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

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

↳ temporal

→ y: 1 1 0 1 1 0 0 0 0

$T_x = 9$

T_x → length of the input sequence

T_y → length of the output sequence

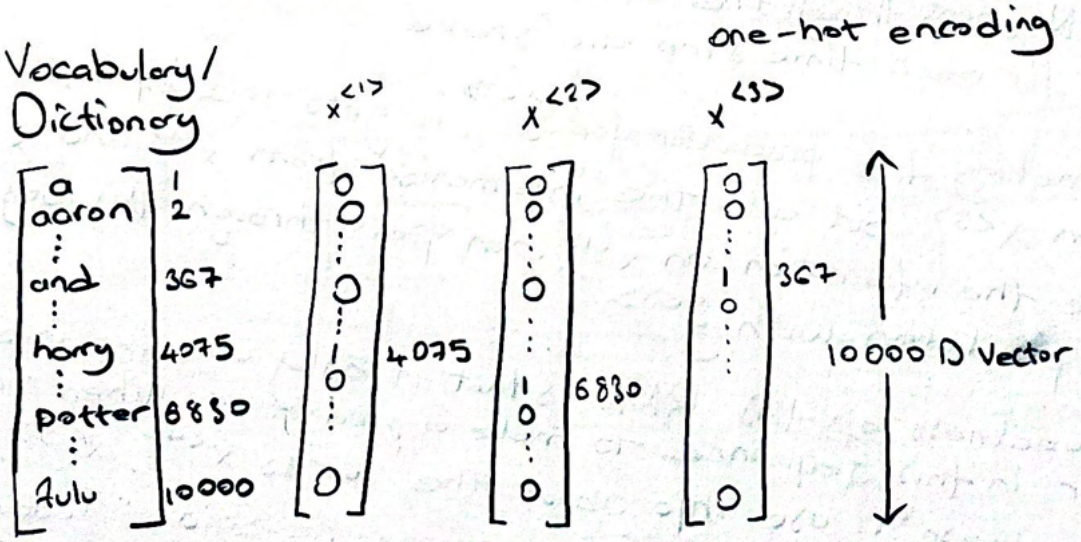
$x^{(i)}$ → i^{th} training example

$x^{(i)<t>}$ → t^{th} element in the sequence of training example i

$T_x^{(i)}$ → input sequence length for training example i

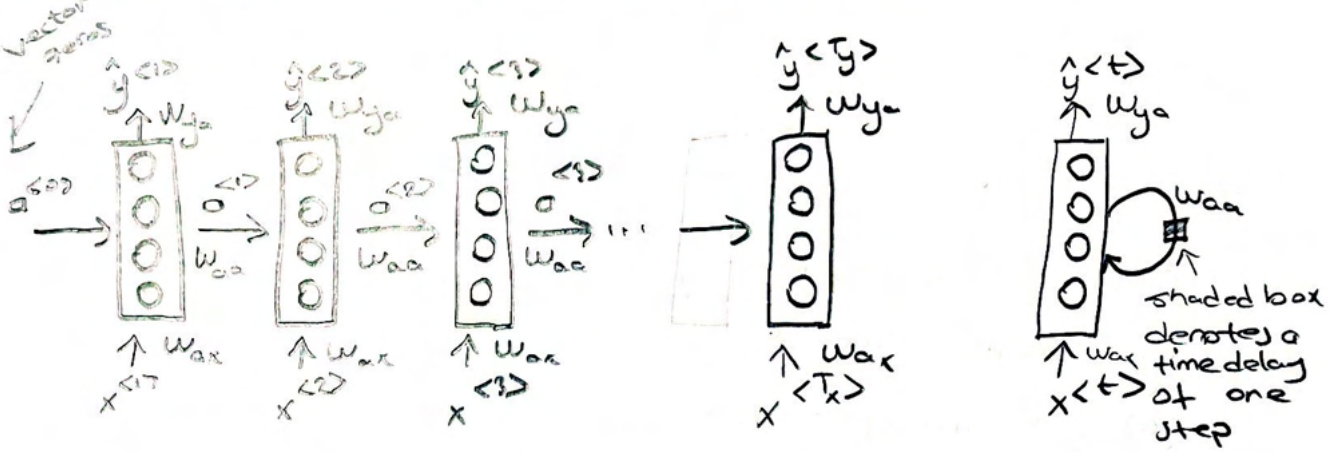
$y^{(i)<t>}$ → t^{th} element in the output sequence of the i^{th} training example

$T_y^{(i)}$ → length of the output sequence in the i^{th} training example



The goal is given this representation for X to learn a mapping using a sequence model to then target output y

RECURRENT NEURAL NETWORKS



Why not a standart network?

- Inputs, outputs can be different length in different examples
- A naive NN architecture doesn't share features learned across different positions of text.

The RNN scans through the data from left to right. The parameters it uses for each time step are shared

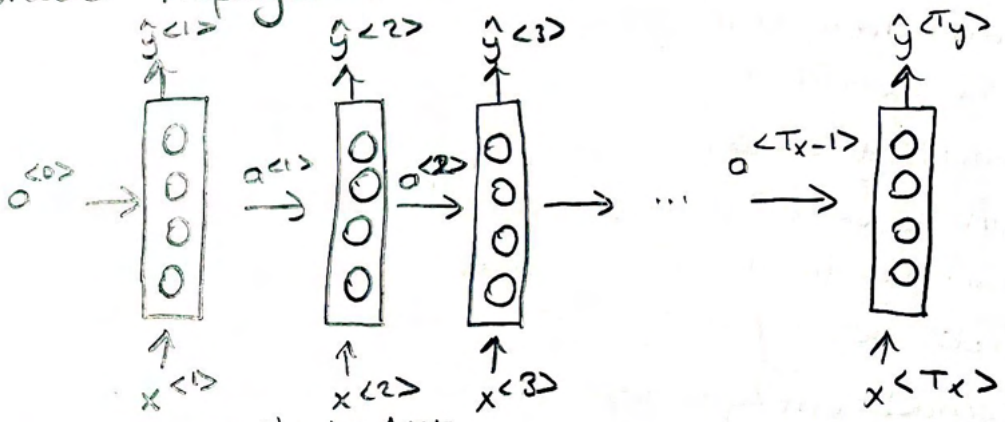
When making the prediction for $\hat{y}^{<3>}$ it gets the information not only from $x^{<3>}$ but also the information from $x^{<1>}$ and $x^{<3>}$ because the information on $x^{<1>}$ can pass through this way to help to prediction with $\hat{y}^{<3>}$

One weakness of this RNN is that it only uses the info that is earlier in this sequence to make a prediction. When predicting $\hat{y}^{<3>}$, it doesn't use info about the words $x^{<4>}$, $x^{<5>}$, $x^{<6>}$ and so on. If you're given a sentence He said, "Teddy Roosevelt was a great President" in order to decide whether or not the word Teddy is a part of a person's name it would be really useful to know not just info from the first two words but to know info from the later words in the sentence as well. The sentence could also have been He said "Teddy bears are on sale!"

Bidirectional RNN (BRNN)

Unidirectional RNN

Forward Propagation $a \leftarrow W_{ax} x^{<1>}$



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

activation fnct

$a^{<1>} = g_1(W_{aa}\, a^{<0>} + W_{ax}\, x^{<1>} + b_a)$ ← tanh | ReLu

$\hat{y}^{<1>} = g_2(W_{ya}\, a^{<1>} + b_y)$ ← sigmoid | softmax

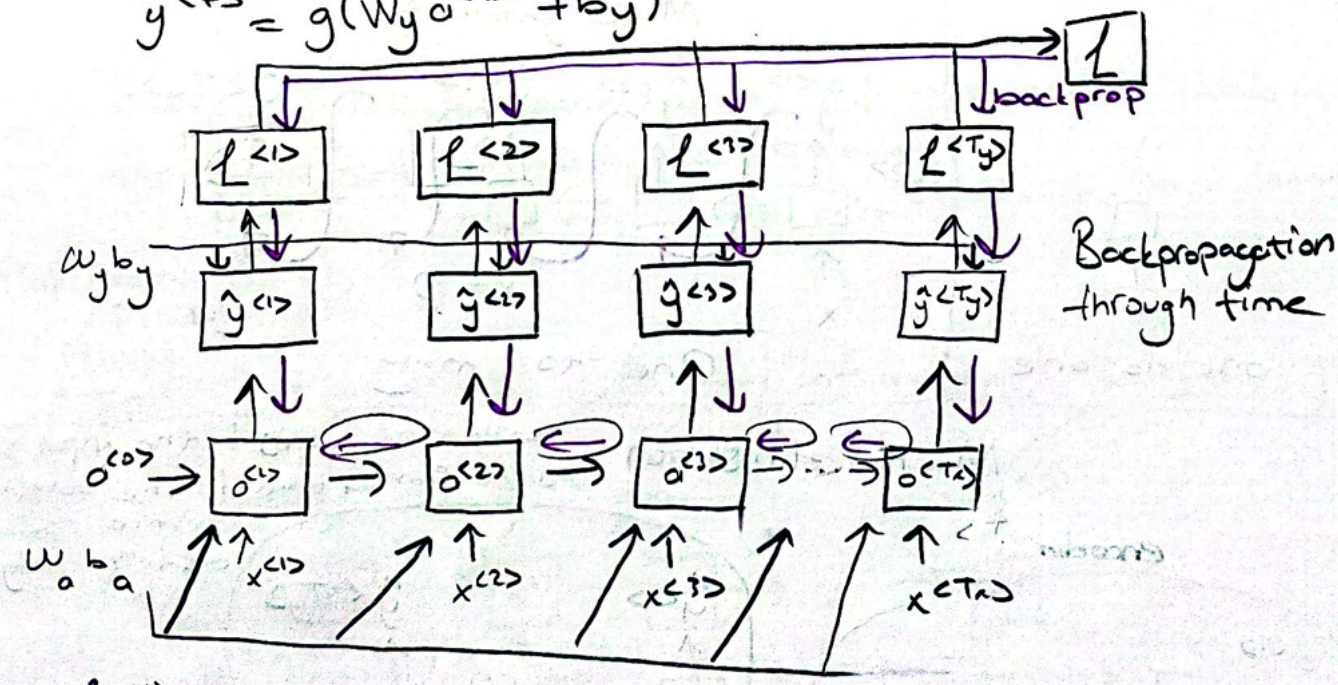
$a^{<t>} = g(W_{aa}\, a^{<t-1>} + W_{ax}\, x^{<t>} + b_a)$

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

tanh prevents vanishing gradient

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

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

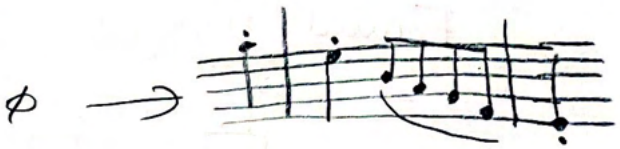


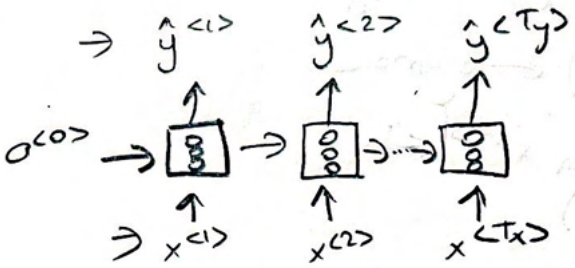
$$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log \hat{y}^{<t>} - (1 - y^{<t>}) \log(1 - \hat{y}^{<t>})$$

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

Examples of sequence data

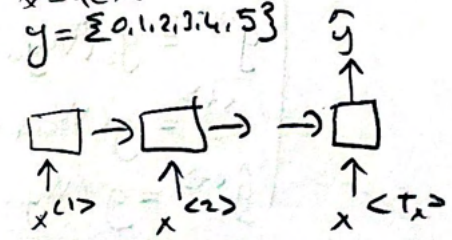
Speech recognition
Music generation
Sentiment classification
DNA sequence analysis
Machine translation
Video activity recognition
Named entity recognition





Many to many
(many inputs many outputs)

Sentiment classification
x = text
y = {0,1,2,3,4,5}

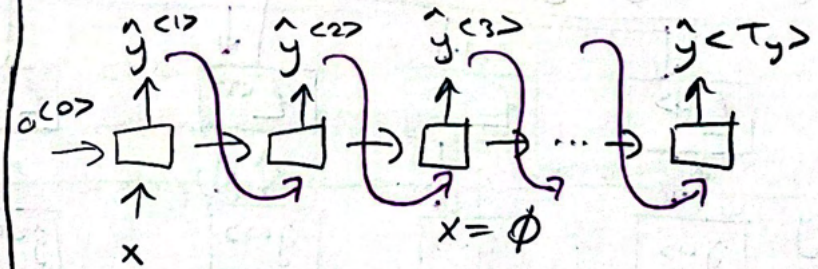


There is ... movie

Many to one

one to one

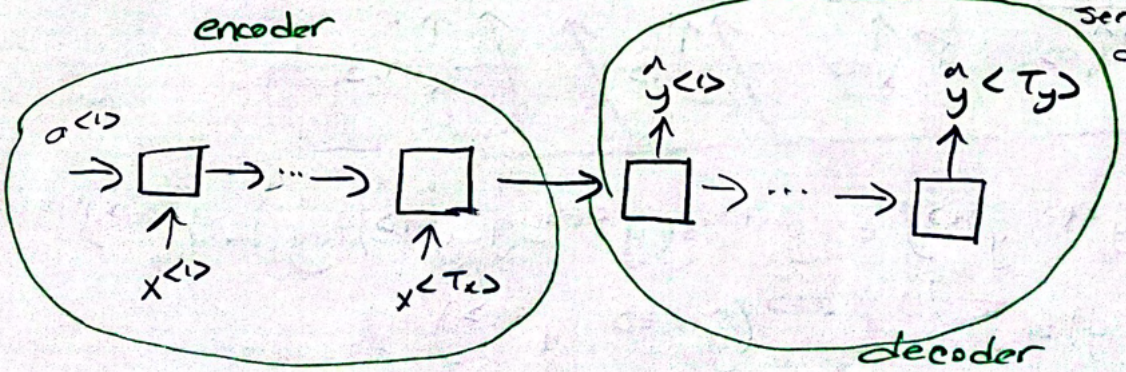
Music generation



One to many

Machine translation

number of words in the input sentence and the num of words in the output sentence could be different length



Many to many

LANGUAGE MODELING

Speech Recognition

The apple and pair salad.
The apple and pear salad.
P(The apple and pair salad) = 3.2×10^{-13}
P(The apple and pear salad) = 5.7×10^{-10}

$P(y^{<1>}, y^{<2>}, \cdots, y^{<T_y>})$

Language Modeling With an RNN

Training set: large corpus of english text.

