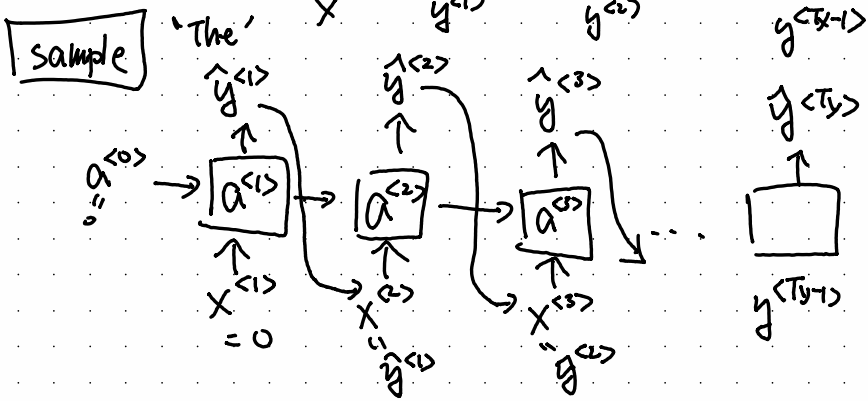
$= P(y^{<1>}) \cdot P(y^{<2>} | y^{<1>}) \cdot P(y^{<3>} | y^{<1>}, y^{<2>})$

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

Sampling a seq from trained RNN

- generate random sentence



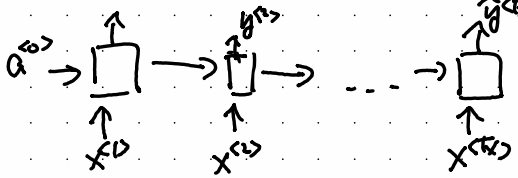


until EOS or length limit

< np.random.choice >

- can also do "character-level" language model
  - very long
  - slower to train
- word level model is more popular.

Vanishing gradients with RNN



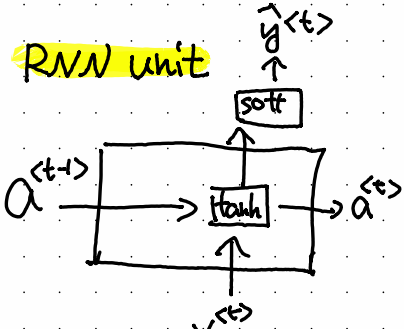
e.g. cats, which ..... , are cute
is cute

- RNN cannot memorize long-term influence

exploding gradient - solve with clipping

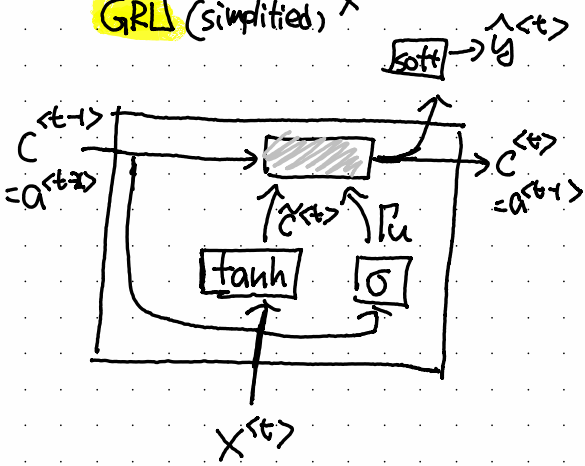
GRU - Gated Recurrent Unit 2014

RNN unit



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

GRU (simplified)



C = memory cell

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

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

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

Γ_u: update (gamma)

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

the cat, which ate ... is full

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

Γ_u = 1 0, 0 . . . . , 1

GRU (Full) — simpler, and can build bigger model

$\tilde{h}$ $\tilde{c}^{\langle t \rangle} = \tanh\left(W_c[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c\right)$

u $\Gamma_u = \sigma\left(W_u[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u\right)$

r $\Gamma_r = \sigma\left(W_r[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_r\right)$ — relevance

h $c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1-\Gamma_u) * c^{\langle t-1 \rangle}$

$c^{\langle t \rangle} = a^{\langle t \rangle}$

LSTM, 1997 — more complex, but more powerful.

$\tilde{c}^{\langle t \rangle} = \tanh\left(W_c[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c\right)$

$\Gamma_u = \sigma\left(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u\right)$ — update

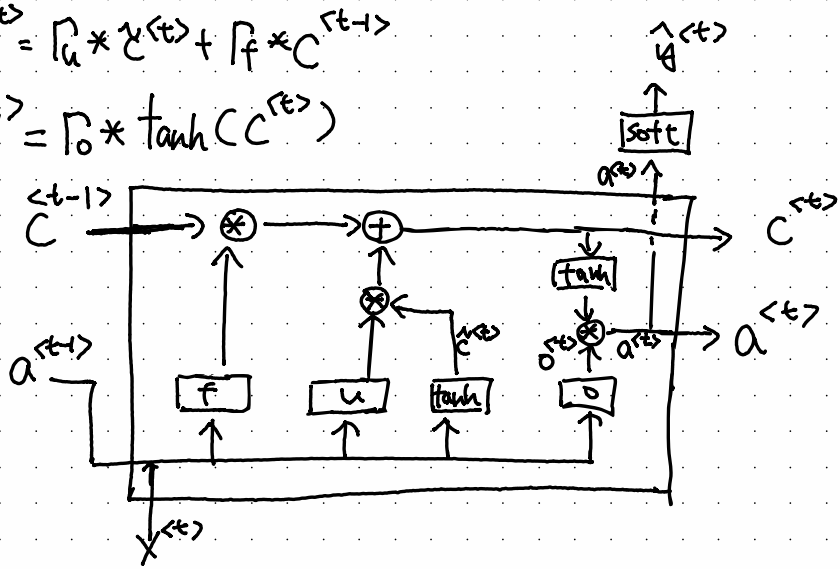
$\Gamma_f = \sigma\left(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f\right)$ — forget

