II spec II course V II week 1

Recurrent neural networks

Why sequence models

Supervised learning

Speech recognition — x: audio clip → y: "The quick brown fox jumped over the lazy dog"

Music generation — x: nothing or ∅ or eg. single integer → y: 

Sentiment classification — "there is nothing to like in this movie" → ★☆☆...

DNA sequence analysis — ACGCCTAAG → ...

Machine translation — french → english

Video activity recognition — movie clip → running

Name entity recognition — sentence → identify names

RNNs have a kind of memory = c

BRNN can work with info from past and future.

Notation example

NLP - Natural Language Processing

x: (Harry Potter) and (Hermione Granger) invented a new spell

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

input sequence length $T_x = 9$ (can be different)

y: 1 1 0 1 1 0 0 0 0

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

output sequence length $T_y = 9$

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

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

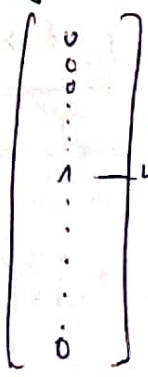
Representing words

Vocabulary (dictionary):

a	1
aaron	2
...	...
and	367
...	...
harry	4075
...	...
potter	6830
...	...
zulu	10000 words

$n_x \in \mathbb{R}$ (10000)

shape (10000,)



<unk> — unknown word

quite small common 50k-100k or millions

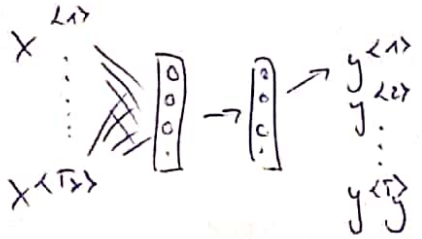
one-hot: one 1 and zeros everywhere

input tensor shape (n_x, m, T_x)

(i) mini-batch size (n_a, m)

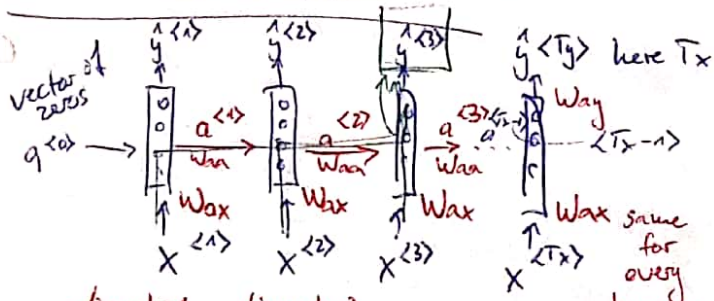
$a^{<t>}$ is called "hidden state"

Why not a standard network?



Problems:
- Inputs, outputs can be different in lengths in different examples
- Doesn't share features learned across different positions of text
  - ↳ as in CNN: want things learned in part of the image to generalize quickly to other parts of the image
  - → weight matrix will be enormous

(unidirectional) Recurrent Neural Network



timestep 1 timestep 2

→ $\hat{y}^{<2>}$ is not only calculated by $x^{<2>}$ but also gets input from timestep 1

Weakness: only uses information that is earlier in the sequence to make a prediction

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

↳ can't tell whether "Teddy" is a part of a person's name, in 1st sentence yes, in 2nd no.

→ later: Bidirectional RNN

Forward Propagation

$a^{<0>} = \vec{0}$ (activation)

$$a^{<1>} = g(W_{aa} \cdot a^{<0>} + W_{ax} \cdot x^{<1>} + b_a)$$
$$y^{<1>} = g(W_{ya} \cdot a^{<1>} + b_y)$$

↖ sigmoid for binary classification

$$a^{<t>} = g(W_{aa} \cdot a^{<t-1>} + W_{ax} \cdot x^{<t>} + b_a)$$
$$\hat{y}^{<t>} = g(W_{ya} \cdot a^{<t>} + b_y)$$

$a \leftarrow W_{ax} x^{<1>}$ ↳ multiplied by some x-like quantity to compute some a-like quantity

Simplified RNN notation:

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

(W_{aa}: (100, 100), W_{ax}: (100, 10'000))

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

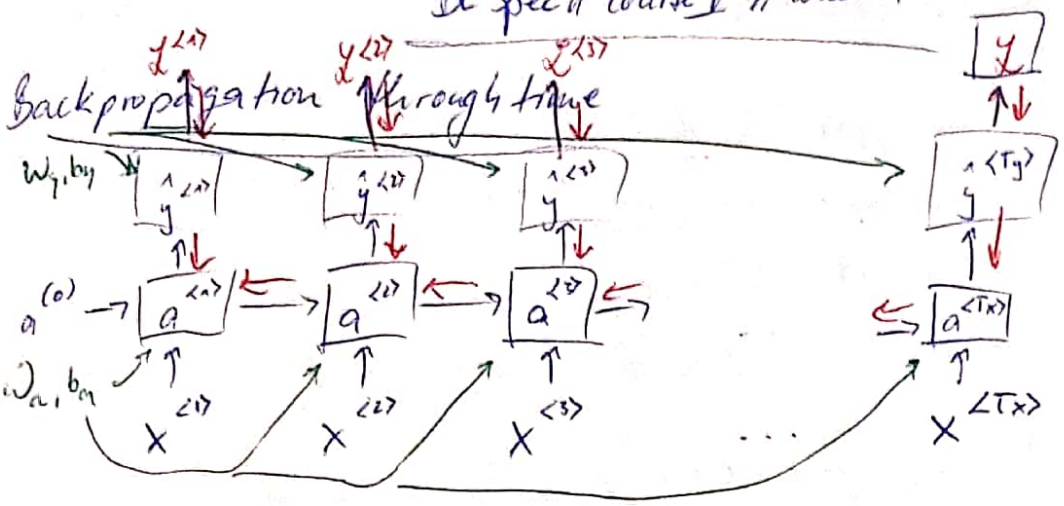
$$a^{<t>} = g(W_a \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} + b_a)$$

($\begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}$: 100 + 10'000 = 10'100)

$100 \begin{bmatrix} W_{aa} & W_{ax} \end{bmatrix} = W_a$ (stacked horizontally)

$100 \quad 10'000$

$$\begin{bmatrix} W_{aa} & W_{ax} \end{bmatrix} \cdot \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = W_{aa} a^{<t-1>} + W_{ax} x^{<t>}$$



Backprop through time

Loss function elementwise:

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

(loss of single prediction in single time step)

overall loss for entire sequence:

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

Different types of RNNs

often $T_x \neq T_y$! or even different classes (in & output lengths)

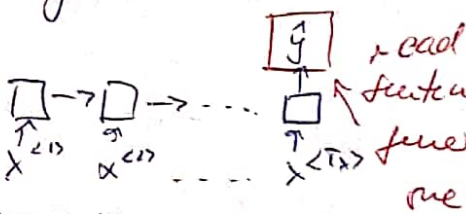
so far "many-to-many"

one-to-one (normal ANN)

Sentiment classification

x = text → suitable for binary class.

y = 0/1 ... 1 ... 5



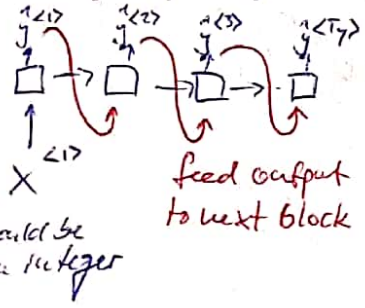
read the entire sentence and generate just one output

There is ... more "many-to-one"

"One-to-many"

Music generation

$x \rightarrow y^{\langle 1\rangle}, y^{\langle 2\rangle} \ldots y^{\langle T_y\rangle}$

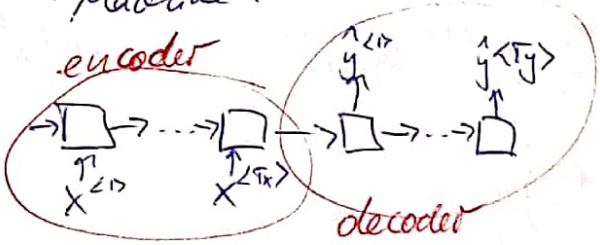


feed output to next block

x could be an integer

other type of "many-to-many" with different T_x, T_y - lengths

→ Machine translation



Language model and sequence generation

Speech recognition: The apple and pair salad (Sentence 1)