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

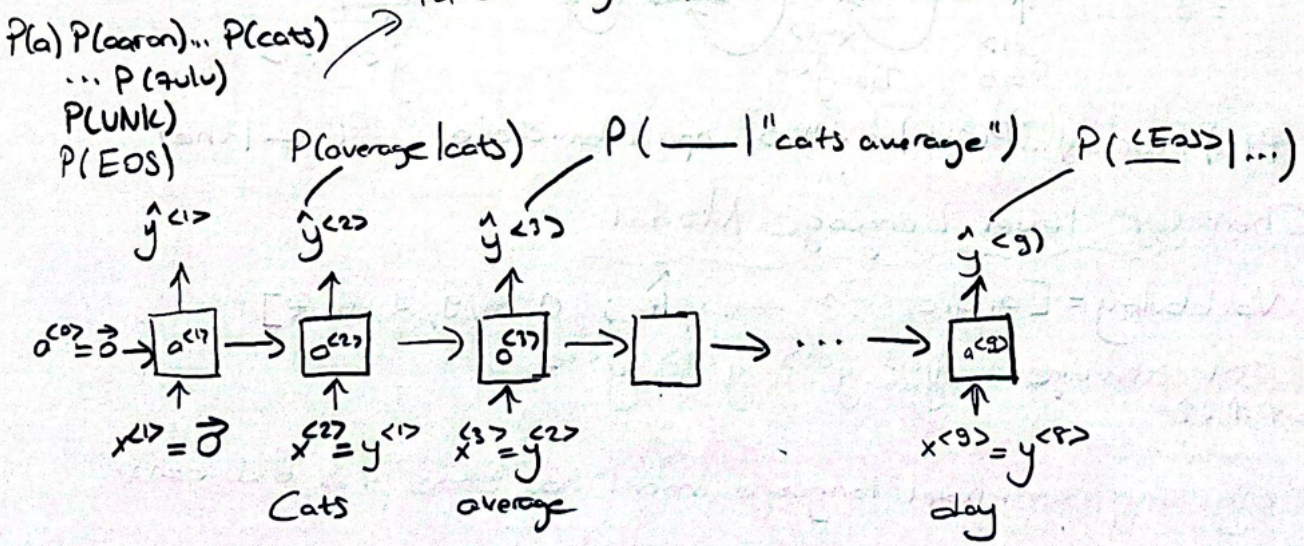
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $y^{<8>}$ $y^{<9>}$

$x^{<t>} = y^{<t-1>}$

* The Egyption ~~Mau~~ is a bread of cat. <EOS>
<UNK>

<UNK> stands for unknown words. If some of the words in the training set are not in the vocabulary

10,002 way softmax output (Vocabulary length = 10000)



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

What $a^{<1>}$ does is it will make a softmax prediction to try to figure out what is the probability of the first word y.

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

$\hat{y}^{<t>}$: NN softmax predicted; $y^{<t>}$: true word

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

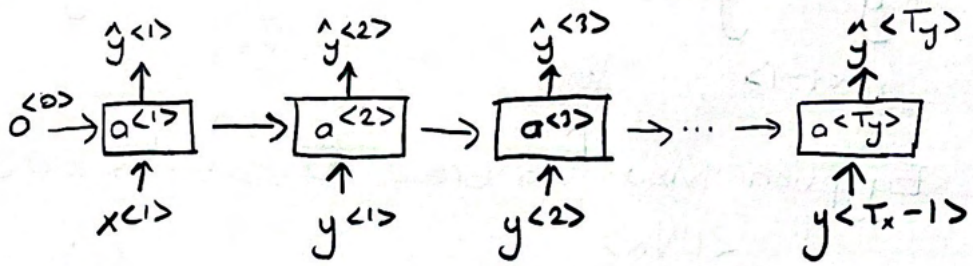
$$P(y^{<1>}, y^{<2>}, y^{<3>})$$
$$= P(y^{<1>})\, P(y^{<2>} \mid y^{<1>})\, P(y^{<3>} \mid y^{<1>}, y^{<2>})$$

$P(y^{<2>} \mid y^{<1>})$ → What's the chance of $y^{<2>}$ given $y^{<1>}$

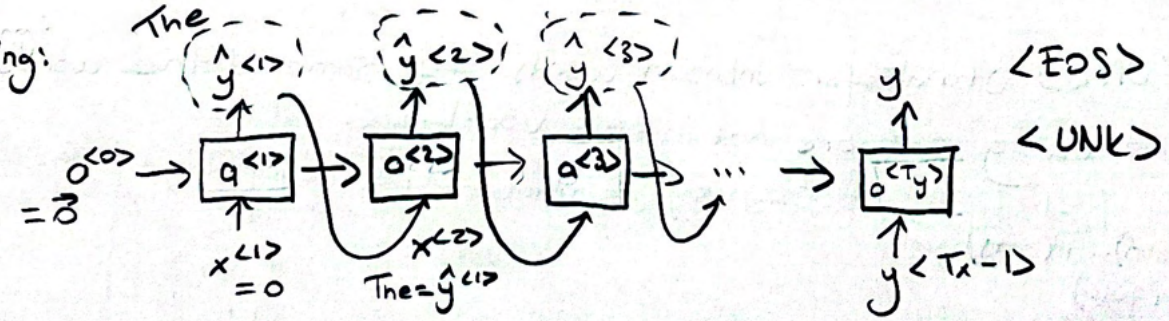
$P(y^{<3>} \mid y^{<1>}, y^{<2>})$ → What's the chance of $y^{<3>}$ given $y^{<1>}$ and $y^{<2>}$

Sampling A Sequence from A Trained RNN

Training:



Sampling:



P(a) P(aaron) ... P(zulu) P(<UNK>) np.random.choice P(_ | the)

Character-level Language Model

Vocabulary = [a, b, c, ..., z, _, ., ,, ;, 0, ..., 9, A, ..., Z]

Cat average. $y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$

↑ ↑ ↑ ↑

Using character-level language model has some pros and cons.

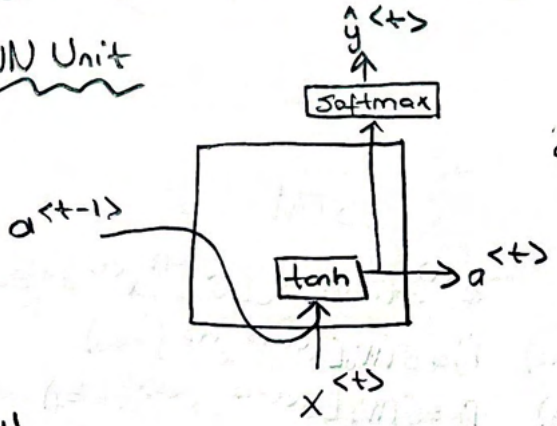
Character-level language model;

+ no <UNK> tokens
- The main disadvantage is much longer sequences.
- Computationally expensive to train

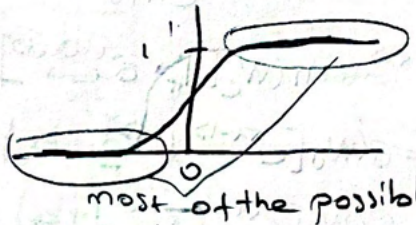
Vanishing gradient → GRU, LSTM
Exploding gradient → Gradient clipping can be used

Gated Recurrent Unit (GRU)

RNN Unit

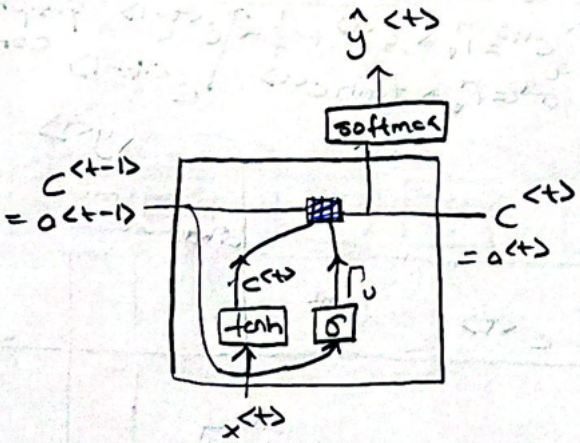


$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$



most of the possible

GRU



c = memory cell value

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

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

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

↖ update

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

$\Gamma_u = 1$ (element-wise)

$\Gamma_u = 0.000001$

Γ : gate

$\tilde{c}$: candidate value

it doesn't suffer from vanishing gradient problem

$\Gamma_u = 1 \quad \Gamma_u = 0 \quad \Gamma_u = 0 \quad \Gamma_u = 0 \ldots$

$c^{<t>} = 1 \quad \ldots\ldots\ldots\ldots = 1$

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

singular or pural

↓ 1 ↓ 0

$c^{<t>}$ can be a vector

Full GRU

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

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

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

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

Γ_r: tells you how relevant is $c^{\langle t-1 \rangle}$ to computing the next candidate for $c^{\langle t \rangle}$.

GRU

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

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

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

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

$1 - \Gamma_u \rightarrow \Gamma_f$

LSTM

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

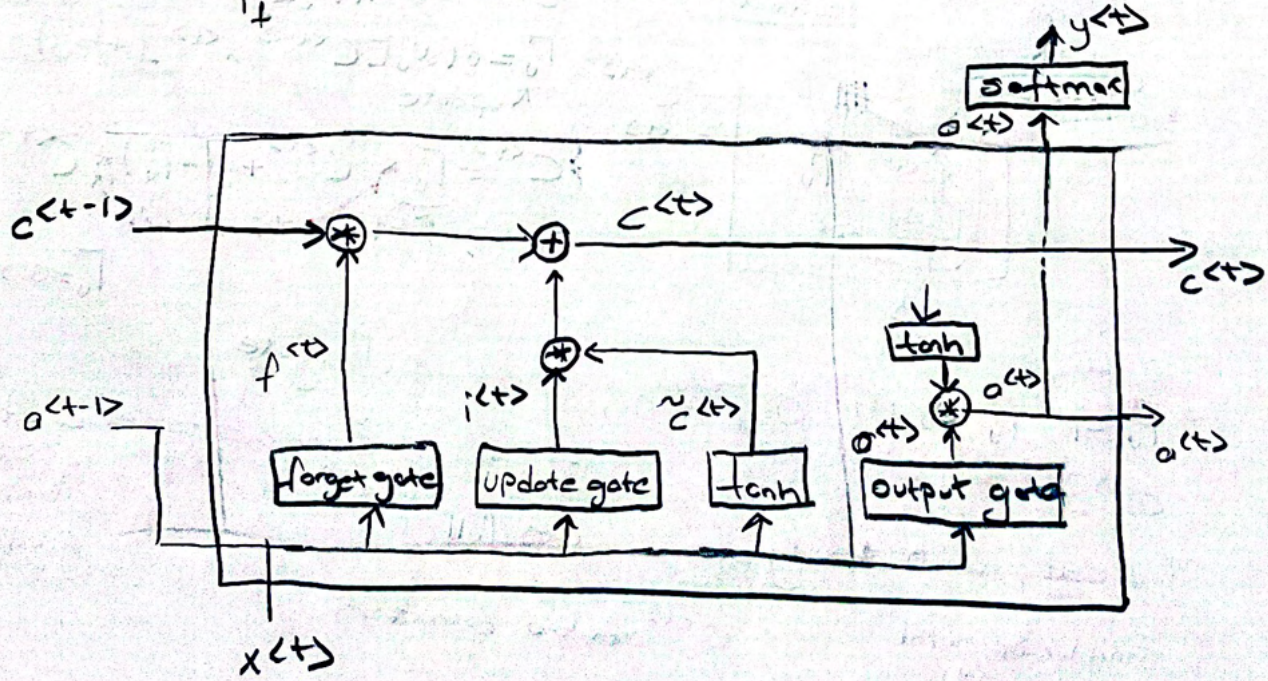
(update) $\Gamma_u = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$