$\Gamma_o = \sigma\left(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o\right)$ — output

- peephole connection: $c^{\langle t-1 \rangle}$ affects $\Gamma_u, \Gamma_f, \Gamma_o$

$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + \Gamma_f * c^{\langle t-1 \rangle}$

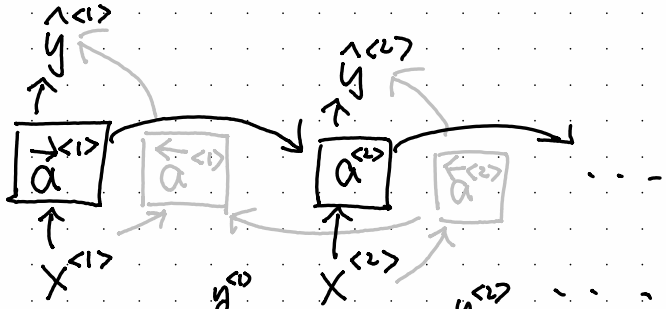
$a^{\langle t \rangle} = \Gamma_o * \tanh(c^{\langle t \rangle})$



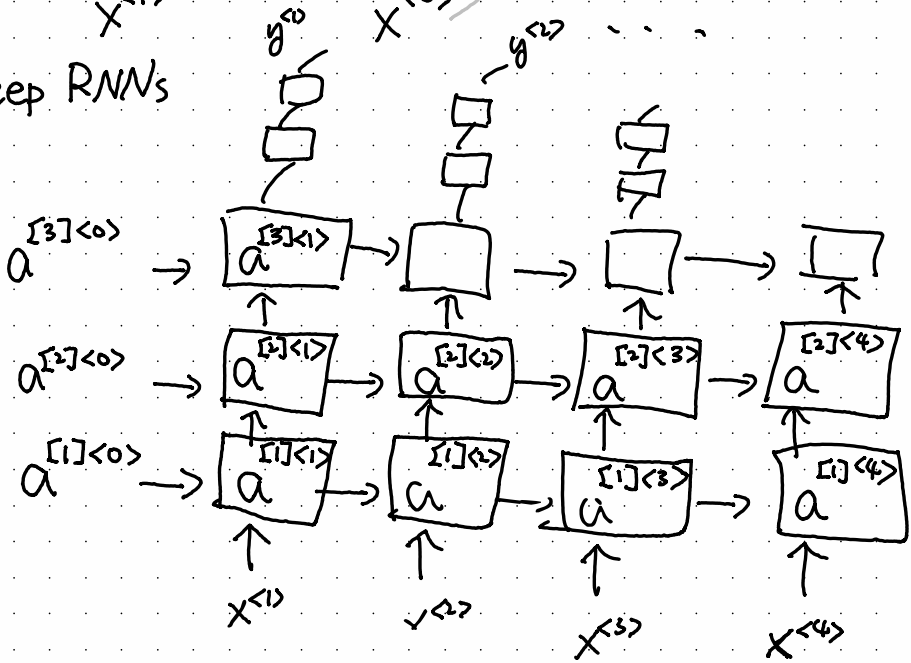
Bidirectional RNN (BRNN)

getting info from future

He said Teddy bears are cute
He said Teddy Roosevelt was a good man



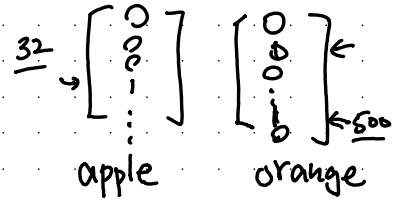
Deep RNNs



$$a^{[2]\langle 3\rangle} = g\left(W_a^{[2]}\left(a^{[2]\langle 2\rangle}, a^{[1]\langle 3\rangle}\right) + b_a^{[2]}\right)$$

Word Embeddings

one-hot representation



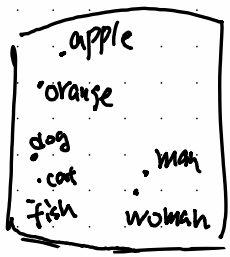
← dot pndct is 0, doesn't reflect semantic distance

featurized representation: word embedding

	man	woman	apple	orange	...
gender	-1	1	0	0	
age	0.02	0.03	0.02	0.03	
size	0.4	0.4	0.01	0.02	
food	0.01	0.02	0.95	0.94	
e.g. 300D	:	:	:	:	

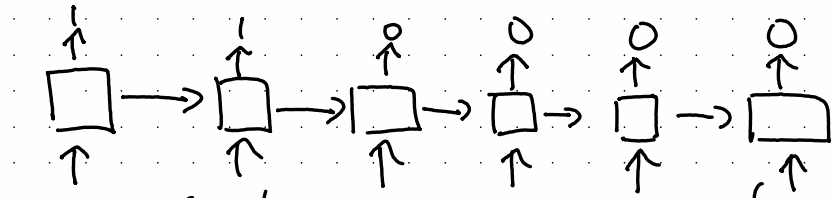
Visualize word embedding t-SNE

embed 300D to 2D for viz



"embed" words into feature space

ex: Name entity recognition



Harvey Smith is an orange farmer

Robert Lee is a durian cultivar

↑ name not coporate name

↖ what if we have fruit not trained before?