The apple and pear salad (Sentence 2)

Probability: $P(\text{Sentence 1}) = 3.2 \times 10^{-13}$

$P(\text{Sentence 2}) = 5.7 \times 10^{-10}$! model would pick this

$P(\text{Sentence}) = ?$ Language model $\rightarrow P(y^{\langle 1 \rangle}, \dots, y^{\langle T_y \rangle})$

Language modelling with RNN

Training set: large corpus of english text

"Cats average 15 hours of sleep a day" <EOS> — end of sentence as extra token

$y^{\langle 1 \rangle}$ $y^{\langle 2 \rangle}$ $y^{\langle 3 \rangle}$ .... $y^{\langle 9 \rangle}$

"Tokenize" 1st to do

Form a vocabulary, map each of the words to an one-hot vector

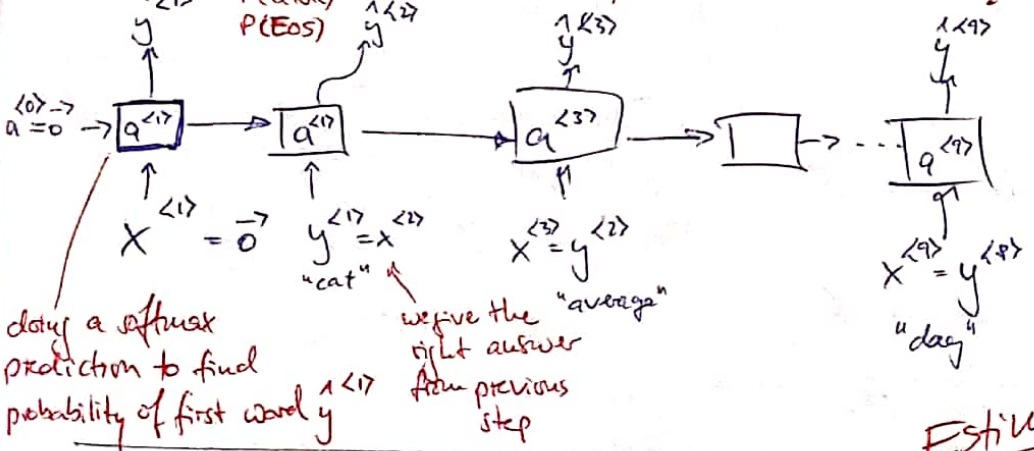
"The Egyptian ~~Mau~~ <UNK> is a breed of cat. <EOS>

<UNK> (unknown word)

RNN model

$P(a), P(\text{aaron}), P(\text{cat}) \dots, P(\text{zulu}), P(\text{UNK}), P(\text{EOS})$

$P(\text{average} \mid \text{cats})$ $P(\_ \mid \text{"cats average"})$ $P(\langle \text{EOS} \rangle \mid \dots)$



RNN learns to predict a word going from left to right

doing a softmax prediction to find probability of first word $\hat{y}^{\langle 1 \rangle}$

we give the right answer from previous step

Estimating: $P(y^{\langle t \rangle} \mid y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, \dots y^{\langle t-1 \rangle})$

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

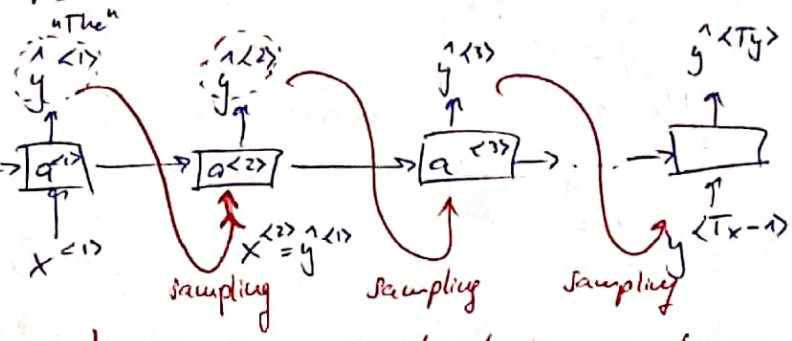
$$\mathcal{L} = \sum_t \mathcal{L}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle})$$

DL spec // course V // week 1

Sampling a sequence from a trained RNN

Probability output by RNN is not used to pick the highest probability word. Use probability output by RNN to randomly sample a chosen word for that time step as $\hat{y}^{\langle t \rangle}$. Then pass this word to the next time step!

Sampling: $a^{\langle 0 \rangle} = \vec{0}$ → $a^{\langle 1 \rangle}$ → $a^{\langle 2 \rangle}$ → $a^{\langle 3 \rangle}$ → ... → $\hat{y}^{\langle T_y \rangle}$



→ np.random.choice → randomly chosen sentence

Vocabulary = [a, aaron, ..., zulu, <UNK>]

or = [a, b, c, ..., z, ␣, ., ;, 0, ..., 9, A, ..., Z]

Characters

cat

$y^{\langle 2 \rangle}, y^{\langle 2 \rangle}, y^{\langle 3 \rangle}$

pro: ☺ don't have to worry about unknown tokens
☺ character-based language model is able to give "Mau" a non-zero probability!

con: ☹ much more and longer sequences
☹ difficult to capture long-range dependencies

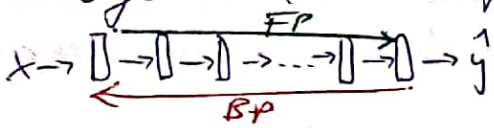
Vanishing gradients with RNNs (bigger problem than exploding gradients)

long-range dependencies "The cat, which ate a lot..., was full"
"The cats, ..., were full"
↳ disturbs predictability

↳ NaN
↳ gradient clipping answers exploding gradients

updates may become so large that they overshoot the optimal value during back propagation

Vanishing gradient problem arises for very deep NNs as $\hat{y}$ has a hard time to backpropagate to layer $a^{[1]}$ (activation layer 1)



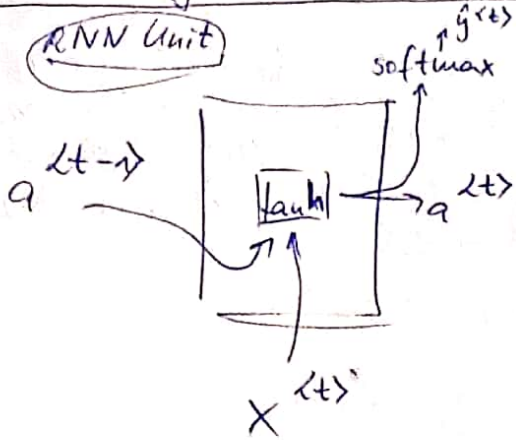
Gated Recurrent Unit (GRU) → helps with vanishing gradients

[Cho, 2014 On the properties of mach. translation: Encoder-decoder...]

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

[Chung, 2014 Empirical Evaluation of Gated RNN...]

RNN Unit



GRU (simplified) c = memory cell

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

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

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

will decide if updated or not! is either 0 or 1

update

element-wise mult.

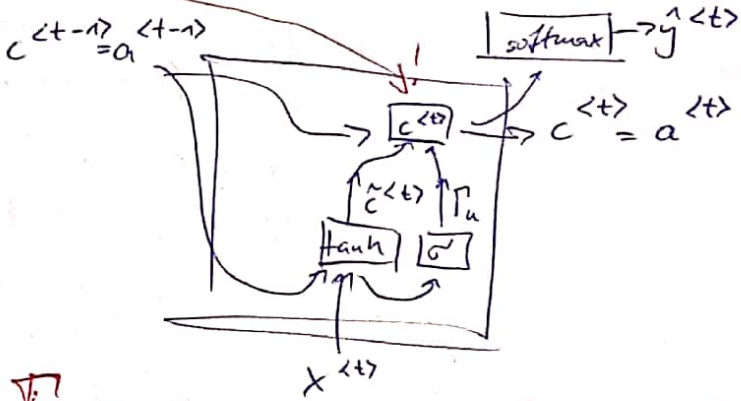
sigmoid

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

$\Gamma_u = 1$ ⇒ $c^{\langle t \rangle} = 1$

$\Gamma_u = 0$ .....