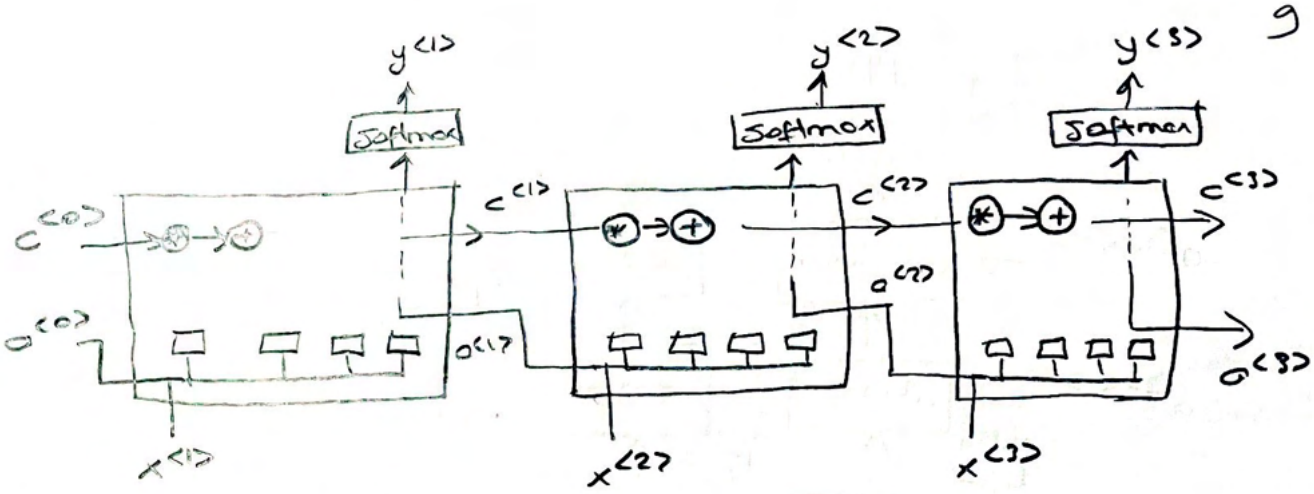
(forget) $\Gamma_f = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$

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

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

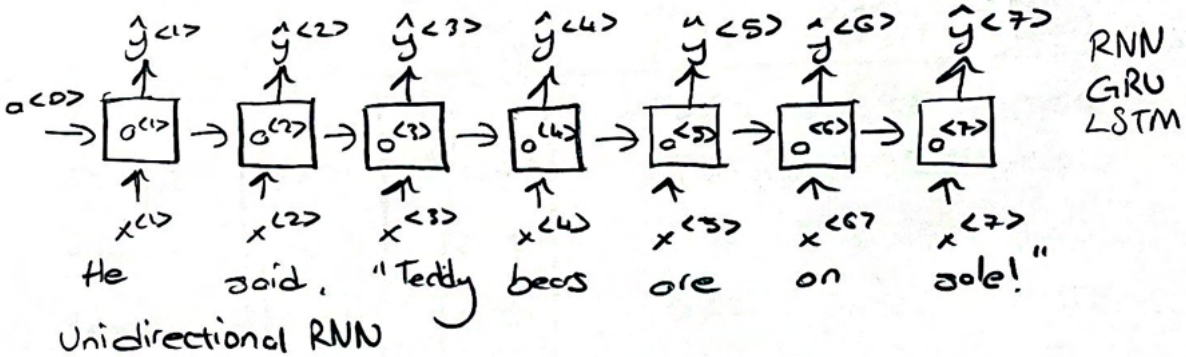
$a^{\langle t \rangle} = \Gamma_o * \tanh c^{\langle t \rangle}$ $c^{\langle t-1 \rangle}$





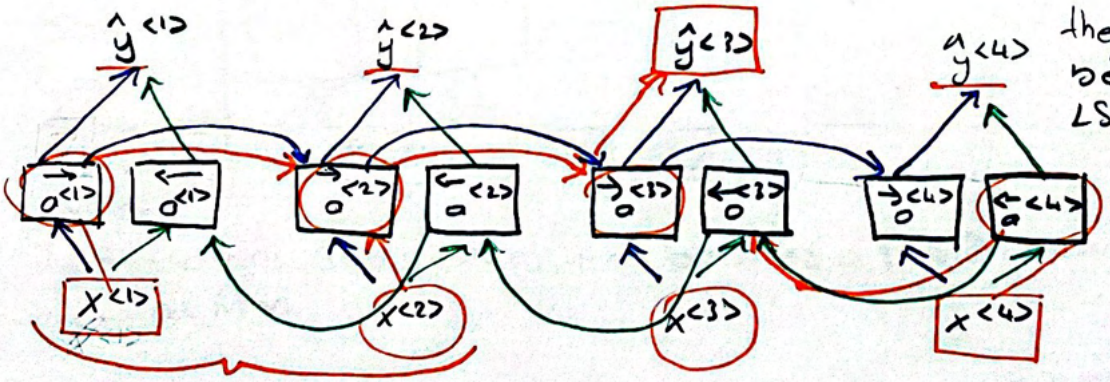
Bidirectional RNNs

He said, "Teddy bears are on sale!"
He said, "Teddy Roosevelt was a great President!"



RNN
GRU
LSTM

Unidirectional RNN



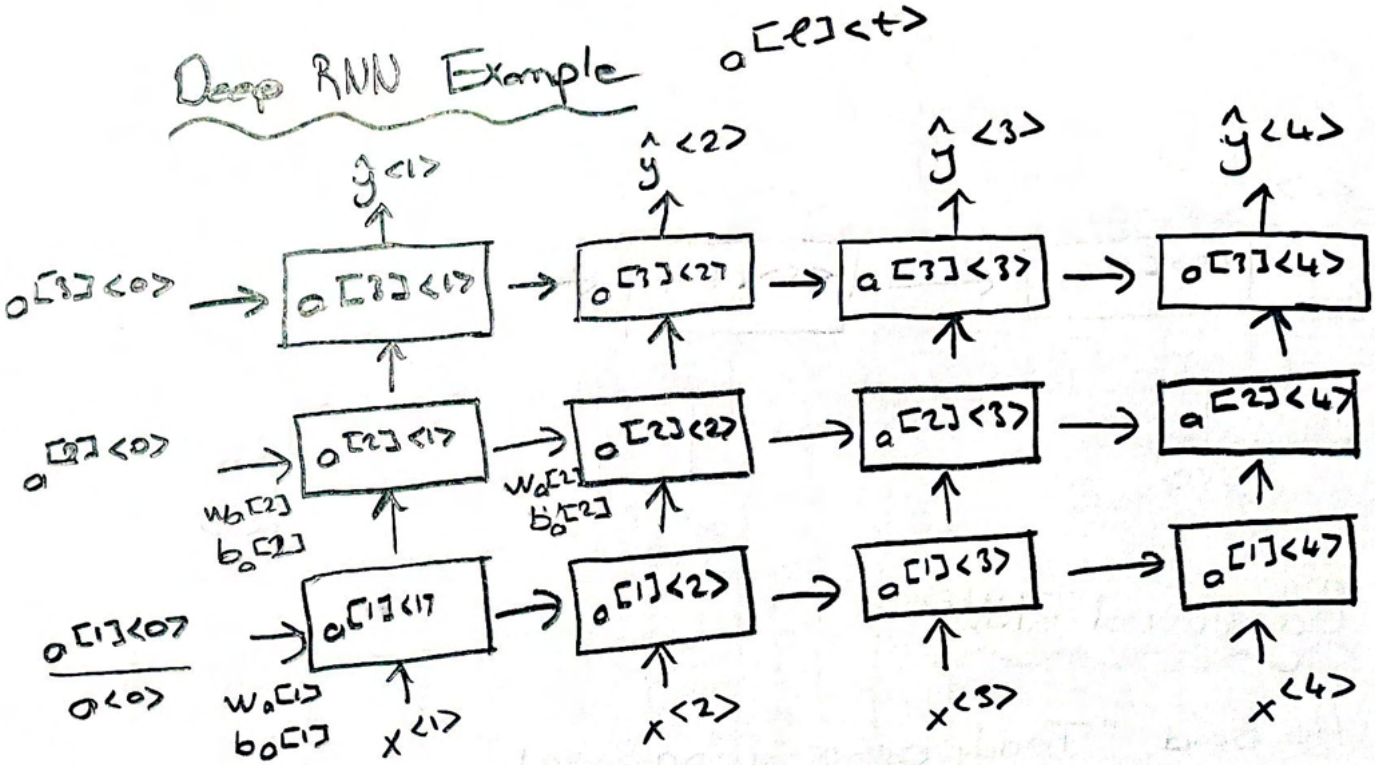
they can also be GRU or LSTM blocks

Acyclic graph (bipartite, no graph cycles)

$$y^{<t>} = g(W_y[\vec{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$$

Bidirectional LSTMs are commonly used in NLP

Deep RNN Example



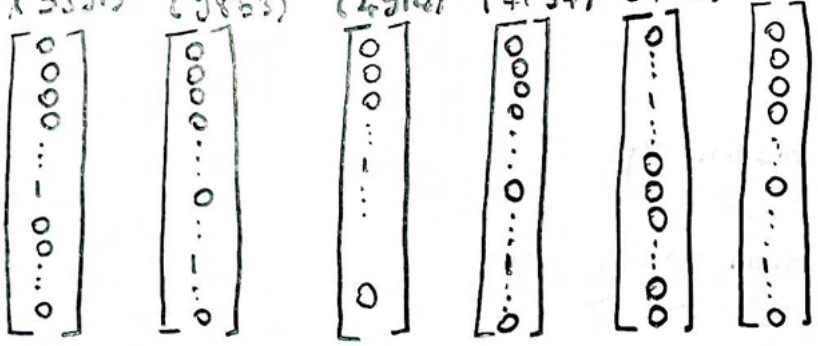
$$a^{[2]<3>} = g(W_a^{[2]}[a^{[2]<2>}, a^{[1]<3>}] + b_a^{[2]})$$

Word Representation

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

one-hot representation

Man (5391) Woman (9853) King (4914) Queen (7157) Apple (456) Orange (6257)



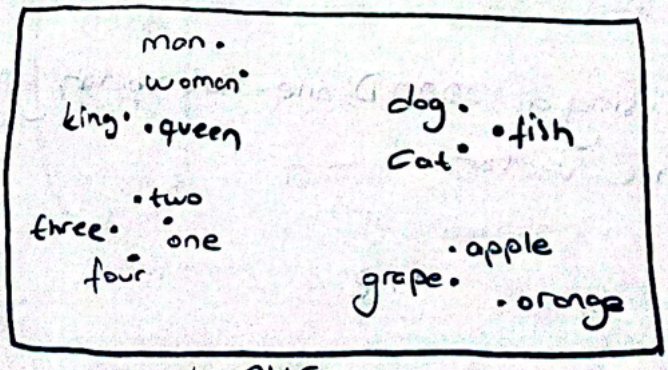
I want a glass of orange juice
I want a glass of apple ?