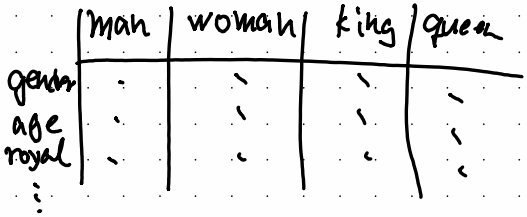
Transfer Learning

1. learn from large text corpus
   (or download pre-trained embedding online)
2. transfer embedding to new tasks w. smaller set
3. (optional) continuously finetune word embedding

properties of word embedding, 2013 Mikolov

e.g. man → woman

king → ?



$e_{man} - e_{woman} \approx e_{king} - e_?$

find word w

$$\arg\max_w \underline{\underline{sim}}(e_w, e_{king} - e_{man} + e_{woman})$$

30~75% accuracy in papers.

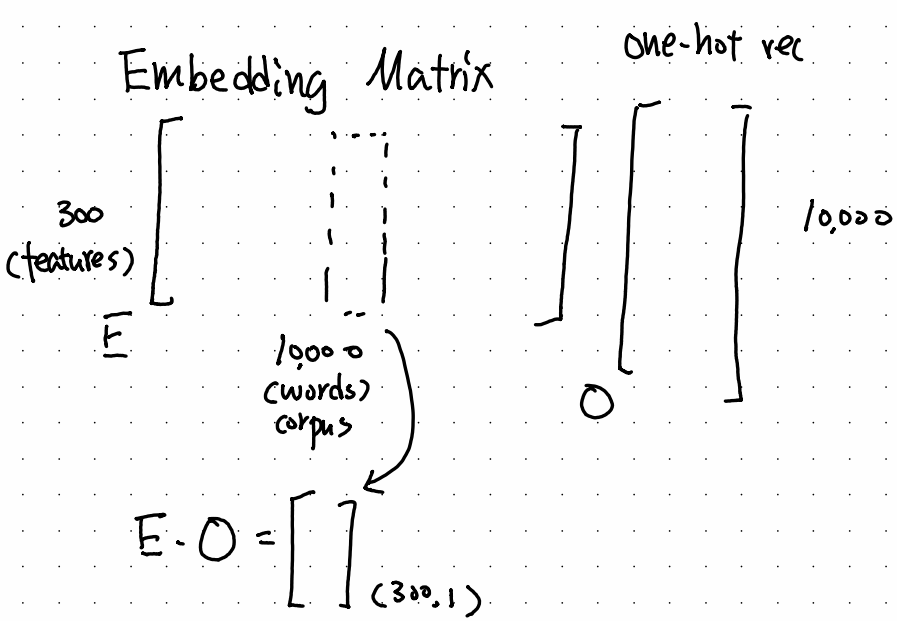
Cosine similarity

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

man : woman = boy : girl

RMB : China = Rupee : India

⋮



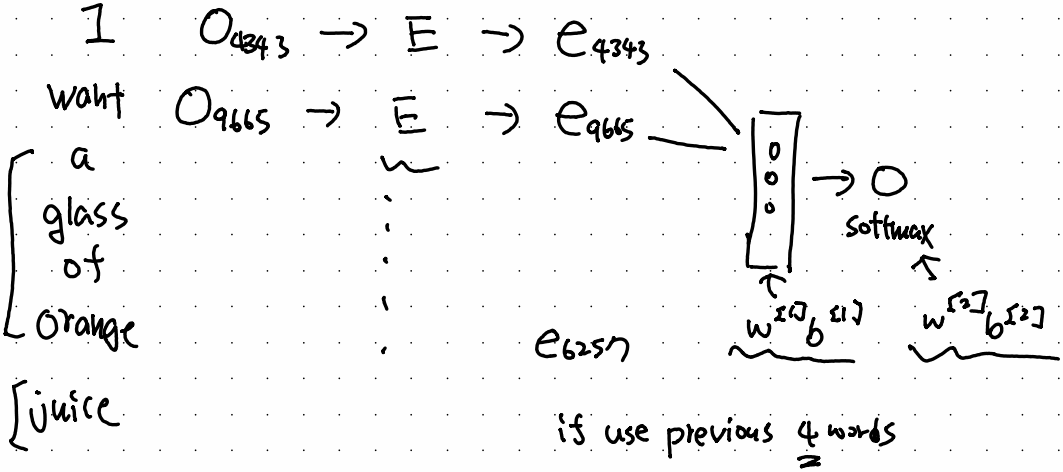
$E \cdot O_j = e_j$ - embedding for word j

(in practise, API extract the column)

Learning Word Embeddings

history: difficult → simple

Neural Language Model, 2003, Bengio



if use previous 4 words

$4 \times 300 = 1200$ dim

learn E, $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}$

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

↑ target

Context:

last 4 words

last / next 4 words

last 1 word

"nearby" 1 word (skip gram)

Word2Vec, 2013, Mikolov

skip-gram

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

context	target
orange	juice
orange	glass
orange	my

randomly chose

e.g. vocab size = 10000.