The cat, which already ate ..., was full (=1)



Γ_u can be very close to zero (0.000001) thus it is not suffering from vanishing problem!

↳ enables very long-range dependencies!

↳ For signal to BP without vanishing we need $c^{\langle t \rangle}$ to be highly dependent on $c^{\langle t-1 \rangle}$

Full GRU

Γ_r = relevant

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

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

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

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

VI

Long Short Term Memory (LSTM) ⇒ for long distance learning, more powerful than GRU

GRU

candidate for update

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

update gate

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

relevance gate

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

candidate

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

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

Γ_f

[Hochreiter & Schmidhuber 1997, LSTM]

GRU came up later actually and is a simplification

LSTM

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

update

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

forget

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

output

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

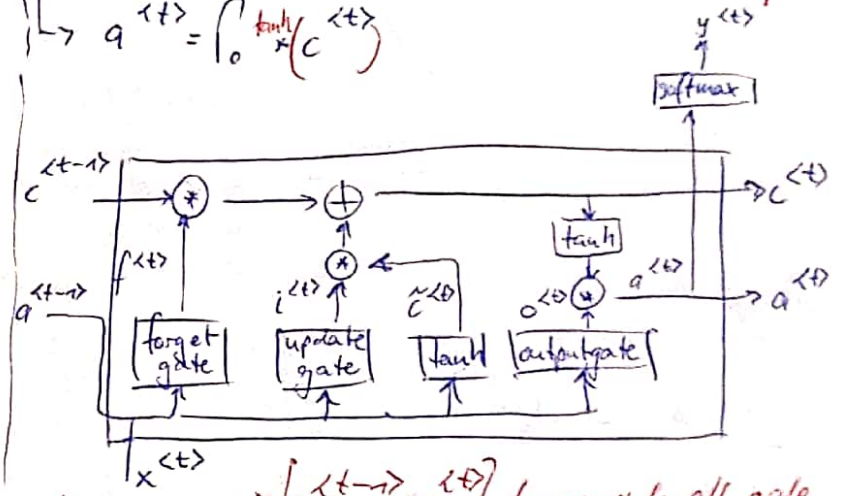
instead of having one update gate controlling both terms ⇒ 2 separate terms

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

→ gives c the option to keep the old value $c^{\langle t-1 \rangle}$ and then add it to new $\tilde{c}^{\langle t \rangle}$

↳ separate update and forget gates!

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



key takeaways → $\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right]$ to compute all gate values

→ get $c^{\langle t \rangle}$ from $c^{\langle t-1 \rangle}$

peephole connection → adding earlier cell memory

$\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}, c^{\langle t-1 \rangle}\right]$

⇒ more flexible and powerful

Γ_u is a vector of dimension equal to the number of hidden units in the LSTM

In literature "update" subscript is referred to as "i" (and not "u")

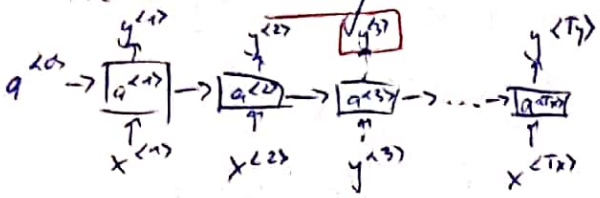
Bidirectional RNN

→ commonly BRNN with LSTM for NLP

Getting information from the future

He said, "Teddy bears are on sale!"

He said, "Teddy Roosevelt was a great President"

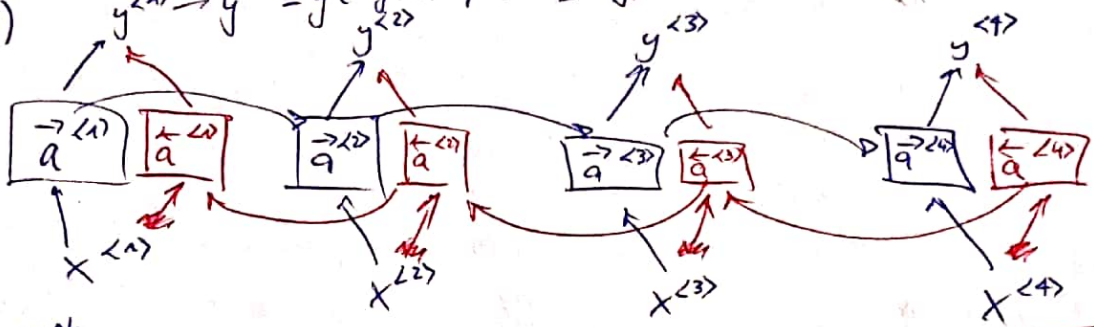


→ unidirectional RNN
GRU
LSTM

"Teddy" → can't tell if it's the name or a bear

Bidirectional RNN (BRNN)

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



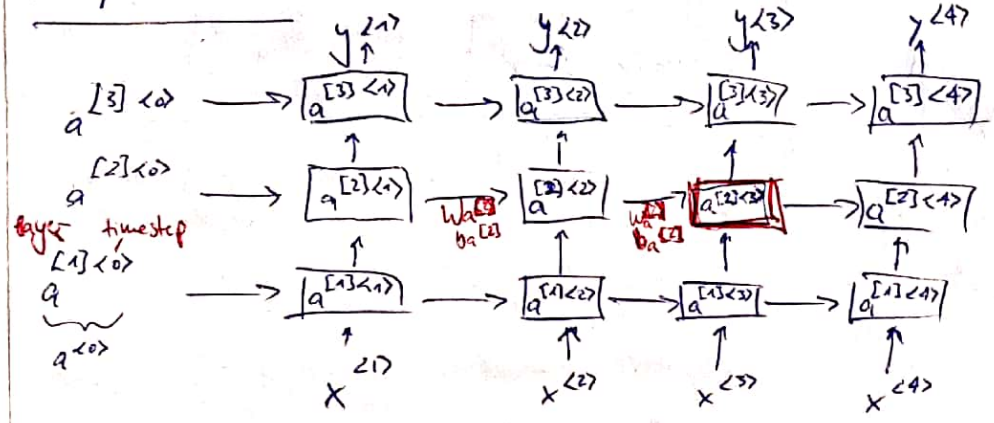
Acyclic graph

↳ allows to include past and future parts into predictions in the middle of a sequence

→ contra: need entire sequence before a prediction can be made

↳ not for real-time NLP!

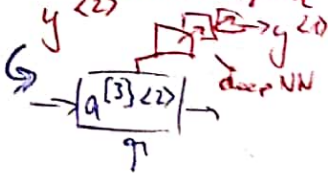
Deep RNNs



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

for RNNs 3 layers can be already a lot

But outputs like $y^{<2>}$ can be added from layer 2 / deep NN with not horizontal/ temporal connection and then compute $y^{<2>}$



DL spec // course V // week 2

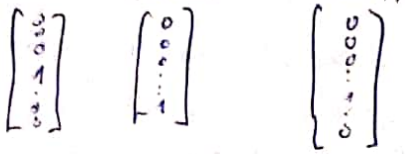
RNNs

NLP and word Embeddings

Word representation

$V = [a, aaron, \dots, zulu, <UNK>]$ $|V| = 10000$ → One-hot representation

Man (5391) Woman (9853) Orange (6257)



Weakness of this representation:
Treats each word as a thing itself and enables no cross-relation

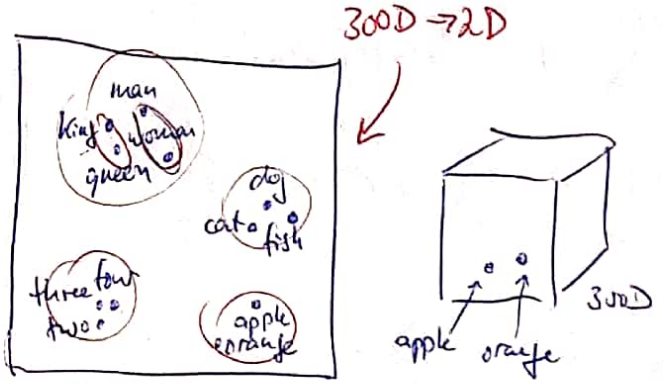
"I want a glass of orange juice."
"I want a glass of apple ?"
} algorithm does not know that apple juice and orange juice are both popular things

because the inner product of the one-hot vectors of "apple" and "orange" is zero!