Featured representation : word embedding

	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	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Size						
Cost						
Alive						
Verb						
Noun						

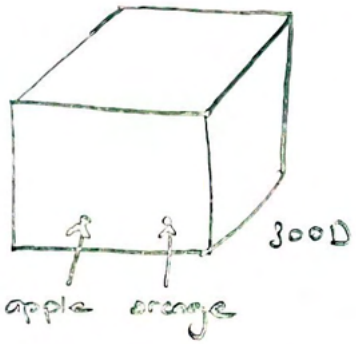
e_{5391} e_{9853}

If there are 300 features, then becomes a 300D vector for representing the word Man.



t-SNE

embeddings → 300D → 2D

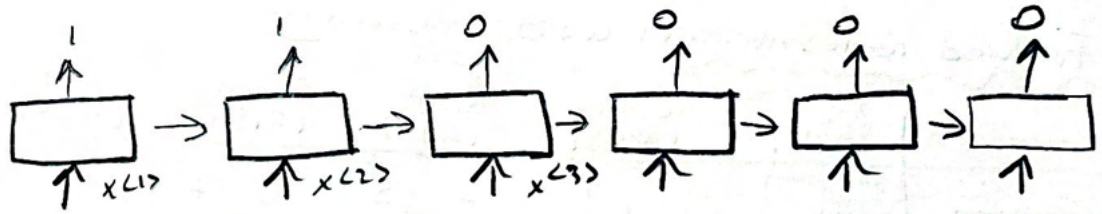


Orange gets embedded to a point in this 300D space

featurized representations
one hot representations

Using Word Embeddings

Named entity recognition example:



Sally	Johnson	is	on	orange	farmer
Robert	Lin	is	on	apple	farmer
BRNN				a	durian cultivator

Transfer learning and word embeddings

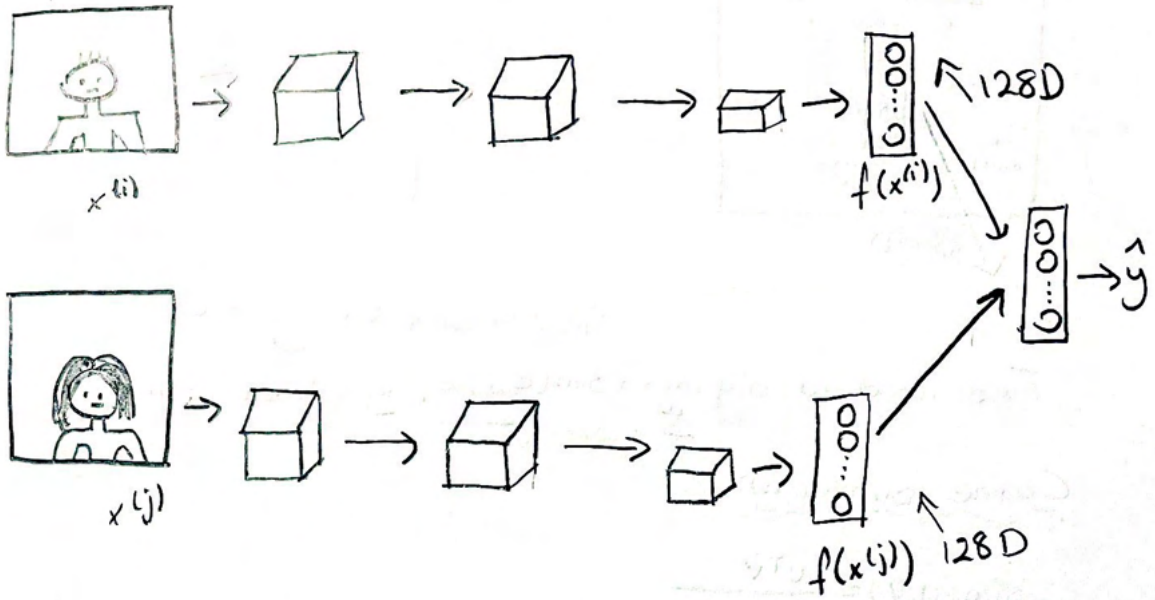
1- Learn word embeddings from large text corpus (1-100B words)
(Or download pre-trained embedding online)

2- Transfer embedding to new task with smaller training set

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

Rather than using a 10000D one-hot vector, you can instead use a 300D dense vector.

Relation to face encoding (embedding)



For face recognition we train a Siamese network orchitecture that would learn a 128 D representation for different faces and compare these encodings in order to figure out if these two pictures are of the same face.

Properties of Word Embeddings

	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	0.7	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

e_{5391} e_{man} e_{woman}

4D embeddings

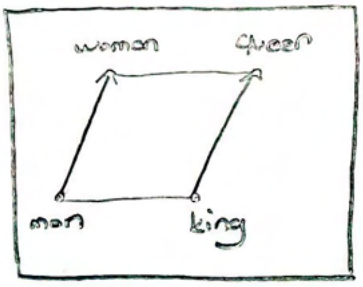
$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

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

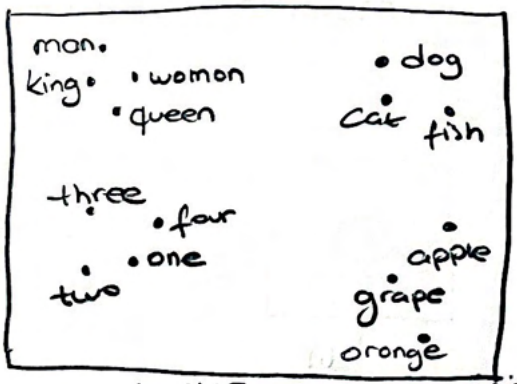
Man → Woman as King → ? Queen

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

Analogies using word vectors



300D



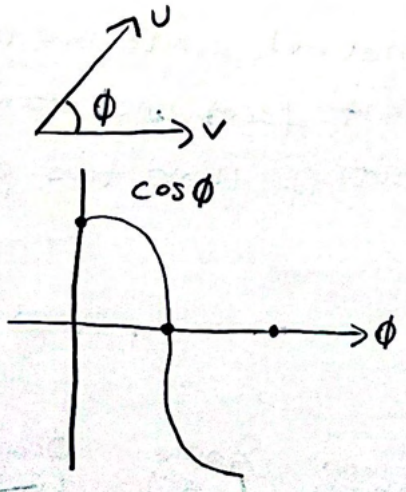
t-SNE non linear mapping

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

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

Cosine similarity

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



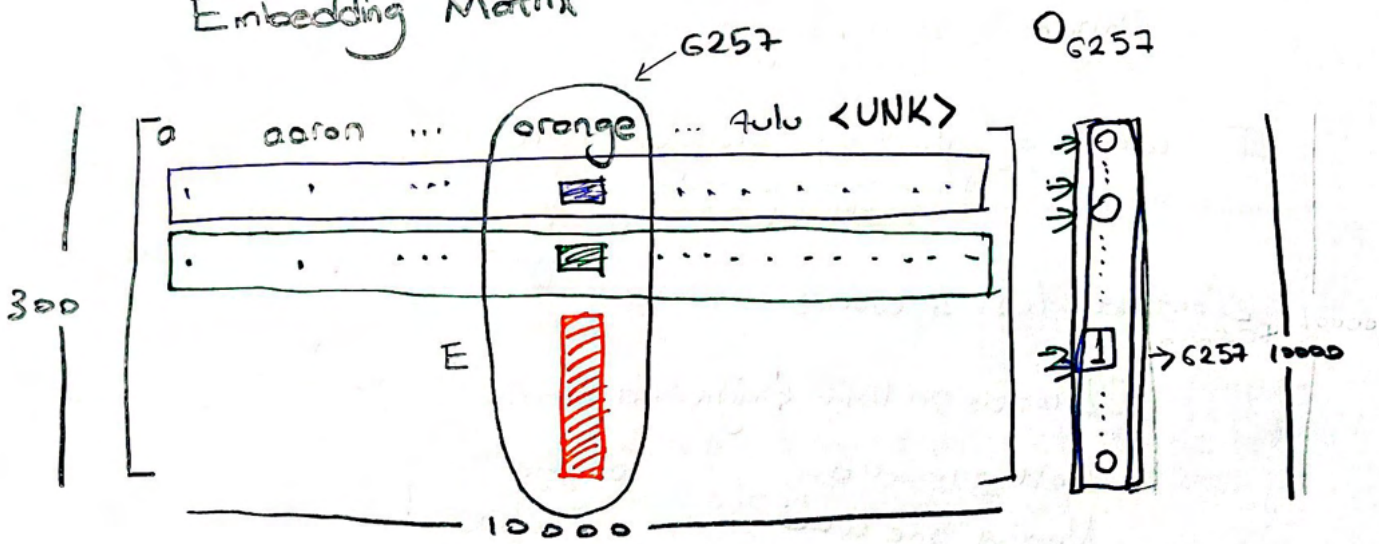
squared distance or euclidian distance can also be use

$\|u - v\|^2$

Man: Woman as Boy: Girl
Ottawa: Canada as Nairobi: Kenya
Big: Bigger as Tall: Taller
Yen: Japan as Ruble: Russia

Embedding Matrix

Embedding Matrix

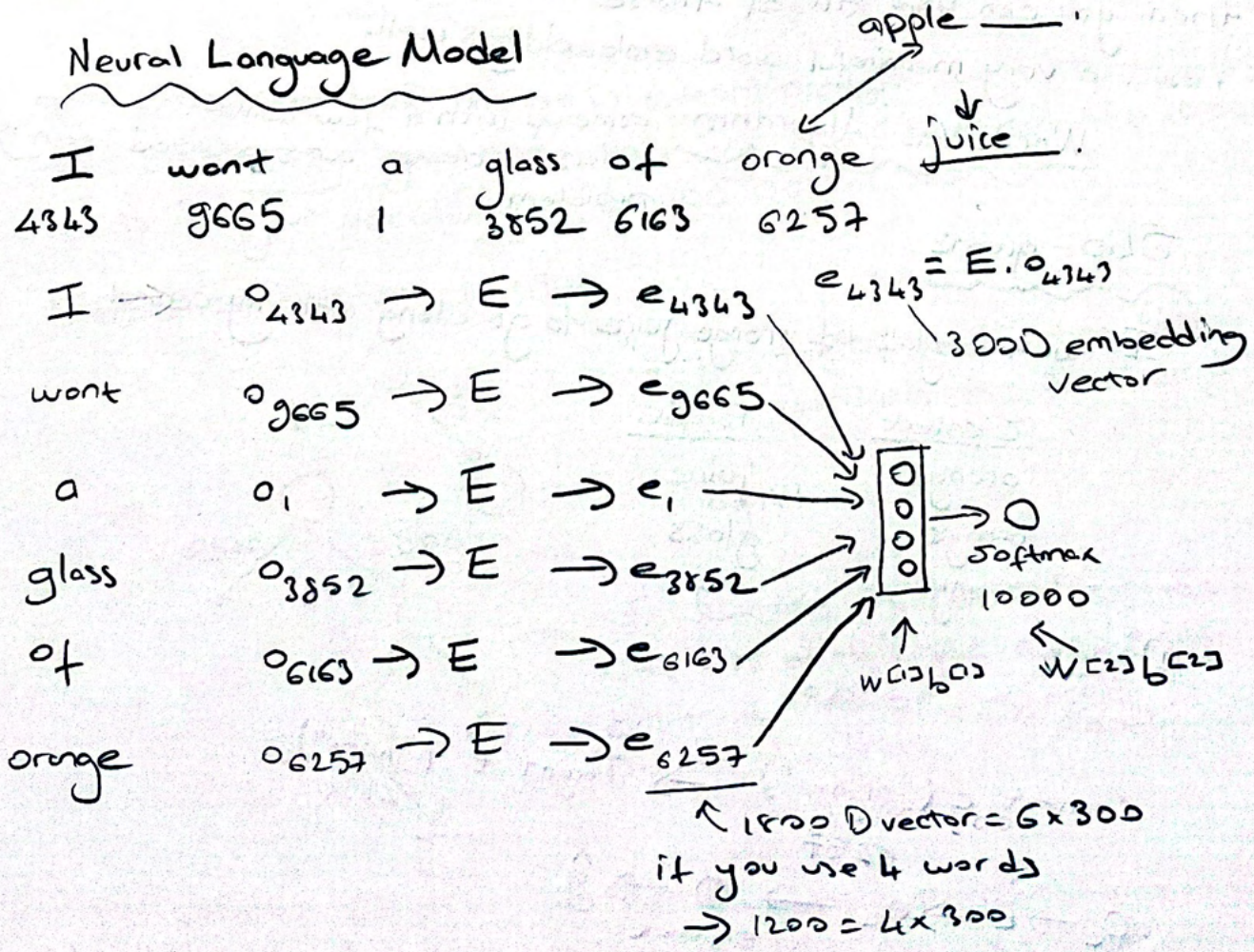


$$E \cdot O_{6257} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}_{(300,1)} = e_{6257}$$

$E \rightarrow (300, 10000)$

$O_{6257} \rightarrow (10000, 1)$

Neural Language Model



Other context/target pairs

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

context → target

Context: last 4 words

try to predict

4 words on left & right — a glass of orange ? — to go along with

Last one word — orange ?

Nearby one word — glass ?

Skip gram

What researchers found was that if you really want to build a language model, it's natural to use the last few words as a context. But if your main goal is to learn a word embedding then you can use all of these other contexts and they will result in very meaningful word embeddings well.

Word2Vec Algorithm come up with a few context to target errors to create our supervised learning problem

Skip-grams

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

Context	Target
orange	juice
orange	glass
orange	my

Model

Vocab size = 10000

Context c ("orange") → Target t ("juice")

6257 4834

$O_c \rightarrow E \rightarrow e_c \rightarrow$ (softmax) $\rightarrow \hat{y}$

$e_c = E O_c$

one-hot vector

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

θ_t = parameter associated with output t

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{10000} y_i \log \hat{y}_i \qquad y = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow 4834$$

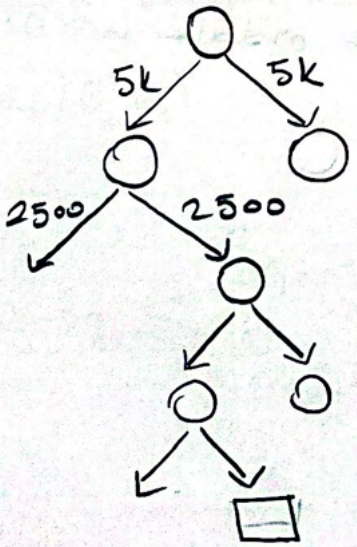
This is called skip-gram model because it is taking as input one word like orange and then trying to predict some words skipping a few words from the left or the right side.