Model

$x \rightarrow y$

Context c ("orange") $\rightarrow$ Target t ("juice")

$O_c \rightarrow E \rightarrow e_c \rightarrow o \xrightarrow{\text{softmax}} \hat{y}$

θ_t: parameter associates with output t

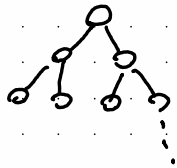
i.e. target t being the label

$$\text{Softmax: } p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10000} e^{\theta_j^T e_c}}$$

slow. so,

$$\mathcal{L}(\hat{y}, y) = -\sum_j y_i \log \hat{y}_i \qquad y = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \end{bmatrix}$$

hierarchical softmax



how to sample context c?

too common → the, a, of, ...

→ orange, apple...

$p(c)$ based on frequency

Negative Sampling

2013, Mikolov

problem: are context/word together.

give x, predict y

context (x)	word (x)	target (y)	
orange	juice	1	← context in data
orange	king	0	← random chosen from dict
orange	book	0	k
orange	of	0	k
↑ c	↑ t	↑ y	

k=5~20 small data

k=2-5 larger

Model

softmax:

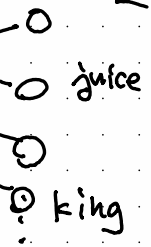
$$P(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_j e^{\theta_j^T e_c}}$$ ← 10,000 softmax, slow!

$$P(y=1|c,t) = \sigma(\theta_c^T e_c)$$

orange: O_{6257}

$O_{6257} \rightarrow E \rightarrow e_{6257}$ ← 10,000 binary classification

each iteration, train k+1 classifier



sampling:

freq $P(w_i)$

uniform $\frac{1}{|v|}$

$$P(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum f(w_i)^{\frac{3}{4}}}$$ heuristic

GloVe Word, 2014. Pennington.
(global vector)

C, t

X_{ij} = # times j appear in context of i $\quad X_{ij} = X_{ji}$

$$\min \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij}) \left(\theta_i^T e_j + b_i + b_j' - \log X_{ij} \right)^2$$

(t: θ_i^T, c: e_j)

weighting

$f(X_{ij}) = 0$ if $X_{ij} = 0$ $\quad 0 \log 0 = 0$

θ_i, e_j are symmetric

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

this, is..
durian
zulu

note:

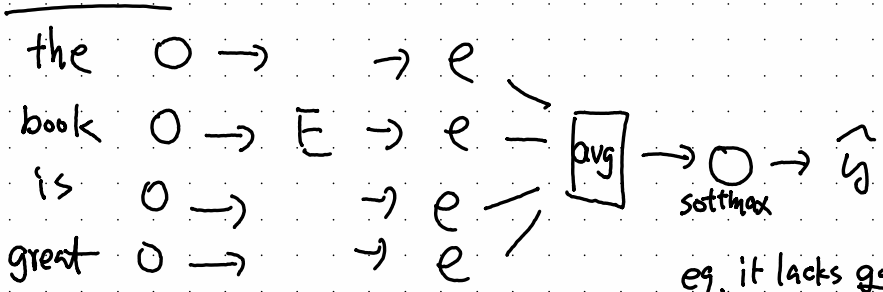
~~e_i~~ might not align with defined interpretable feature axis of the embedding matrix

sentiment classification

x	y
the book is great	★★★★
the book lacks depth	★
reads ok	★★
⋮	

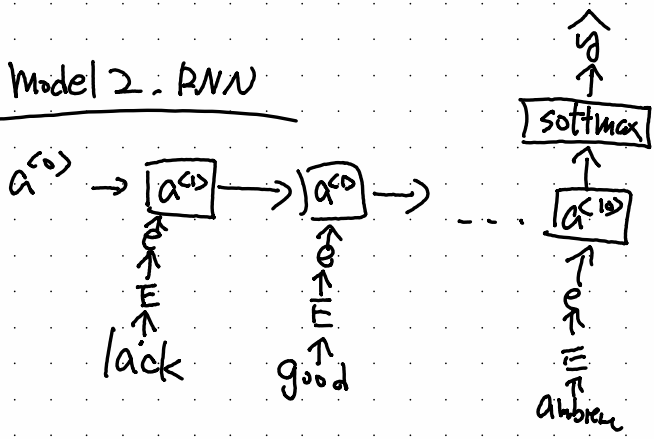
data set is smaller.

model 1.



eg. it lacks good insight, good depth, good taste ... problem!

Model 2. RNN



many-to-one

Debiasing Word Embeddings, 2016.

Man : ComputerProgrammer as Woman : Homemaker

×

word embedding reflect gender, ethnicity used to train the model.