Featurized representation: word embedding

	Man (5391)	Woman (9853)	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
:						
size						
cost						
action						
:						
features 300	e_{5391}	e_{9853}				

$\times E$ (300, 10000) → 300



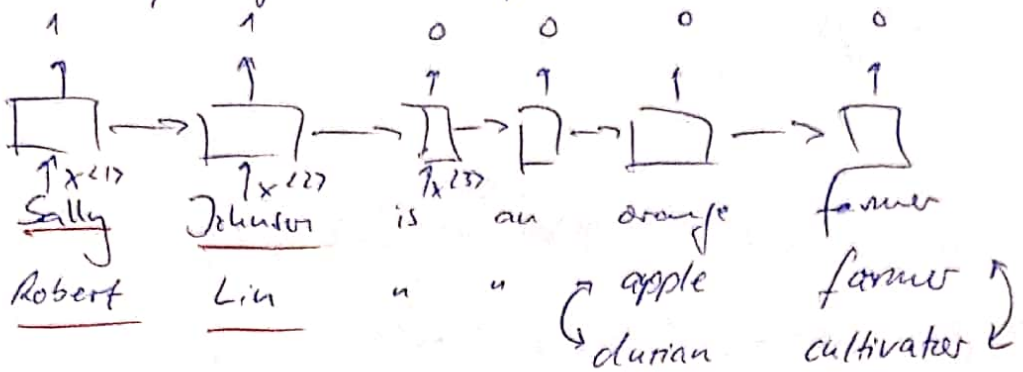
t-SNE
[van der Maaten and Hinton, 2008, Visualizing data t-SNE]

Embeddings group reasonable words together

One-hot vectors do not do a good job at capturing similarities between words (every 1-hot-vec has the same Euclidean distance)

Using word embeddings

Named entity recognition example



Gallows to carry out transfer learning examing lots of unlabeled text (from internet download)

→ 1 billion words – 100 B words

→ 100k words ← pass to smaller labeled data-set

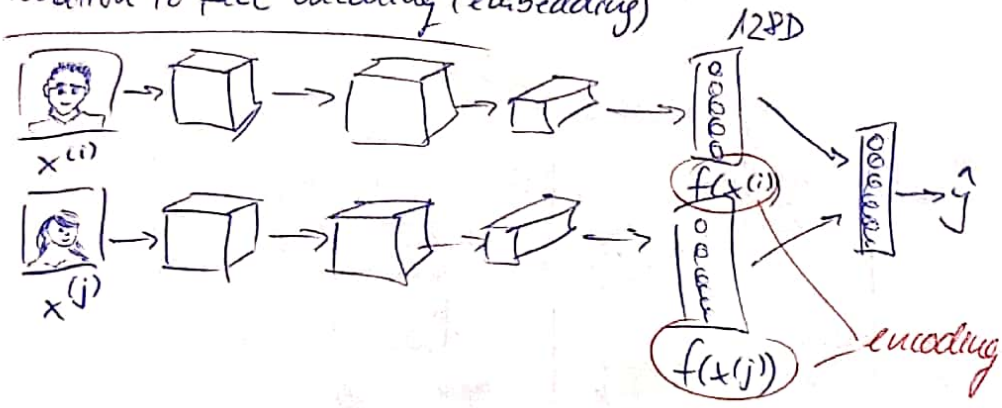
BRNN

Transfer learning and word embeddings

A → B

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 (100k words)
   ⇒ pro: can use lower-dim feature vectors 10'000 one-hot → 300-dim vector dense vector
3. Optional: Continue to finetune the word embeddings with new data

Relation to face encoding (embedding)



difference:

In face recognition we want to train a NN to classify an image that wasn't seen before and in NLP we work with a fixed vocabulary

DL spec // course V // week 2

Properties of word embeddings

[featurized matrix]

	Man	Woman	King	Queen	...
gender	-1	1	-0.95	0.97	
Royal	0.01	0.02	0.93	0.95	
Age	0.03	0.02	0.70	0.69	
Food	0.09	0.01	0.02	0.01	
	e_{man}	e_{woman}			

[Mikolov, 2013, Linguistic regularities in continuous Analogy reasoning space word representation]

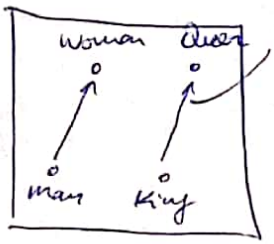
$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

$e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ } common difference is the gender

Man → Woman as King → ? Queen

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

Analogies using word vectors



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

high similarity

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

similarity

accuracy reported 30% - 75%

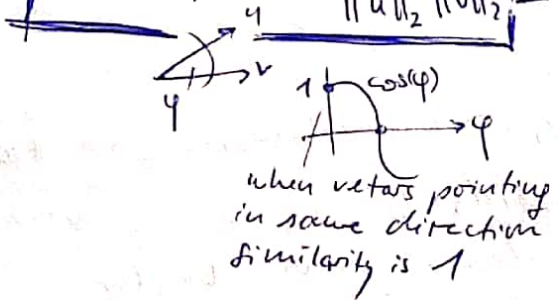
Cosine similarity → Measure similarity between 2 words

$\text{sim}(e_w, e_{king} - e_{man} + e_{woman})$

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

inner product tends to be large when similarity is high

Euclidean lengths



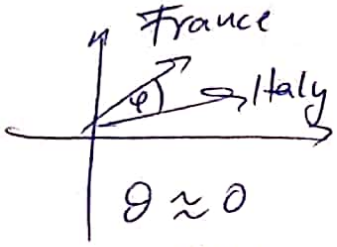
when vectors pointing in same direction similarity is 1

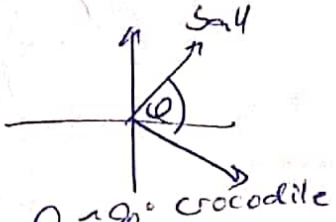
$\|u - v\|^2$ → measure of dis-similarity

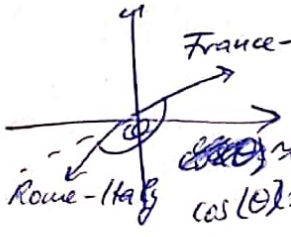
word embedding algorithm can learn:

man:woman as boy:girl

big:bigger as tall:taller

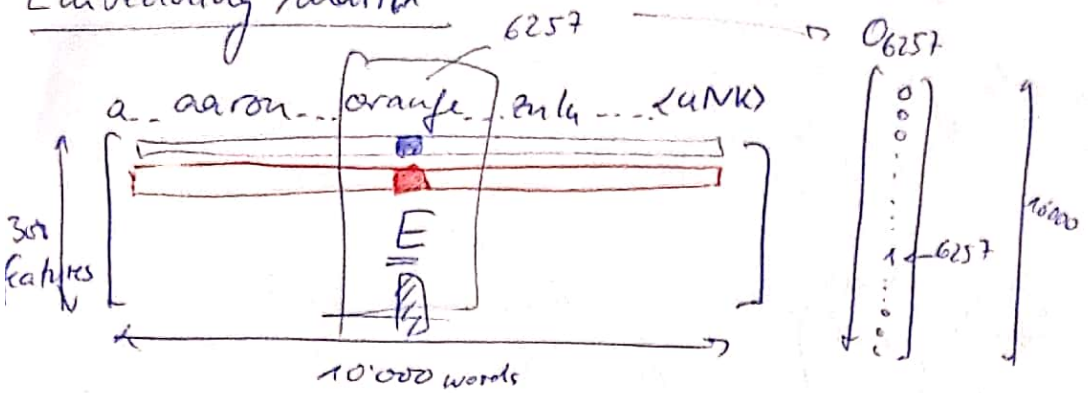






vectors similar but encode country and city the other way round

Embedding matrix



$E \cdot O_{6257} = e_{6257}$ e_w = embedding for word w

(300,10000) (10000,1) (300,1)

$E \cdot O_j = e_j$ (Initialize E randomly)

= embedding for word j

In practice use specialized function to look up an embedding

Keras → Embedding layer calls column of interest

Learning Word Embeddings: Word2vec & GloVe

[Bengio 2003, A neural probabilistic LM]

I want a glass of orange juice .