There are a couple problems with using this algorithm. The primary problem is computational speed

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

For the softmax model, everytime you want to evaluate this probability, you need to carry out a sum overall 10000 words in your vocabulary. (or 100,000-1,000,00)

Hierarchical softmax classifier.



is the target word in the first 5000 words in the vocabulary?

$\log|V|$

In practice, hierarchical softmax classifier doesn't use a perfectly balanced tree or perfectly symmetric tree.

Hierarchical softmax classifier can be developed so that the common words tend to be on top where as the less common words can be buried much deeper in the tree.

How to sample the context c?

How do you choose the context c? One thing you can do is just sample uniformly, at random, from your training corpus. When you do that, you find that there are some words like "the, of, a, and, to ..." appear extremely frequently. Your context to target mapping pairs just get these types of words very frequently, whereas there are other words like apple, orange, durian that don't appear that often. You spend almost all the effort updating e_c.

So in practice, the distribution of words P_c isn't taken just entirely uniformly at random for the training set purpose. But instead there are some heuristics that you could use in order to balance out something from the common words together with the less common words.

CBow (continuous backwards) model takes the surrounding contexts from middle word, and uses the surrounding words to try to predict the middle word.

Negative Sampling

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

mapping: x → y

context	word	target
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

k = 5-20 for small datasets
k = 2-5 for large datasets

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

Define a logistic regression model:

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

context	word	target
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0
c	t	y

k-to-1 ratio of − to + examples

orange 6257

$\theta_{6257} \rightarrow E \rightarrow e_{6257}$

juice? king? ... 10000

10000 binary logistic regression classifiers
10000 binary classification problems

On every iteration we're only going to train five of them (k+1) Computation cost of this algorithm is much lower

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10000} f(w_j)^{3/4}} \qquad \frac{1}{|V|}$$

GloVe (global vectors for word representation)

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

c, t

X_{ij} = # times j appears in context of i

c t — t — c

$X_{ij} = X_{ji}$ depending on the definition of context

Symmetric relationship

For the purposes of the GloVe algorithm, we can define context and target as whether or not the two words appear in close proximity.

Model

$$\text{minimize} \sum_{i=1}^{10000} \sum_{j=1}^{10000} f(X_{ij})(\theta_i^T e_j + b_i + b_j' - \log X_{ij})^2$$

(θ_i: t, e_j: c)

this, is, of, a... → stop words

θ_i, e_j are symmetric

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

How related are those two words?

words t and c

j i

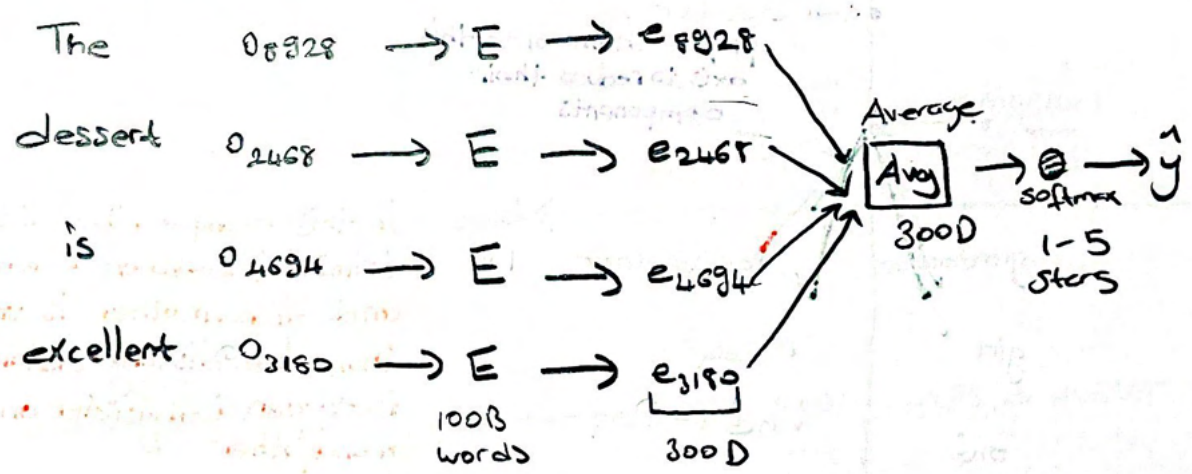
End product is a good predictor for how often the two words occur together

Sentiment Classification

Simple Sentiment Classification Model

The dessert is excellent. ★★★★☆

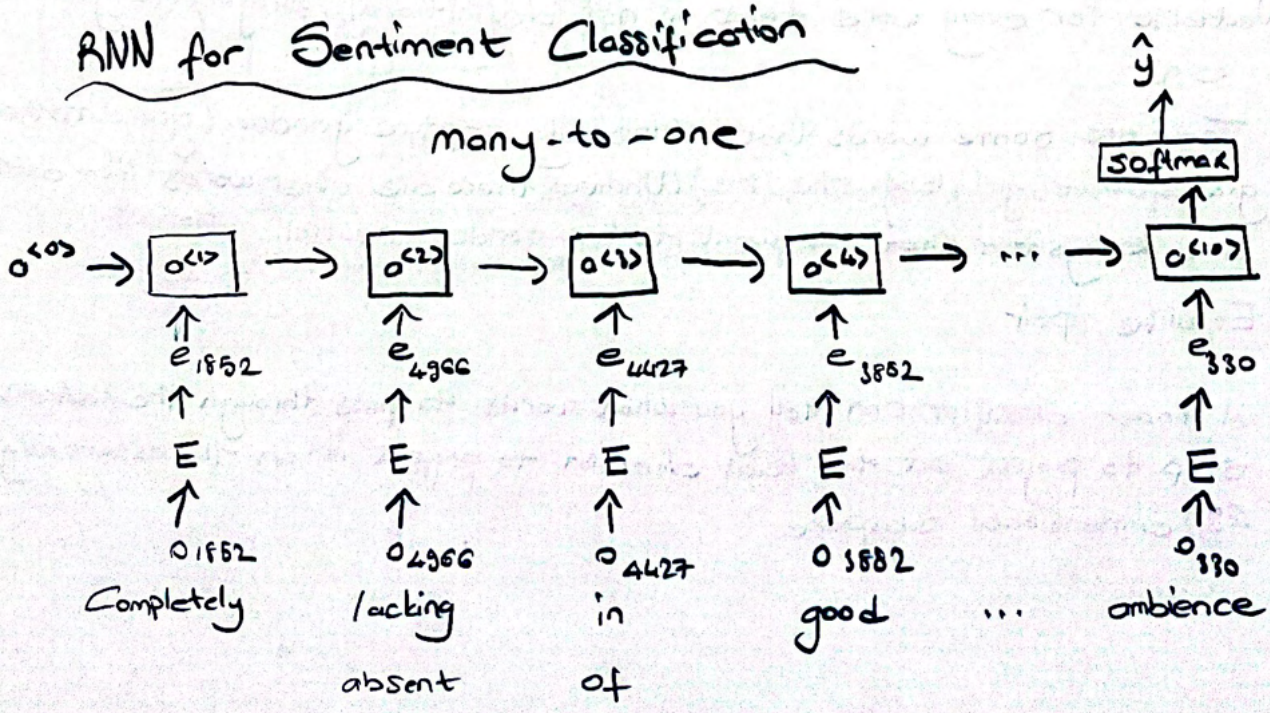
8928 2468 4694 3180



One of the problems with this algorithm is it ignores word order.

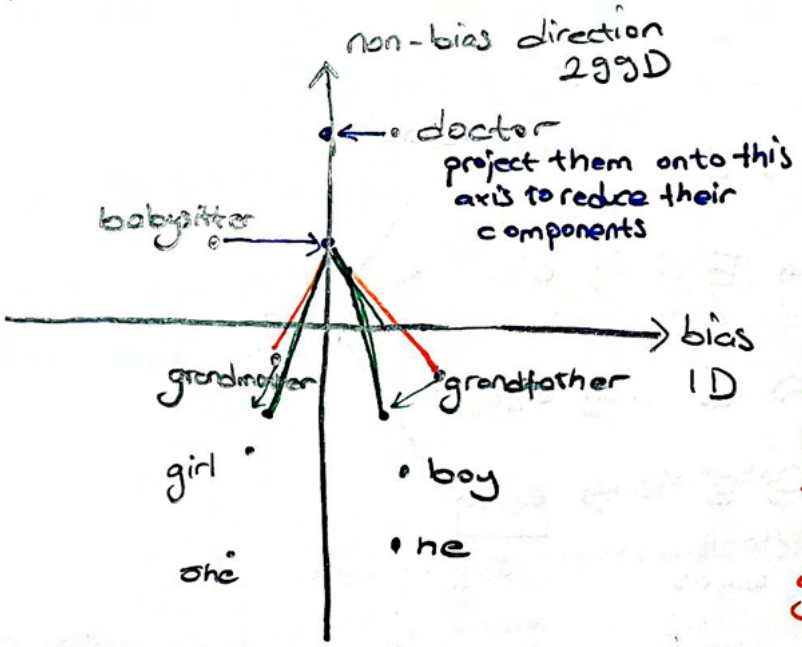
"Completely lacking in good taste, good service, and good ambience" this is a very negative review.

RNN for Sentiment Classification



Debiasing Word Embeddings

Addressing Bias in Word Embeddings



Singular Value Decomposition (SVU)
uses ideas similar to PCA

in this example, the distance (similarity) between babysitter and grandmother is smaller than the [redacted] distance between babysitter and grandfather

1- Identify bias direction

$e_{he} - e_{she}$ } average
$e_{male} - e_{female}$ }

⋮

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

There are some words that intrinsically capture gender. (grandmother, grandfather, girl, boy, she, he) Whereas there are other words like doctor and babysitter that we want to be gender neutral.

3- Equalize pair

A linear classifier can tell you what words to pass through the neutralization step to project out this bias direction to project it on this essentially 299 dimensional subspace

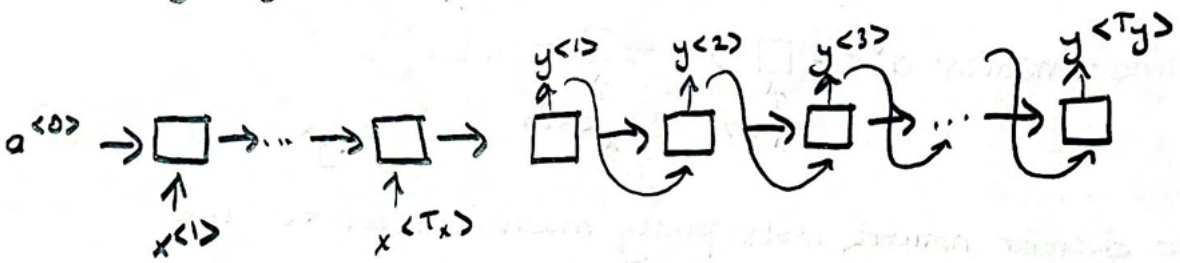
Sequence to Sequence Models

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

Jane visite l'Afrique en septembre

→ Jane is visiting Africa in September.

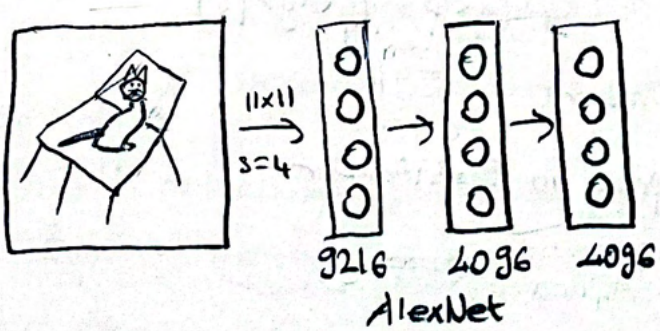
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$



encoder decoder

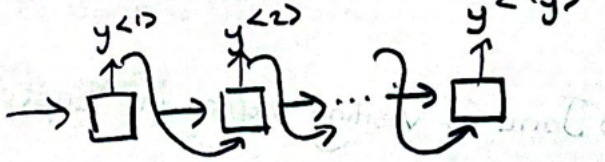
After ingesting the input sequence the RNN outputs a vector that represents the input sequence.