1. identify
   identify bias axis
2. neutralize
   project onto axis except definitive terms
3. equalize
   move definitive pairs to same distance around axis

Seq to Seq model 2014

e.g. translation

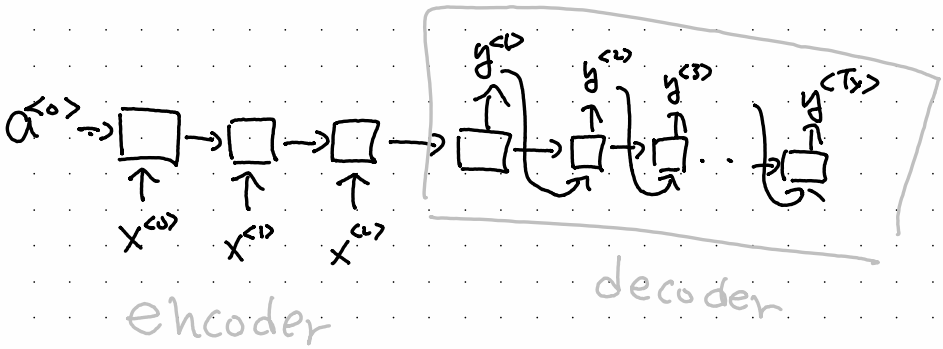
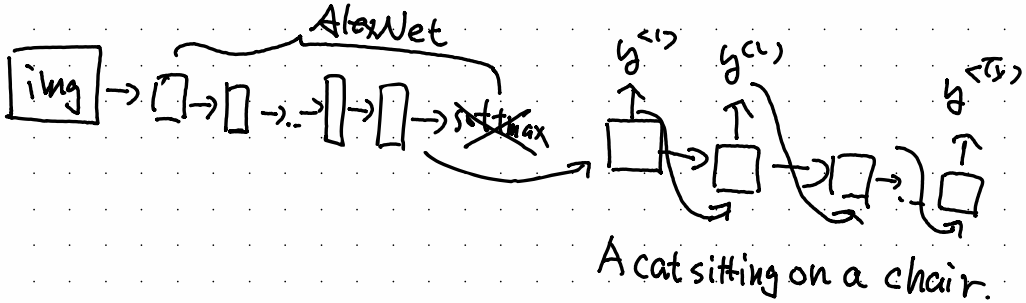


Image Captioning, 2014, 2015

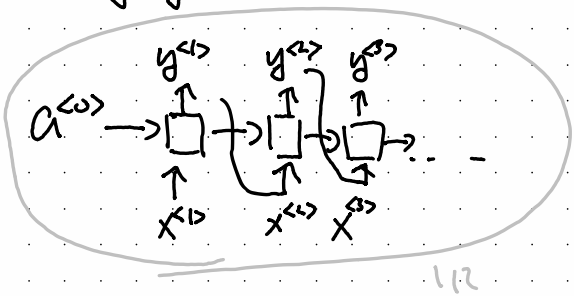


seq2seq vs language model

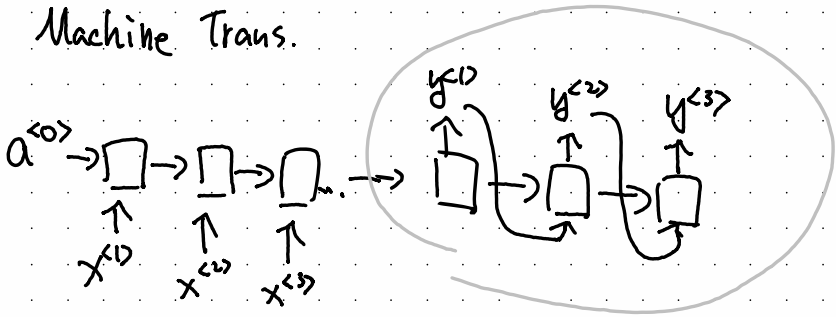
machine trans as building conditional language model.

Language Model:

$P(y^{<1>}, \ldots y^{<T_y>})$



Machine Trans.



conditional language model $P(y^{<1>} \ldots y^{<T_y>} | x^{<1>} \ldots x^{<T_x>})$

e.g. $P(\text{Eng} \mid \text{Franch})$

$\underset{y^{<1>} \ldots y^{<T_y>}}{\text{argmax}} P(y^{<1>} \ldots y^{<T_y>} | x)$