4343 9665 1 3852 6163 6257

I want a glass of orange

one-hot-vec matrix embed.vector

$O_{4343} \rightarrow E \rightarrow e_{4343} = E \cdot O_{4343}$

e_{9665}, e_1, e_{3852}, e_{6163}, e_{6257} → 1800 dim-layer → Softmax 10'000 possible outputs → "juice"

$W^{[1]}, b^{[1]}$ $W^{[2]}, b^{[2]}$

using 300-dimensional word embeddings → 6 words → 6×300 = 1800 dim-vector

→ in practice look at four previous words to make a prediction: 4 × 300 → 1200 dimensional vector

↳ can deal with sentences of any size

DL spec // course V // week 2

Other context/target pairs — Skipgram model

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

context: orange — target: juice

context:
- last four words. "a glass of orange ___ ".
- 4 words on left & right — " — ___ to go along with "
- last 1 word
- Nearby 1 word

Word2vec - algorithm [Mikolov 2013, efficient estimation of word representations in vector space)

Skip-grams

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

Context (randomly chosen)	Target
orange	juice (1 w. left/r)
"	glass (2 before)
"	my (...)

Model details

Vocab size = 10'000k

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

6257 ("orange"), 4834 ("juice")

one-hot vector: $O_c \rightarrow E \rightarrow e_c = E \cdot O_c \longrightarrow O \xrightarrow[\text{Softmax}]{} \hat{y}$

E has a lot of parameters

Softmax: probability of t parameterized by c

$$p(t \mid c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10'000} e^{\theta_j^T e_c}}$$

θ_t = parameter associated with output

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{10'000} y_i \log \hat{y}_i$$

$\hat{y} \in \mathbb{R}^{10'000}$ (probabilities for all target words)

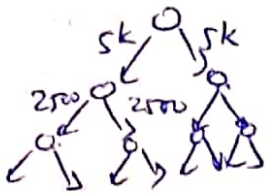
y = one-hot vector with a 1 at position 4834

Problems with softmax classification

$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10'000} e^{\theta_j^T e_c}}$$

→ has to sum overall words in vocab in each iter.

↳ solution: **Hierachical softmax classifier**



→ scales in $\log |V|$

→ common words rather on top and vice versa

How to sample context c?

→ the, of, a, and, to, ...

→ separate common from uncommon words

Negative sampling [Mikolov 2013, Distributed representations of words and phrases and their compositionality]

Defining a new learning problem

↳ have one positive example and chosen k-negative words to train the model more efficiently

context (given x)	word	target (y)
orange	juice	1
orange	king	0
"	book	0
"	the	0
"	of	0

"orange" "juice"

→ want to predict if this is a target pair

k

taken randomly from vocab.

label them!

How to sample negative case words?

→ empirical frequency $f(w_i)$ in language

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10'000} f(w_j)^{3/4}}$$

$k = 5-20$ for smaller datasets

$k = 2-5$ larger datasets

context (x)	word	target (y)
orange	juice	1
"	king	0
"	book	0
"	the	0
"	of	0
c	t	y

for every positive example we have k-negative examples to train the model

reduces computation cost

Model

$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10'000} e^{\theta_j^T e_c}}$$

logistic regression-like model

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

sigmoid

$O_{6257} \rightarrow E \rightarrow e_{6257}$ → juice, king → 10'000

GloVe word vectors [Pennington 2014, GloVe: Global vectors for word representation]

c, t

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

(i: c, j: t)

$X_{ij} = X_{ji}$ - symmetrical relationship

Model

now symmetric

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

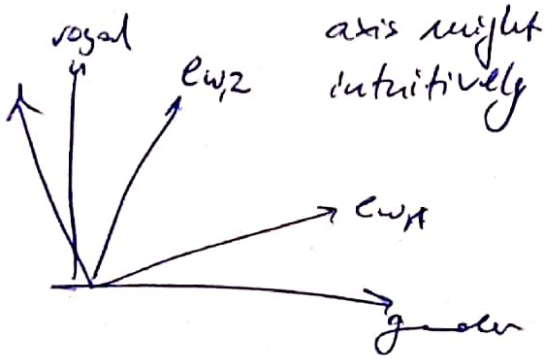
(θ: t, e: c, b_j': prime)

- weighting term: $f(X_{ij}) = 0$ if $X_{ij} = 0$ ("$0 \log 0 = 0$")
- "$\theta_t^T e_c$": inner product of indicator
- how related are c and t?

θ_i, e_j are symmetric

- one way to train the algorithm is to initialize both randomly and run GD and the for every word take the average

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



axis might not be intuitively aligned

Sentiment classification

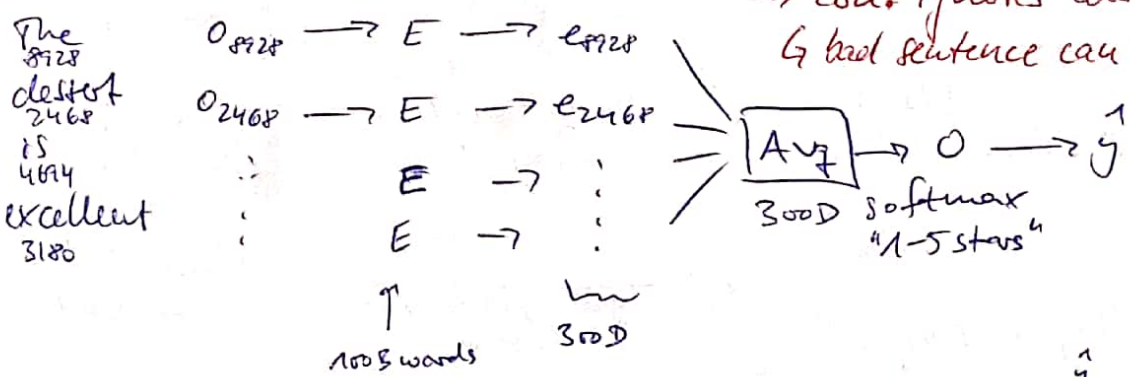
restaurant ratings x —map→ y

The dessert is excellent	4 stars
Service was slow	2 stars
Nothing special	3 stars
Bad	1 stars

– "completely lacking in good taste, good service, and good ambience"

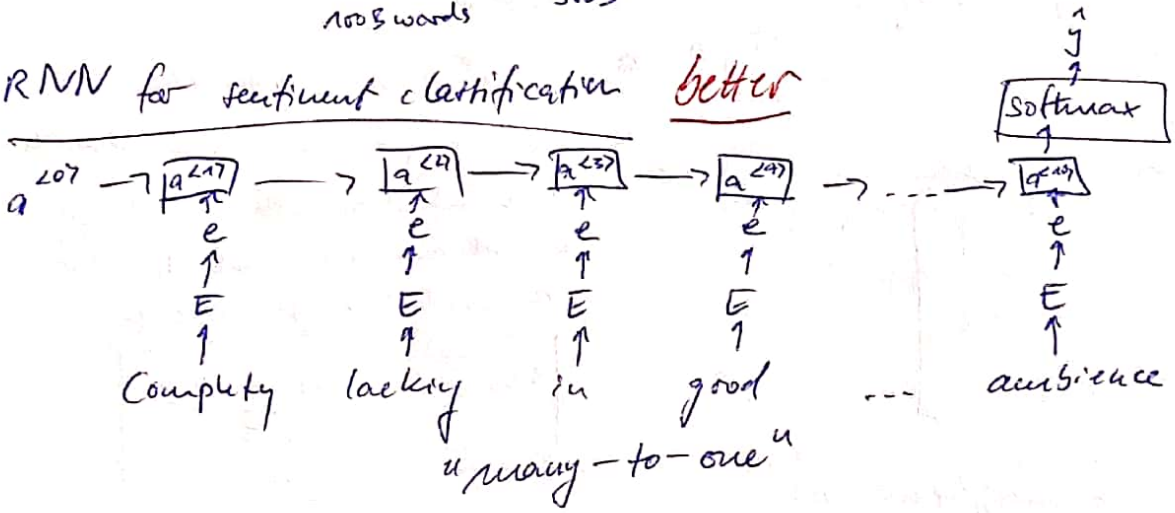
Simple sentiment classification model

→ con: ignores words' order
Bad sentence can have the word "good" often



RNN for sentiment classification

better



DL spec // course IV // week 2

Debiasing word embeddings [Bolukbasi, 2016: Man is to computer programmer as woman is to homemaker? Debiasing word embeddings]

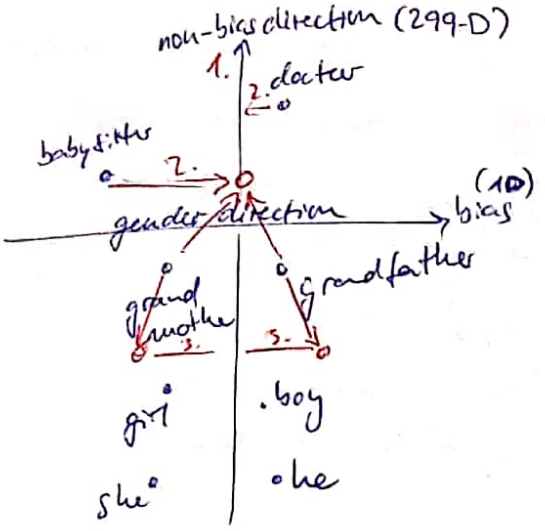
here bias: gender, ethnicity ...