Image Captioning



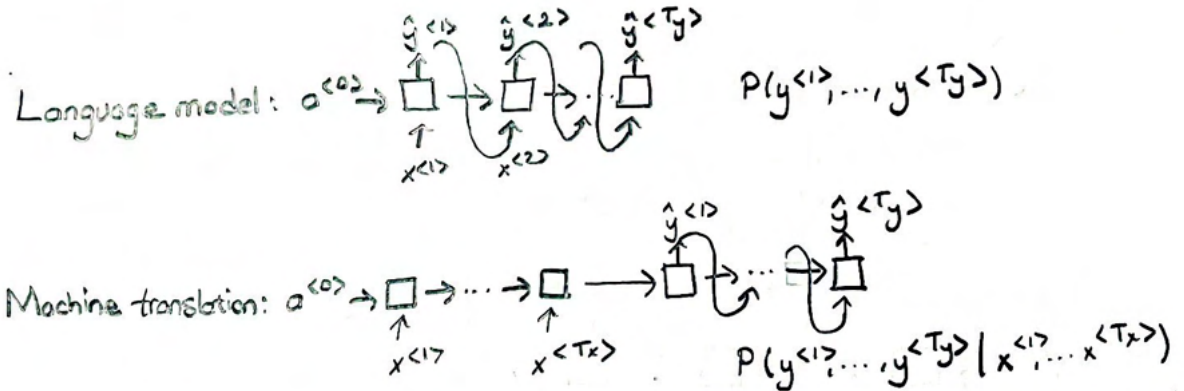
A cat sitting on a chair.

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



Pre-trained AlexNet can be the encoder network for the image. Now you have a 4096 dimensional vector that represents the image.

Machine Translation as Building a Conditional Language Model



The decoder network looks pretty much identical to the language model.

"Conditional language model".

Modeling the probability of the English translation, conditions on some input French sentence.

Finding the Most Likely Translation

Jane visite l'Afrique en septembre.

$P(\overbrace{y^{<1>}, \dots, y^{<T_y>}}^{\text{english}} | \underbrace{x}_{\text{french}})$

→ Jane is visiting Africa in September

→ Jane is going to be visiting Afrika in September

→ In September, Jane will visit Africa.

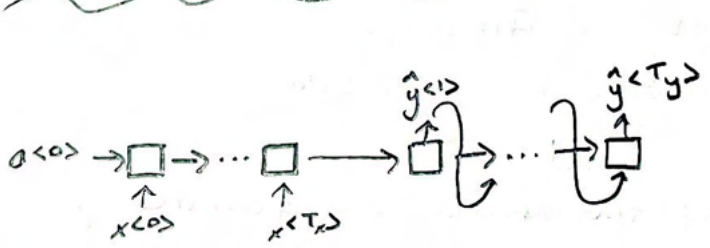
→ Her African friend welcomed Jane in September.

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

When you're using this model for machine translation, you're not trying to sample at random from this distribution. Instead, what you would like to find the English sentence y, that maximizes that conditional probability. Developing a machine translation system, one of the things you need to do is come up with an algorithm that can actually find the value of y that maximizes this term.

Greedy search says to generate the first word just pick whatever is the most likely first word according to your conditional language model.

Why not a greedy search?



$$\arg\max_{y} P(\hat{y}^{<1>}, \hat{y}^{<2>}, \ldots, \hat{y}^{<T_y>}|x)$$

→ Jane is visiting Africa in September

→ Jane is going to be visiting Africa in September

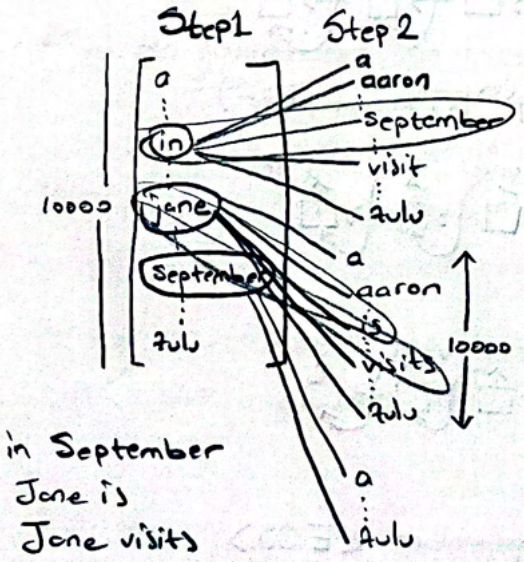
$$P(\text{Jane is going}|x) > P(\text{Jane is visiting}|x)$$

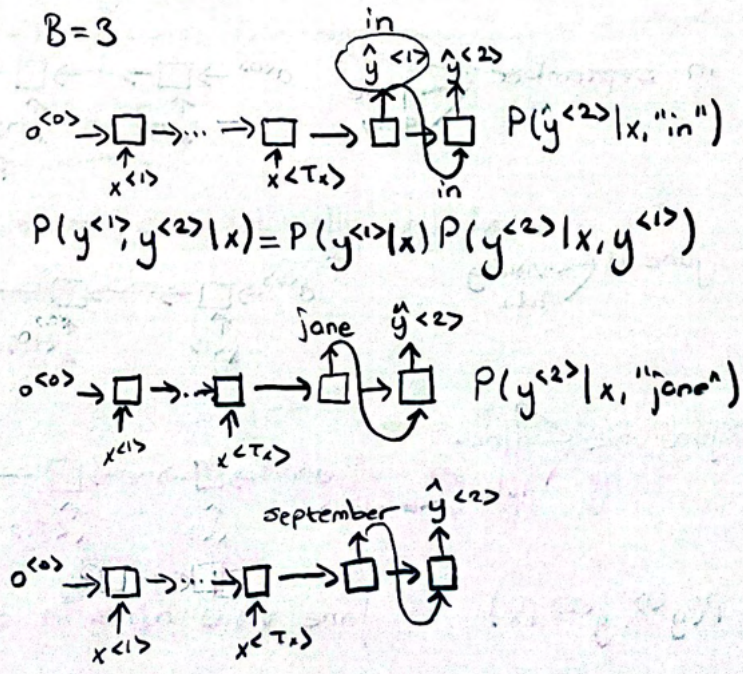
"going" is more common English word.

It's not always optimal to just pick one word at a time.

Beam Search Algorithm

$B=3$



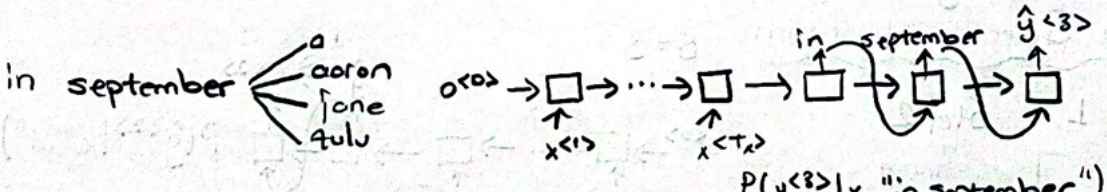


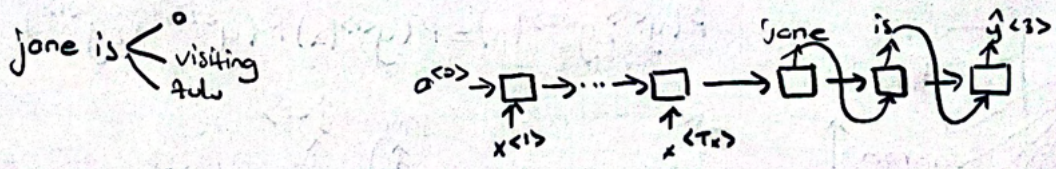
Whereas greedy search will pick only the one most likely word and move on, Beam search instead can consider multiple alternatives. So the Beam search algorithm has a parameter called B, which is called the beam width. This means beam search will consider not just one possibility but consider (for this example) 3 at a time.

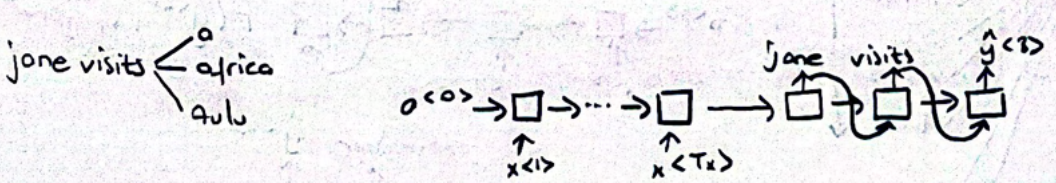
In step1, beam search picked Jane and September as the 3 most likely choice of the first word. In step2, what beam search will do now, is for each of these 3 choices consider what should be the second word.

What is the probability of the second word given the input French sentence and that the first word of the translation has been the word "in"

If beam search decides that the most likely choices are in the first and second words are "in September," "Jane is", or "Jane visits" then what that means is that it is now rejecting September as a candidate for the first words of the output English translation. So We are now down to two possibilities for the first word but we still have a beam width of 3. Because of B=3, every step you instantiate 3 copies of the network to evaluate these partial sentence fragments in the output.







$P(y^{<1>}, y^{<2>}|x)$ jane visits africa in september. <EOS>

Refinements to Beam Search

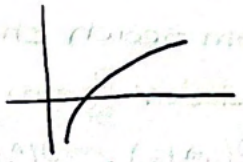
Length Normalization

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

$$\arg\max_y \prod_{t=1}^{T_y} P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})$$

these probabilities are all numbers less than 1 result in a tiny number. which can result in numerical underflow. It's too small for the floating part representation in your computer to store accurately. So in practice, instead of maximizing this product, we will take logs

$$\arg\max_y \sum_{y=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})$$



Maximizing $\log P(y|x)$ should give you the some result as maximizing $P(y|x)$

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

$\alpha = 0.7$ $\alpha = 1$ $\alpha = 0$

This takes the average of the log of the probability of each word. And this significantly reduces the penalty for outputting longer translations.

How do you choose the beam width B?

Large B: better result, slower, computationally expensive

Small B: worse result, faster

In production → 10, 100, 1000, 3000

Unlike exact search algorithms like BFS or DFS. Beam Search runs faster but is not guaranteed to find exact maximum for $\arg\max_y P(y|x)$.

Error Analysis in Beam Search

Beam search is an approximate search algorithm, also called a heuristic search algorithm. So it doesn't always output the most likely sentence.

Jane visite l'Afrique en septembre.

Human: Jane visits Africa in September. (y^*)

Algorithm: Jane visited Africa last September. ($\hat{y}$)

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

Case 1: $P(y^*|x) > P(\hat{y}|x)$ $\quad \arg\max_y P(y|x)$

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

Conclusion: Beam search is at fault

Case 2: $P(y^*|x) \leq P(\hat{y}|x)$

y^* is a 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	Algorithm	$P(y^*\|x)$	$P(\hat{y}\|x)$	At fault?
Jane visits Africa in September	Jane visited Africa last September	2×10^{-10}	1×10^{-10}	B
...	...	—	—	R
...	...	—	—	B
				R
				R
				⋮

Figures out what fraction of errors are "due to" beam search vs. RNN model

Bleu Score (Bilingual Evaluation Understudy)

If there are multiple great answers, how do you measure accuracy?

French: Le chat est sur le tapis.

Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

Given a machine generated translation, bleu score allows you to automatic compute a score that measures how good is that machine translation.

MT output: the the the the the the the

Precision: $\frac{7}{7_{\text{words}}}$ Modified precision: $\frac{2 \leftarrow \text{count}_{clip}(\text{"the"})}{7 \nwarrow \text{count}(\text{"the"})}$

In modified precision measure, we will give each word a credit up to the maximum number of times it appears in the reference sentences.

Bleu Score on Bigrams — Unigram / Trigram

Bigrams means pairs of words appearing next to the each other.

Example: 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. ($\hat{y}$)

The possible bigrams are:

	Count	count_{clip}	
the cat	2 ←	1 ←	
cat the	1 ←	0	$\frac{4}{6}$
cat on	1 ←	1 ←	
on the	1 ←	1 ←	
the mat	1 ←	1 ←	

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

(unigram)

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

(n-gram)

Bleu Details

P_n = Bleu score on n-grams only

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

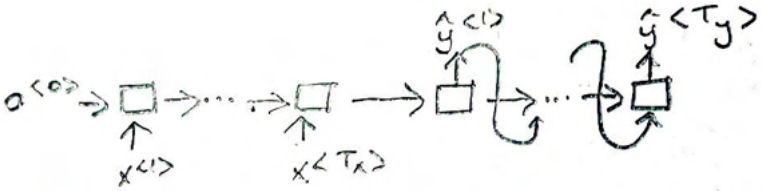
BP = Brevity Penalty

$$BP = \begin{cases} 1, & \text{if MT\_output\_length} > \text{reference\_output\_length} \\ \exp(1 - \text{reference\_output\_length}/\text{MT\_output\_length}), & \text{otherwise} \end{cases}$$

If you output very short translations, it's easier to get high precision. Because probably most of the words you output appear in the references. (We don't want this) BP is an adjustment factor that penalizes translation systems that output translations are too short.