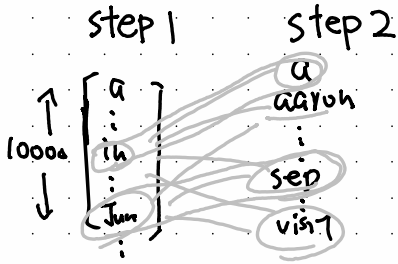
French sentence

(beam search

△ greedy search usually result in worse sentence

e.g. picking common word like "going" "the"... etc

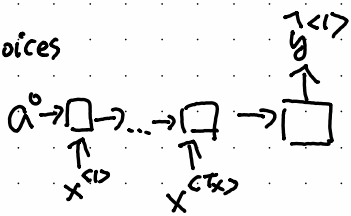
Beam Search



B=3 beam width ⇒ keep track of top B choices

step 1 $P(y^{<1>}|x)$

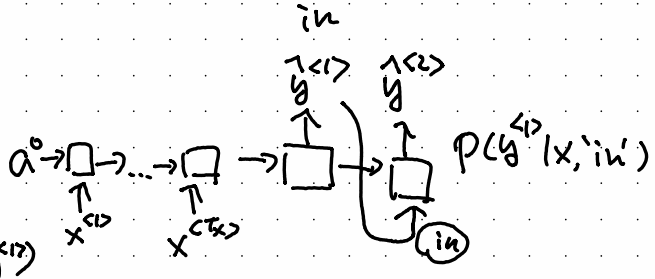
memorize 3 good choices

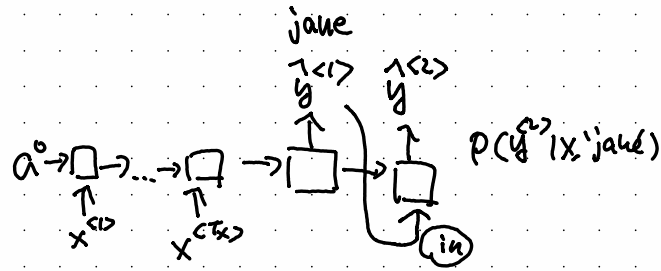


step 2

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

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

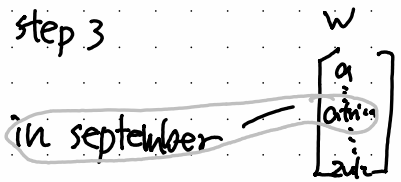




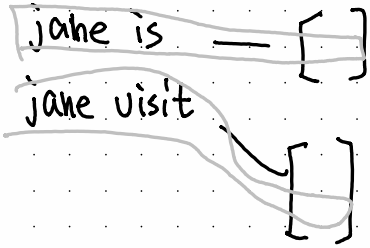
beam width

eventually pick B=3 most likely output then moves on to next step.

step 3



keep B NNs.



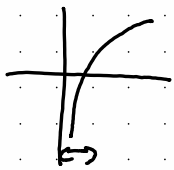
B=1 ~ beam search
"greedy" search.

Refinement: Length normalization

$$\underset{y}{\text{argmax}} \prod_{t=1}^{T_y} P(y^{<t>} | X, y^{<1>} \dots y^{<t-1>})$$

- multiplication of p minimize value
  → tend to pick short sentence

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



$$\Rightarrow \frac{1}{T_y^{\alpha}} \sum_{t=1}^{T_y} \log P(y^{<t>} | X, y^{<1>} \dots y^{<t-1>})$$

$\alpha = 0.7$

---

large B → slow, better
small B → fast, faster.

production B~10
research : B~100~1000
to squeeze performance

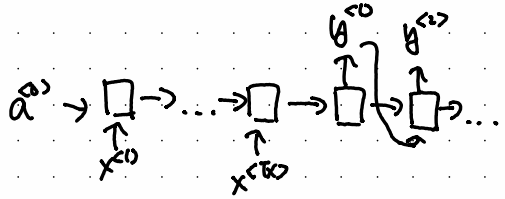
△ compared to BFS, DFS, beam search doesn't gurantee best solution
- heuristic search

Error analysis in Beam Search.

Jane visite l'Afrique en septembre

Human: Jane visit Africa in September (y^*)

Machine: Jane visit Africa last September ($\hat{y}$)



use RNN to compute

$P(y^*|x)$, $P(\hat{y}|x)$

Case 1.

$P(y^*|x) > P(\hat{y}|x)$

beam search can improve

Case 2.

$P(y^*|x) \leq P(\hat{y}|x)$

RNN model is at fault.

→ y^* is better, but RNN says $\hat{y}$ is better, thus RNN is at fault.

↑

do the above analysis for all dev set,

then decide what to do next. - increase B

or

update RNN

Bleu score: bilingal evaluation understudy, 2002
- deal with equally good translations

Bleu: Unigram
Sentence: Le chat est sur le tapis

ref 1 : the cat is on the mat
ref 2 : There is a cat on the mat.
⋮

clipping # occurrance

MT output: the the the ...

precision: $\frac{7}{7}$ modified: $\frac{2}{7}$ ← $Count_{clip}$('the') / Count('the')

Bleu score on bigrams

MT: The cat the cat on the mat.

	Count	$Count_{clip}$
the cat	2	1
cat the	1	0
the cat	1	1
cat on	1	1
on the	1	1
the mat		

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

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

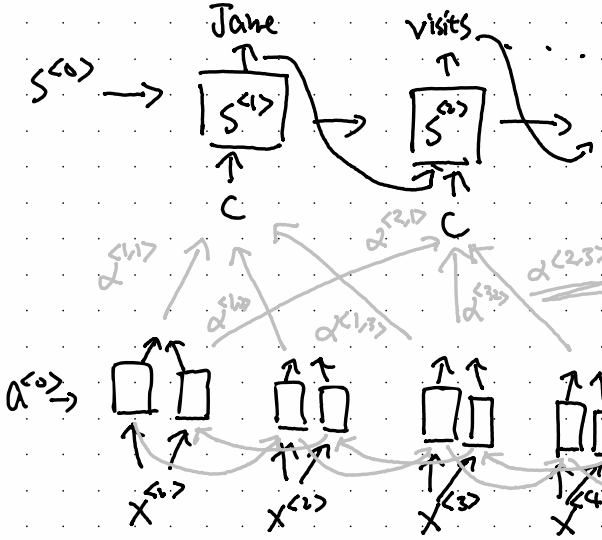
Combined: $P = \exp\left(\frac{1}{n}\sum_{i=1}^{n} P_i\right)$

BP. brief penality → penalize short trans

$$BP = \begin{cases} 1 & \text{if MT output longer than human} \\ e^{\left(1 - \frac{human\_len}{MT\_len}\right)} \end{cases}$$