Man : Woman as King : Queen

Man : Computer_Programmer as Woman : homemaker ✗ ☹

Father : Doctor as Mother : Nurse ✗ ☹

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



1. identify bias direction

$\begin{cases} e_{he} - e_{she} \\ e_{male} - e_{female} \end{cases}$

→ average them → find out gender direction

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

project "doctor", "babysitter" to axis to reduce/eliminate gender component

3. Equalize pairs (hand-pick)

grandmother — grandfather

girl — boy

→ move grandmother/father to that distance to non-bias axis is equal!

DL Spec // course V // week 3

Basic Models — Sequence models & Attention mechanisms

[Sutskever 2014 seq to seq learning. w NN]
[Cho 2014, learning phrase representations using RNN encoder-decoder for statistical mach. trans.]

Sequence to sequence model

$x^{\langle 1\rangle}$ $x^{\langle 2\rangle}$ $x^{\langle 3\rangle}$ $x^{\langle 4\rangle}$ $x^{\langle 5\rangle}$
Jane visite l'Afrique en septembre

→ Jane is visiting Africa in September.
$y^{\langle 1\rangle}$ $y^{\langle 2\rangle}$ $y^{\langle 3\rangle}$ $y^{\langle 4\rangle}$ $y^{\langle 5\rangle}$ $y^{\langle 6\rangle}$

first encoder Network

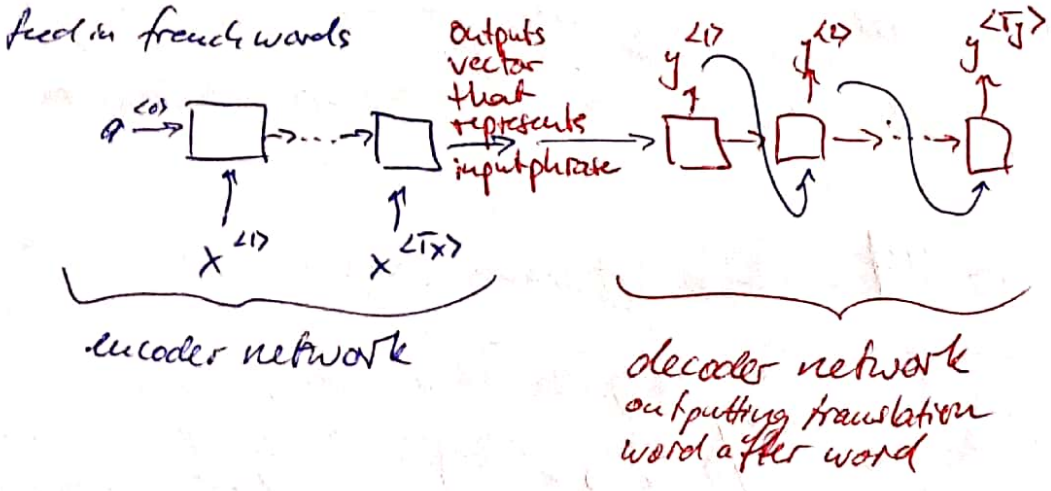


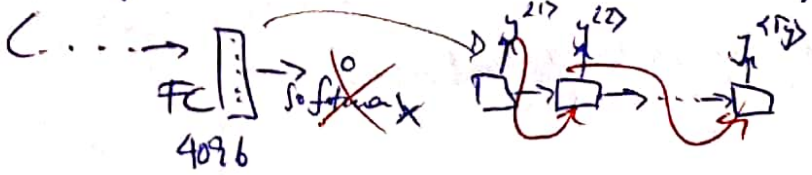
Image captioning

$y^{\langle 1\rangle}$ $y^{\langle 2\rangle}$ $y^{\langle 3\rangle}$ $y^{\langle 4\rangle}$ $y^{\langle 5\rangle}$ $y^{\langle 6\rangle}$ $y^{\langle 7\rangle}$
A cat is sitting on a chair

[Mao, 2014, Deep captioning with multimodal RNN]
[Vinyals 2014, 1st Neural image caption generator]
[Karpathy, FeiFei, 2015, Deep visual-semantic alignments for generating image descriptions]

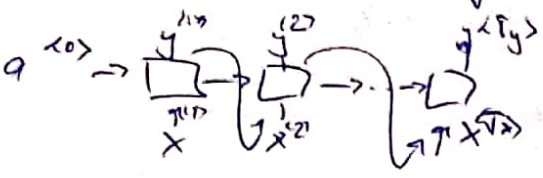
→ use a pre-trained AlexNet CNN → learn features/encoding

→ get rid of softmax and keep the 4096 FC

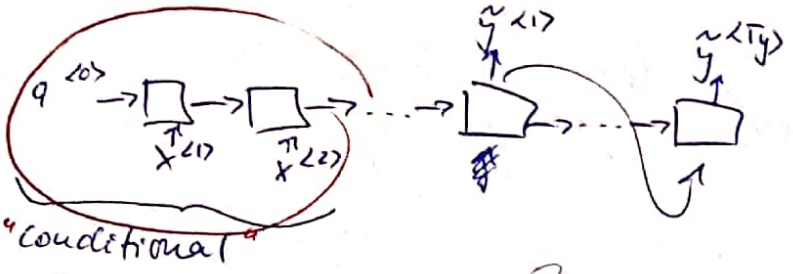


Picking the most likely sentence

Machine translation as building a conditional language model (LM)

LM: 

→ LM gives a probability of a sentence (generate novel sentences as well)

Machine translation: 

"conditional language model"

↳ trying to estimate the probability of some english translation

Instead of modelling the probability of a sentence

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

it is now modelling the probability output english translation conditioned on some input french sentence

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

(English | French)

→ do not use random sampling!

↳ rather: find English sentence y that maximizes that conditional probability

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

Need an algorithm that can find the value y that maximizes

→ Beam Search

Why not a greedy search?

→ generate the first most likely word according to conditional language model
→ after picking 1st word pick the 2nd which is most likely and so on

} $P(y^{\langle 1 \rangle}|x)$

what want is the sequence joint probability $P(y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, \dots, y^{\langle T_y \rangle}|x)$

(A) → "Jane is visiting Africa in September"
(B) → "Jane is going to be visiting Africa in September"

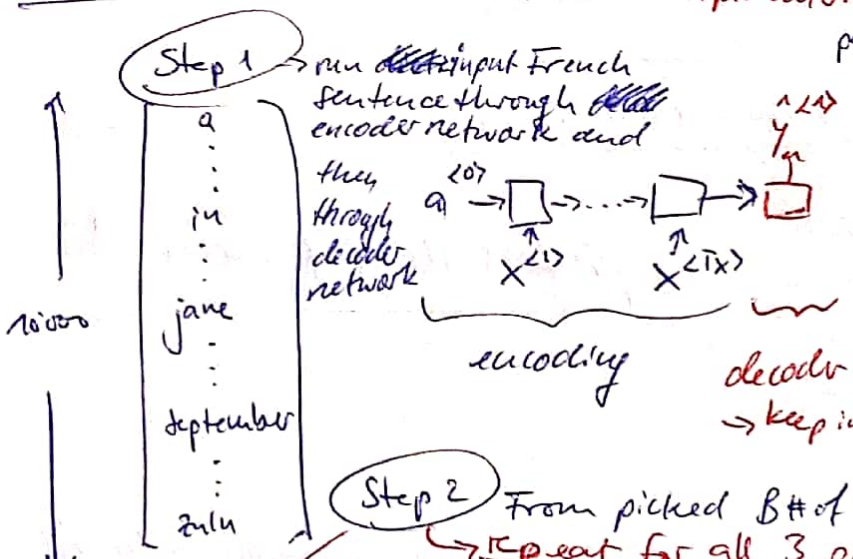
↳ is going is more likely than "is visiting" so the 2nd phrase will be chosen which is not optimal and only because of the word-by-word-prediction.