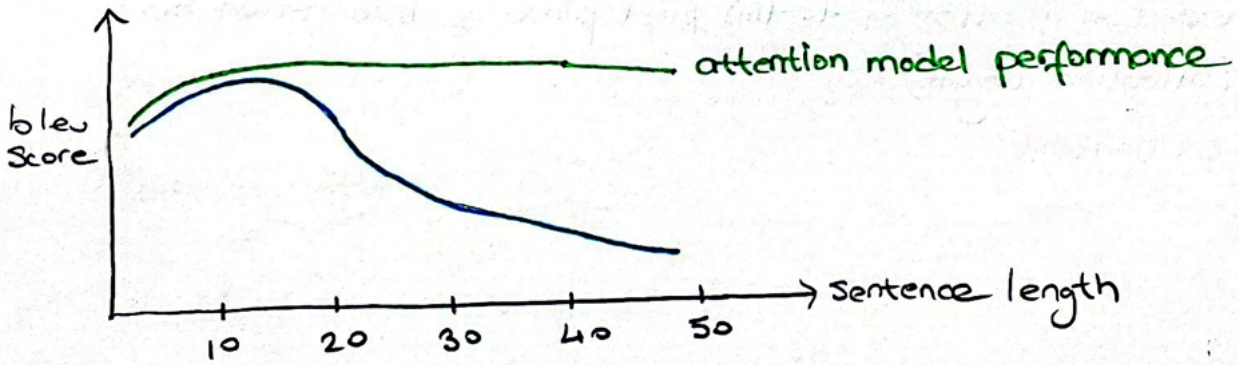
Attention Model Intuition

The Problem of Long Sequences

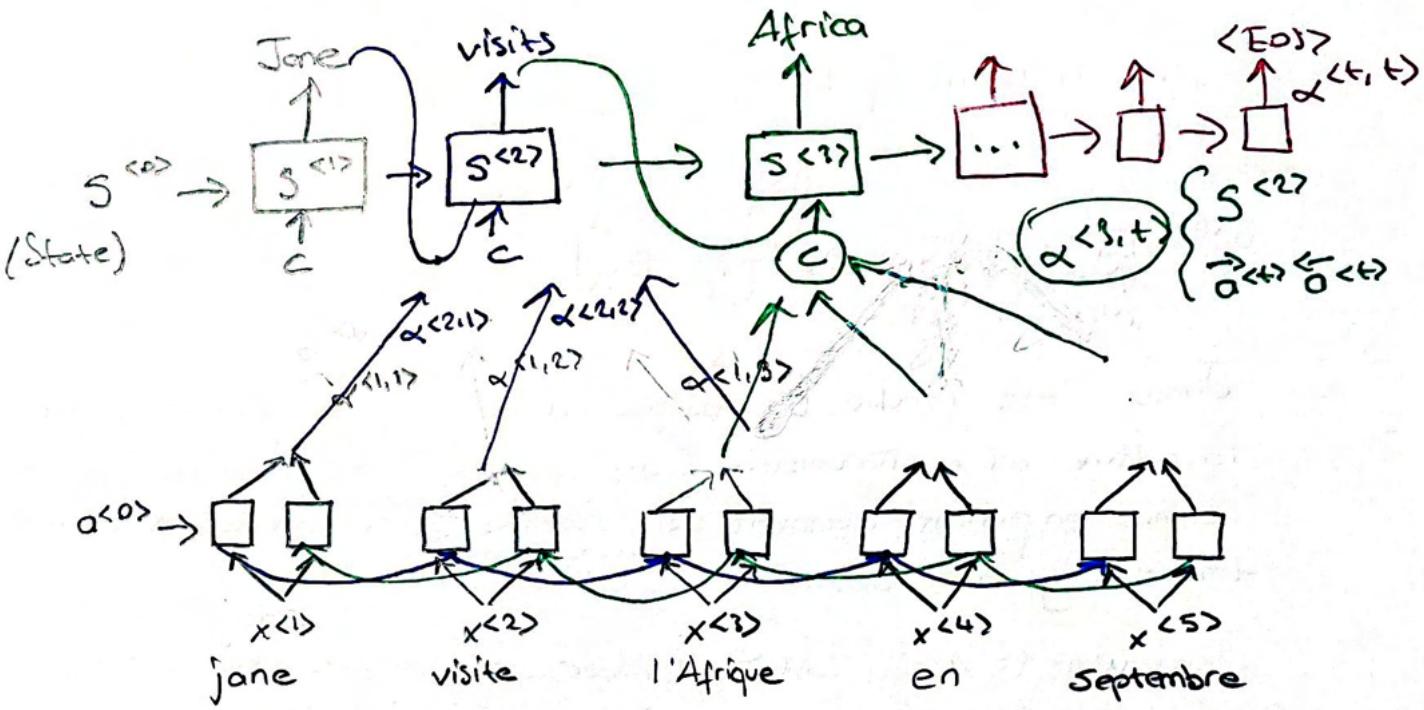


Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.



The encoder has to read the whole sentence and memorize it
A human translator would read the first part of the sentence, maybe generate part of the translation, look at the second part, generate a few more words and so on. (It's hard to memorize the whole sentence)
Attention model translates a bit more like humans might.



$\alpha^{<1,1>}$: when you're generating the first word, how much should you be paying attention to this first piece of information here. (attention weight)

c: context

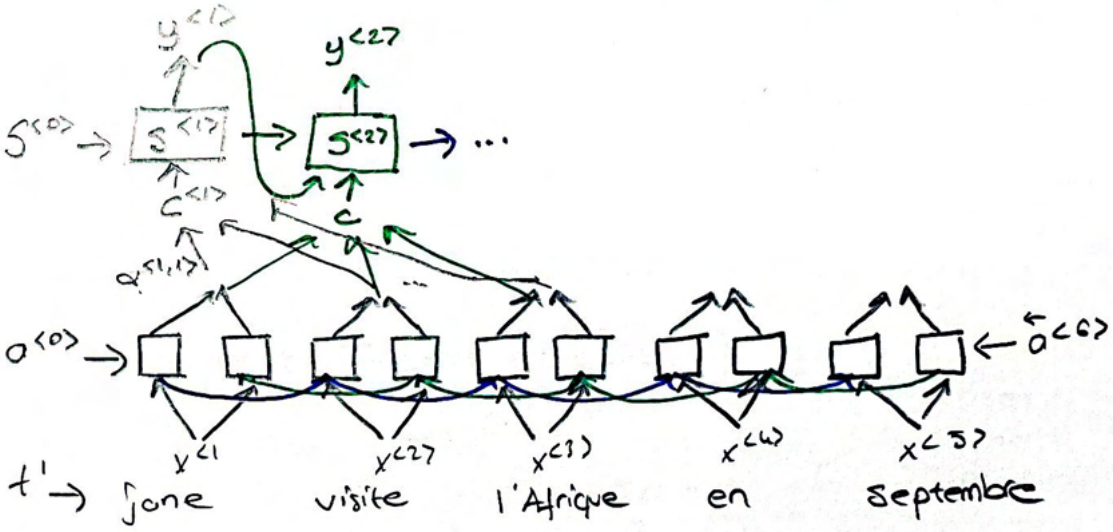
Attention Model

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

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

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

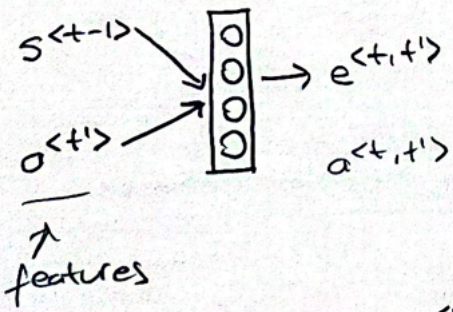
$$C^{<2>} = \sum_{t'} \alpha^{<2,t'>} a^{<t'>}$$

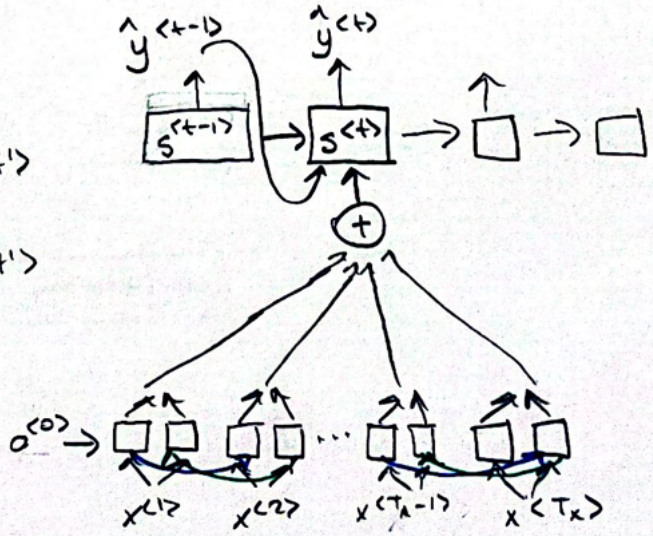


$a^{<t>} = (\overrightarrow{a}^{<t>}, \overleftarrow{a}^{<t>})$ $t' \rightarrow$ french sentence

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

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

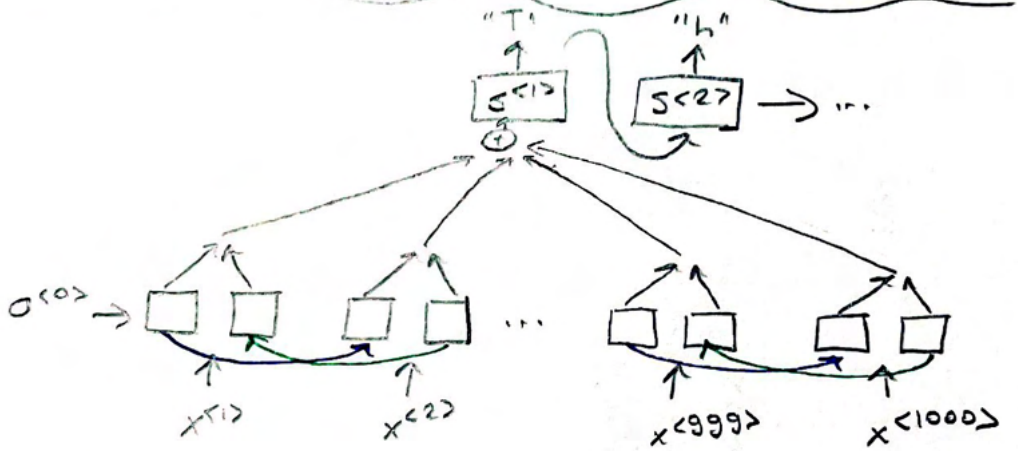




total number of $\alpha^{<t,t'>}$ parameters $T_x \cdot T_y$

Quadratic cost

Attention Model for Speech Recognition

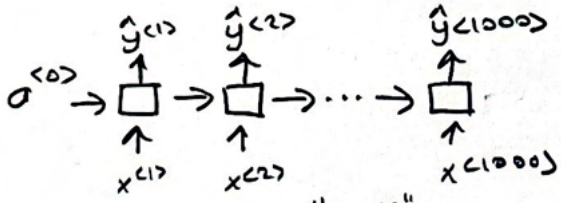


CTC Cost for Speech Recognition

(Connectionist Temporal Classification)

"the quick brown fox" ← 19 characters

10 secs of audio
100 hertz
100 samples per sec

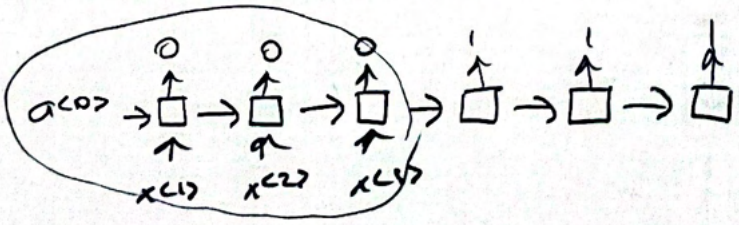


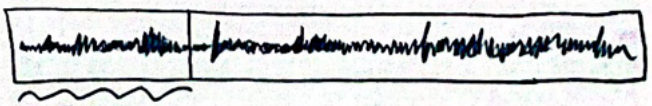
ttt_h_eee___␣___qqq__

"space" "blank"

Basic rule: collapse repeated characters not separated by "blank"

Trigger Word Detection





hey siri

Self-Attention Intuition

$A(q, K, V)$ = attention-based vector representation of a word
↳ calculate for each word $A^{<1>}, \ldots, A^{<5>}$

RNN Attention

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

Transformers Attention

$$A(q, K, V) = \sum_i \frac{\exp(e^{<q \cdot k^{<i>}>})}{\sum_j \exp(e^{<q \cdot k^{<j>}>})} v^{<i>}$$

$A^{<3>}$

$x^{<1>}$	$x^{<2>}$	$x^{<3>}$	$x^{<4>}$	$x^{<5>}$
Jane	visite	l'Afrique	en	septembre

[Vaswani et al. 2017, Attention Is All You Need]