Attention Model, 2014

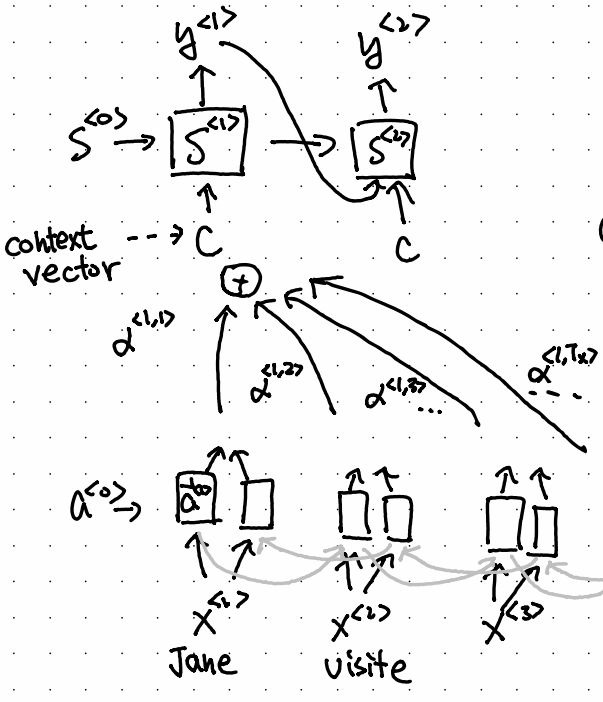
- cannot handle long sequences



$\alpha^{<t,t'>}$

attention weight.

BRNN



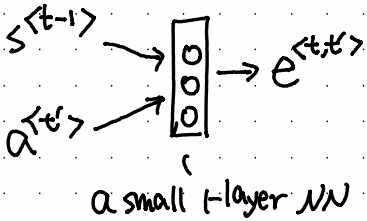
$$a^{<t'>} = (\vec{a}^{<t'>}, \overleftarrow{a}^{<t'>})$$

$$\sum_{t'} \alpha^{<1,t'>} = 1$$

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

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

$$\alpha^{<t,t'>} = \frac{e^{e^{<t,t'>}}}{\sum_{t'=1}^{Tx} e^{e^{<t,t'>}}}$$ <- similar to softmax



runtime

$Tx \cdot Ty$ parameters $\alpha^{<t,t'>}$

usually Tx, Ty are not huge.

2015, image captioning, -> pay attention to a part of the picture

speech recognition

x (audio) → y (transcript)

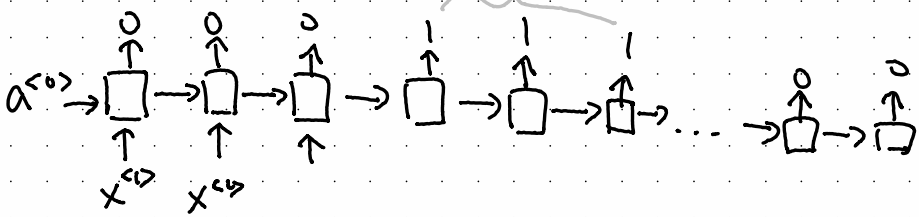
data: 300h ~ 3000h

commercial: 100,000 h

- attention model works well
- CTC cost [2006]
  connectionist temporal classification ttt_h_eee__ _ _ qqq_ _
  △ Tx >> Ty, e.g. 1000 Hz in audio → the quick brown fox ...
  → collapse repeated chars not separated by "blank"

trigger word detection

multiple 1 for balanced data set.

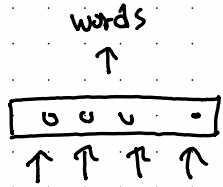


Transformer. 2017

RNN → GRU → LSTM

con:
- increase complexity
- sequential

transformer:
- attention + CNN
  - self-attention. $A^{<1>} A^{<2>} \dots$
  - multi-head attention. multiple copies



---

self-attention

transformer attention

$A(q, K, V)$ – attention-based vector representation

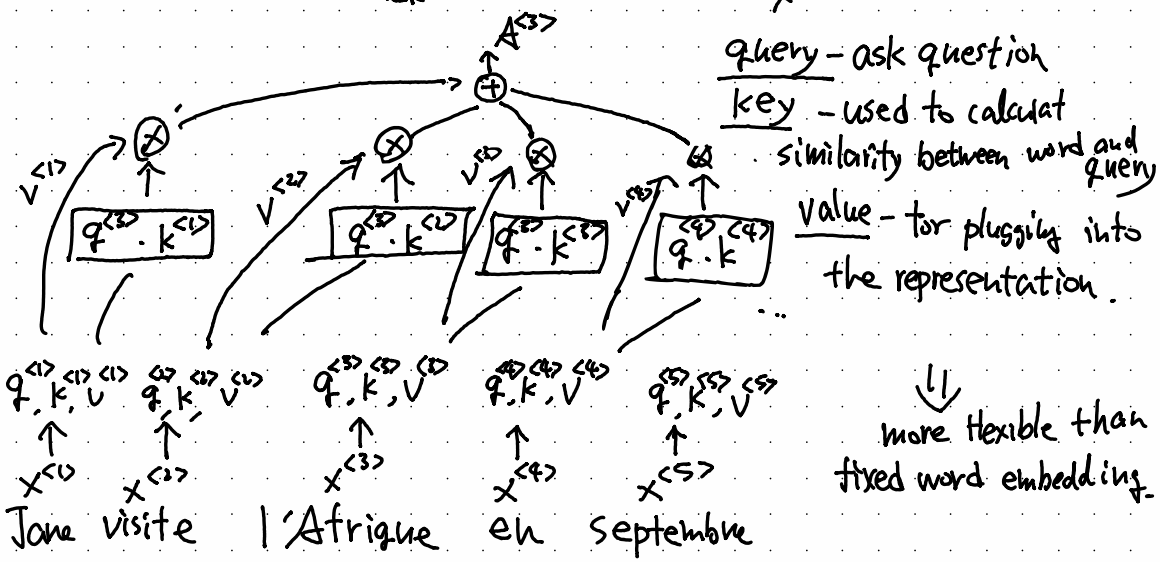
$$= \sum_i \frac{e^{q \cdot k^{<i>}}}{\sum_j e^{q \cdot k^{<j>}}} v^{<i>}$$

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$q^{<3>} = W^Q x^{<3>}$$

$$k^{<3>} = W^K x^{<3>}$$

$$v^{<3>} = W^V x^{<3>}$$



query – ask question

key – used to calculat similarity between word and query

value – for plugging into the representation.

more flexible than fixed word embedding.