$P(A|x) < P(B|x)$

total # of words in an English sentence is exponentially large
If 10'000 w in dict. and a sentence of 10 words long
then there are $10'000^{10}$ possibilities
↳ so the most common thing to do is use an approximate search algorithm.
→ will try to pick the sentence y that maximizes the conditional probability

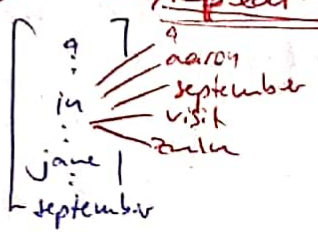
Beam Search → can consider multiple alternatives compared to greedy search

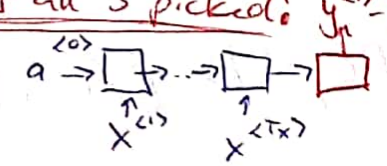
parameter
$B = 3$ (beam width)
↳ 3 alternatives at a time
↳ in jane september



Step 2 From picked B # of words we go on to pick next
↳ repeat for all 3 picked!

considering
$B \cdot 10'000$
$= 30'000$ w



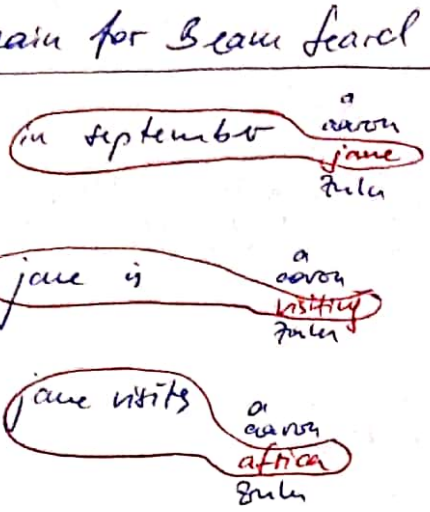


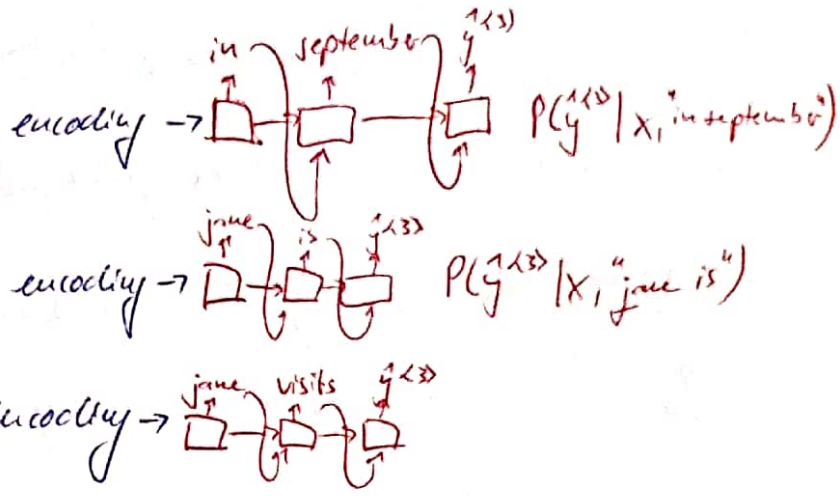
$\hat{y}^{\langle 2 \rangle}$ = "in" $\hat{y}^{\langle 1 \rangle}$ $\in P(y^{\langle 1 \rangle}|x, \text{"in"})$

↳ $P(y^{\langle 1 \rangle}, y^{\langle 2 \rangle}|x) = P(y^{\langle 1 \rangle}|x) \cdot P(y^{\langle 2 \rangle}|x, y^{\langle 1 \rangle})$ (3)

If Beam search decides that 3 most likely are not including "september" as starting word it is rejecting "september" as first word for translation.

again for Beam search (B=3)





$\longrightarrow P(y^{\langle 1\rangle}, y^{\langle 2\rangle}|x)$

⇒ hopefully outcome is "jane visits africa in september" <EOS>

Refinements to Beam search

Beam search:

Length normalization:

$$\arg\max_{y} \prod_{t=1}^{T_y} P(y^{\langle t\rangle}|x, y^{\langle 1\rangle}, \dots, y^{\langle t-1\rangle})$$

(product of probabilities)

In practice:

$$\arg\max_{y} \sum_{t=1}^{T_y} \log P(y^{\langle t\rangle}|x, y^{\langle 1\rangle}, \dots, y^{\langle t-1\rangle})$$

inserting a log here turns the product of a log into a sum of a log and increases numerical stability

maximizing $\log P(y|x)$ should give the same as $\max P(y|x)$

prefers short sentences!

next modification

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

$\alpha = 0.7$

normalization reduced penalty significantly for outputting longer sentences

DL specialization course II / week 3

Beam Search discussion

Beam width B ? → size of B steers computational time

large B: better results bc more probs but slower

small B: worse results, faster

~ 10

up to 100

in research > 1000

Error Analysis in Beam Search (BS)

Jane visite l'afrique en septembre

Human: Jane visits Africa in September (y^*) good translation!

Algorithm: Jane visited Africa in sept. ($\hat{y}$) bad, changed meaning!

encoder → decoder → RNN → Beam Search

↳ B ↑ weakness from ANN error analysis that getting more training data never hurts so B ↑, BUT might not help always.

↳ RNN computes $P(y|x)$

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

Case 1: $P(y^*|x) > P(\hat{y}|x)$: BS chose $\hat{y}$. But y^* attains higher $P(y|x)$

↳ BS failed

Case 2: $P(\hat{y}|x) \geq P(y^*|x)$: y^* is a better translation than $\hat{y}$ but RNN predicted $P(y^*|x) < P(\hat{y}|x)$

↳ RNN failed

Error Analysis Process → go through devset and find mistakes

Human	Algorithm	$P(y^* \mid x)$ (RNN)	$P(\hat{y} \mid x)$ (BS)	At fault?
Jane …	Jane …	2×10^{-10}	1×10^{-10}	BS
…	…	…	…	RNN
		⋮	⋮	BS
				BS
				RNN
				⋮

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

(RNN generates objective function!)

Bleu score (optional) [Papineni 2002, A method for automatic evaluation of machine translation]

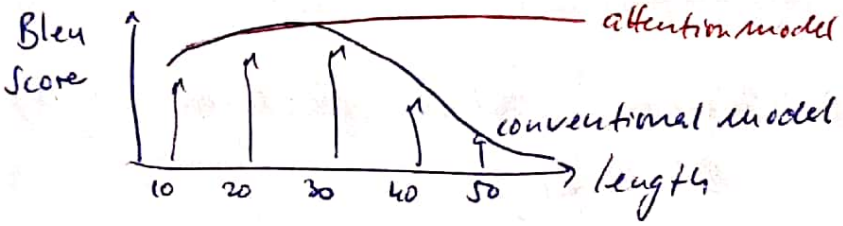
Bilingual evaluation understudy → how good is machine translation

---

Attention Model intuition [Bahdanau 2014, Neural MT by jointly learning to align + translate]