Andrew Ng

vectors: $q^{<1>}$: query; $k^{<1>}$: key; $v^{<1>}$: value

Self-Attention

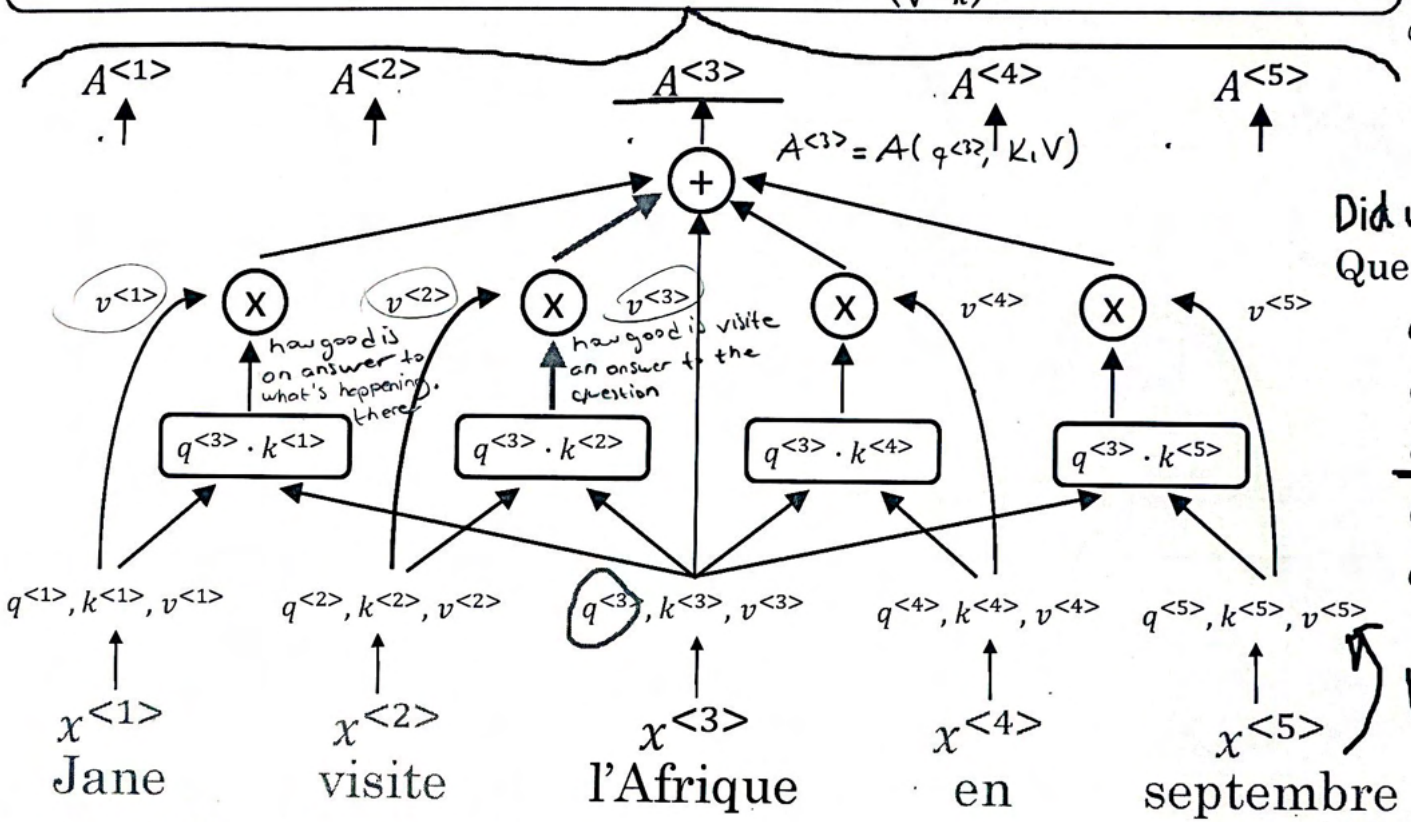
$$A(q,K,V) = \sum_i \frac{\exp(e^{q \cdot k^{<i>}})}{\sum_j \exp(e^{q \cdot k^{<j>}})} v^{<i>}$$

softmax

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

Scaled dot product attention

The complexity of computing the softmax for SA block is quadratic with respect to the length of the input sequence.



Did what?

Query (Q)	Key (K)	Value (V)
$q^{<1>}$	$k^{<1>}$ person	$v^{<1>}$ Jane
$q^{<2>}$	$k^{<2>}$ action	$v^{<2>}$ visit
$q^{<3>}$ What's happening there?	$k^{<3>}$	$v^{<3>}$
$q^{<4>}$	$k^{<4>}$	$v^{<4>}$
$q^{<5>}$	$k^{<5>}$	$v^{<5>}$

W^Q, W^K, W^V

Vaswani et al. 2017, Attention Is All You Need]

$q^{<3>}$ is a question that you get to ask about l'Afrique, represents a question like "what's happening there?"

$q^{<3>} = W^Q x^{<3>}$
$k^{<3>} = W^K x^{<3>}$
$v^{<3>} = W^V x^{<3>}$

Andrew Ng

Multi-Head Attention

consists of multiple Self Attention blocks (heads)

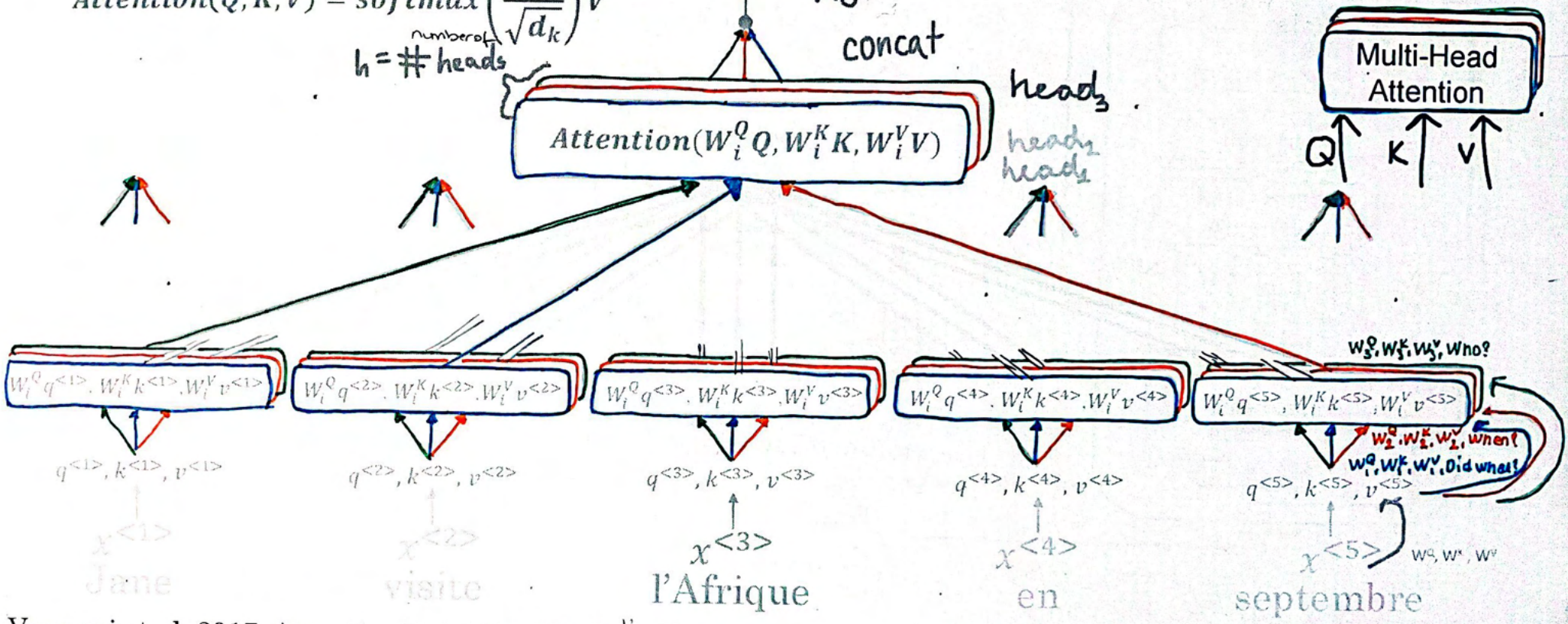
you can compute different heads in a for loop
in practice you can calculate them in parallel because no one has value depends on the value of any other head

$$MultiHead(Q,K,V) = concat(head_1 head_2 \dots head_n)W_o$$

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

$h = \#$ number of heads

$$head_i = Attention\,(W_i^Q Q, W_i^K K, W_i^V V)$$



Vaswani et al. 2017, Attention Is All You Need]

W_1^Q, W^K, W_1^V → What's happening there?

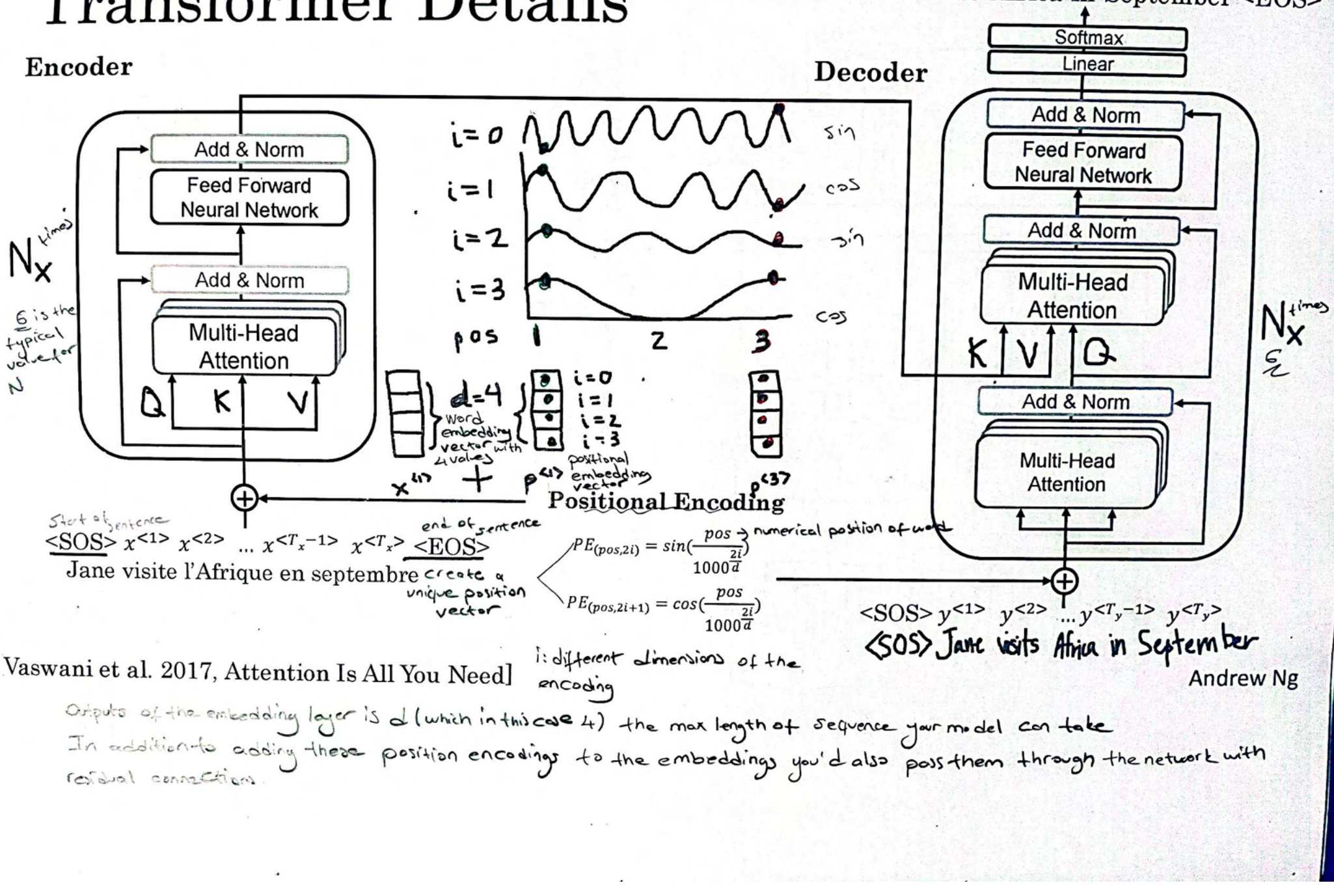
W_2^Q, W_2^K, W_2^V → When's something happening?

W_3^Q, W_3^K, W_3^V → Who? The inner product between Jane's key vector and the l'Afrique query vector will be the highest

The complexity of computing the softmax for SA block is quadratic with respect to the length of the input sequence.

Andrew Ng

Transformer Details



Vaswani et al. 2017, Attention Is All You Need]

i: different dimensions of the encoding

Andrew Ng

Outputs of the embedding layer is d (which in this case 4) the max length of sequence your model can take

In addition to adding these position encodings to the embeddings you'd also pass them through the network with residual connection.