Multi-head Attention

for example

#heads = 8 (usually)

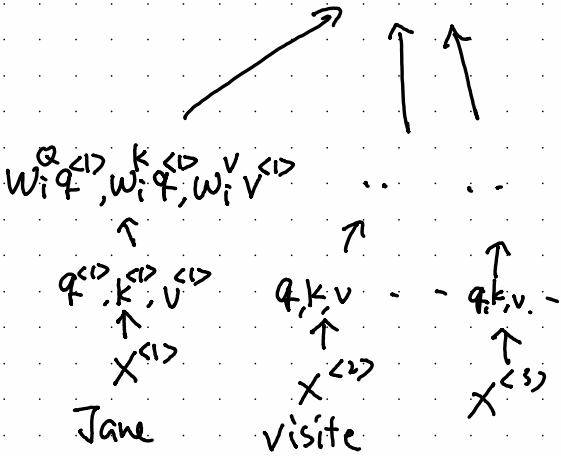
head1 W_1^Q, W_1^K, W_1^V – what's happening

head2 W_2^Q, W_2^K, W_2^V – when?

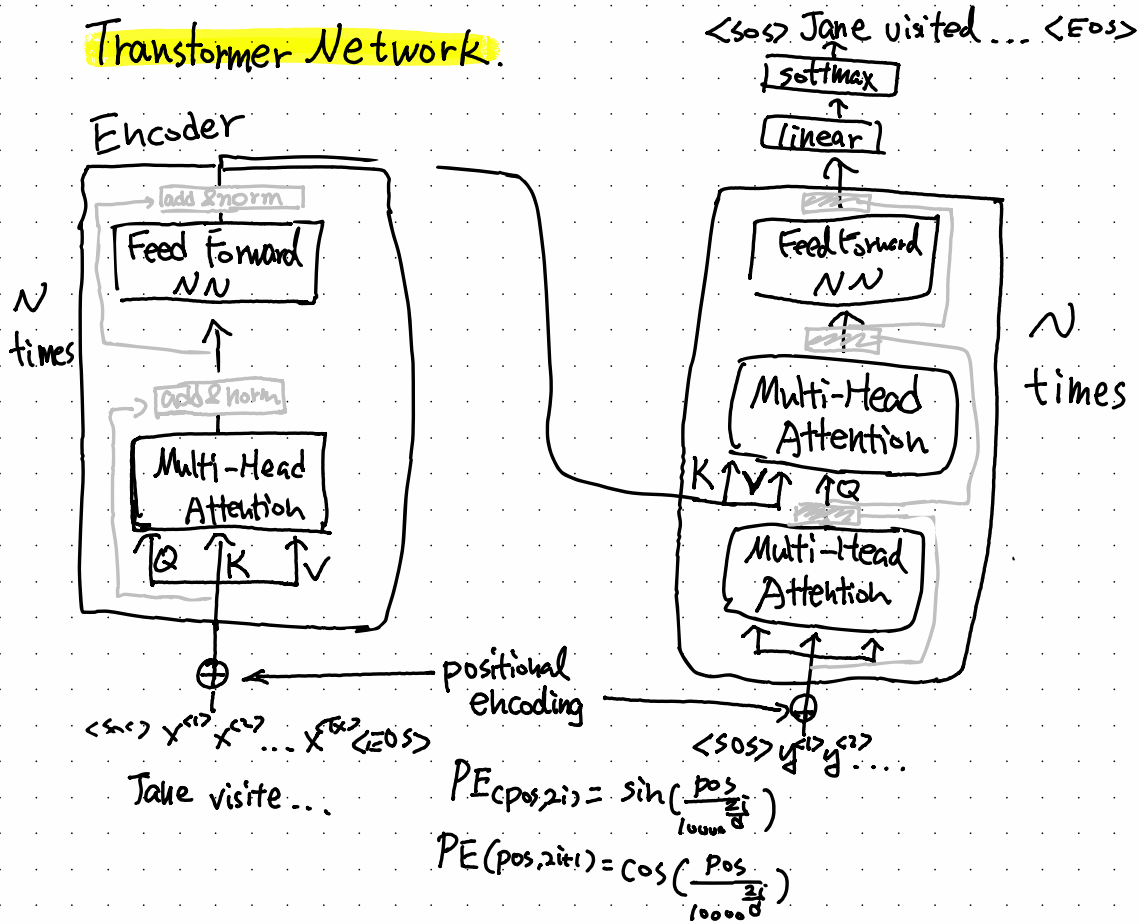
⋮ ⋮ – how?

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1, \text{head}_2, \text{head}_3)W_0$$

$$\text{Attention}(W_i^Q Q, W_i^K K, W_i^V V)$$



Transformer Network



$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

masked multi-head attention

- for training.