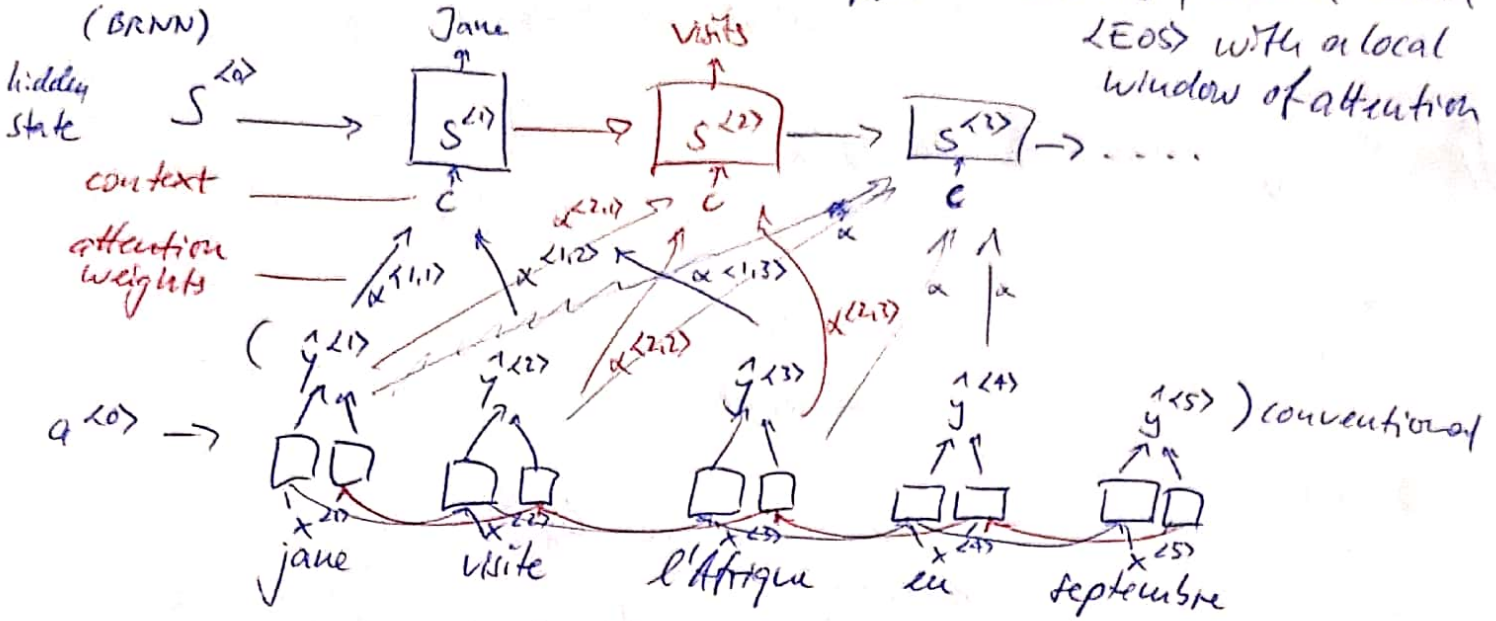
input very long French sentence → encoding → decoding (memorize sentence) + store in activations & generate translation

→ human would not memorize whole sentence but a few words at a time + translate "live"



Bi-directional RNN here:
(BRNN)

RNN marches forward until <EOS> with a local window of attention



contexts change with advancing words to translate

Attention Model → pay attention to a part of a sentence at a time

parts from BRNN ← forward, backward

$$a^{\langle t' \rangle} = (\vec{a}^{\langle t' \rangle}, \overleftarrow{a}^{\langle t' \rangle})$$

Each step generates a new context

$$\sum_{t'} \alpha^{\langle 1,t' \rangle} = 1$$

$$c^{\langle 1 \rangle} = \sum_{t'} \alpha^{\langle 1,t' \rangle} a^{\langle t' \rangle}$$

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

Computing attention $\alpha^{\langle t,t' \rangle}$ [Bahdanau ...]

[Xu, 2015, Show attention and tell: neural image caption generation with visual attention]

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

use softmax to sum to 1

How compute vectors $e^{\langle \cdot \cdot \rangle}$?

→ Use small NN

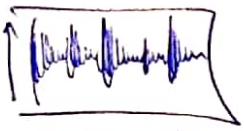
$s^{\langle t-1 \rangle}$, $a^{\langle t' \rangle}$ → [NN] → $e^{\langle t,t' \rangle}$

Con: takes quadratic time to run ☹

Attention examples July 20th 1969 → 1969-07-20
23 April, 1564 → 1564-04-23

Speech recognition - Audio data

x: audio clip → y: transcript



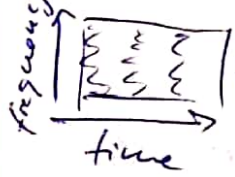
time

"the quick brown fox"

earlier:
↳ phonemes: de kwik braun

↳ today direct processing is possible due to much larger datasets
academic datasets: 300h - 3000h (transcripted data)
best commercial systems: 100'000h

human ear identifies intensities of frequencies

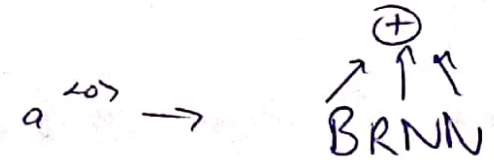


How build speech recognition?

---

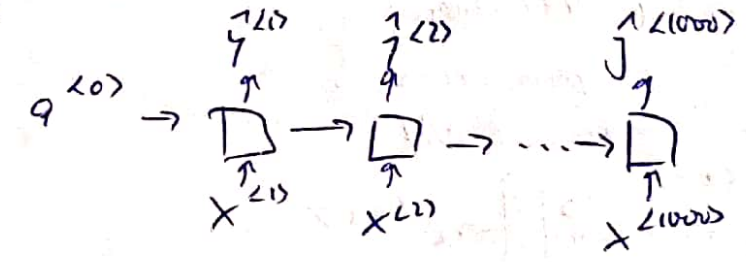
could do:

Attention model → output transcript



what works well is: CTC cost for speech recognition (Connectionist temporal classification)
[Graves 2006, CTC "Labeling unsegmented sequence data with RNN]

"the quick brown fox"



CTC cost allows

ttt_h_eee___␣__qqq...

↳ correct output for "the quick brown fox"

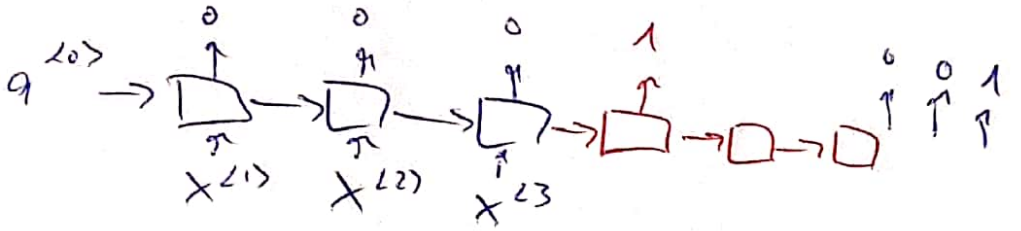
Rule: collapse repeated characters not

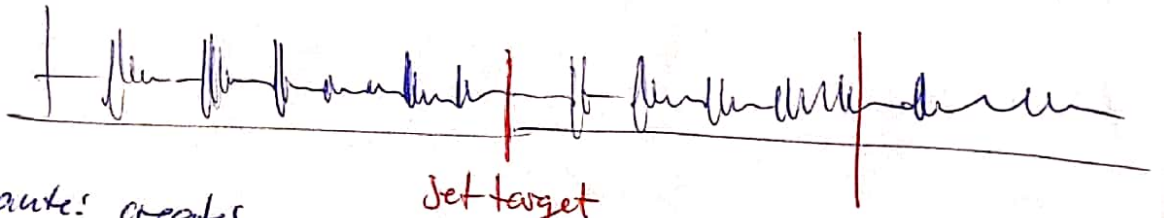
DL spec // course V // week 3

Trigger Word Detection

Alexa, Baidu, Hey Siri, OK Google

algorithm





disadvantage: creates unbalanced training set with more 0s than 1s → fix: 0000 111 000 111 → make it output more often

1 2 3

1 True ✗ false ✓

2 slowly memory better ✓ → " ✓

3 True ✓ → " ✓

4 No, RNN ✓ → " ✓

5 True ✓ ✓

6 sum over t' ✗ notall ... for t' ✗ $\Sigma t'$ + ✓ $\Sigma t'$ & ... for t

7 True ✓ → ✓ ✓

8 large ✓ → ✓ ✓

9 cokbok ✗ → cookbook ✓

10 t-th input ✗ → Features ✓