Sequence Models

3/5/25

Week One 1/12 Why Sequence Models?

Sequence data/problems

- speech
- music
- text
- sentiment classification
- DNA sequences
- translation
- video activity recognition
- name entity recognition

Week One 2/12 Notation

e.x. Name entity recognition

x: Harry Potter and Hermione Granger went on a walk.

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

y: 1 1 0 1 1 0 0 0 0

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

$T_x = 9$ $T_y = 9$ > sequence length

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

Representing words in Natural Language Processing (NLP)

Vocabulary

a	1
aaron	2
and	367
harry	4075
potter	6830
zulu	10,000

<unk>

x: Harry Potter and Hermione Granger went on a walk

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

4075 6830 367 10,000

one-hot

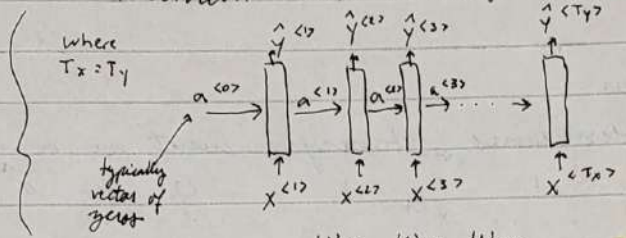
used to represent words not in your vocabulary

Week One 3/12 Recurrent Neural Network Model

Why not a standard network? 2 main problems.

1) inputs, outputs can have different lengths
2) doesn't share features learned across different positions of text

Recurrent Neural Networks



forward prop

W_{ax} W_{aa} W_{ya}

↑ same params used each time step

limitation: uses info from earlier in input but not later

Solution: bi-directional

Forward propagation:

$a^{<0>}$ = vector of zeros

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

$\hat{y}^{<1>} = g(W_{ya}\, a^{<1>} + b_y)$ ← g: sigmoid, softmax, depends on what task is

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

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

Simplified notation

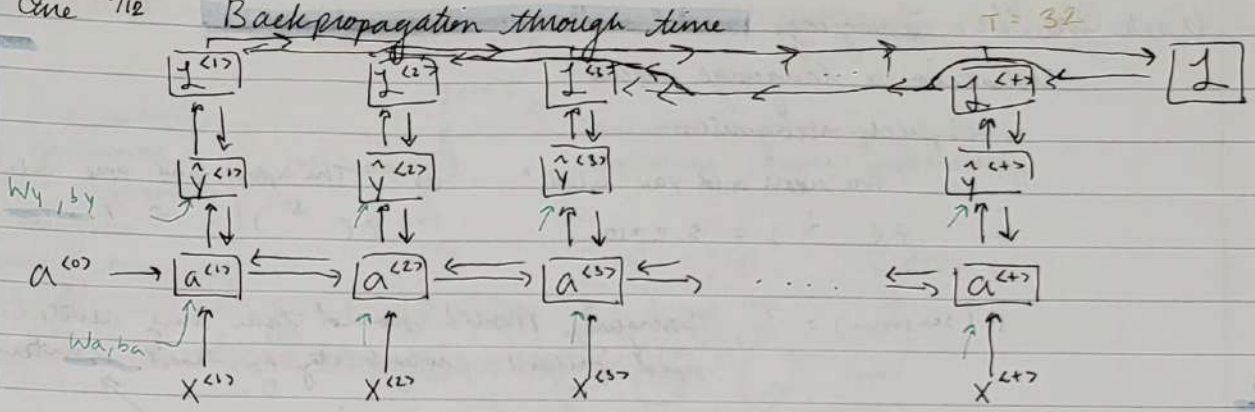
$a^{<t>} = g(W_a [a^{<t-1>}, x^{<t>}] + b_a)$

$W_a = [W_{aa} \vdots W_{ax}]$ ← stack matrices W_{aa} and W_{ax} horizontally to create W_a matrix

$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ \cdots \\ x^{<t>} \end{bmatrix}$ ← stack vectors on top of each other

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

Week One 4/12 Backpropagation through time

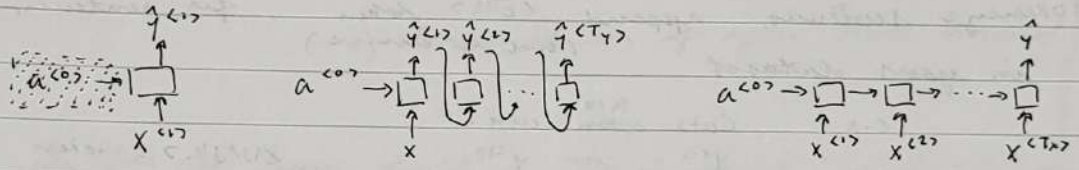


$$\mathcal{L}^{\langle t\rangle}(\hat{y}^{\langle t\rangle}, y^{\langle t\rangle}) = -y^{\langle t\rangle} \log \hat{y}^{\langle t\rangle} - (1 - y^{\langle t\rangle}) \log (1 - \hat{y}^{\langle t\rangle})$$

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

Week One 5/12 Different types of RNNs

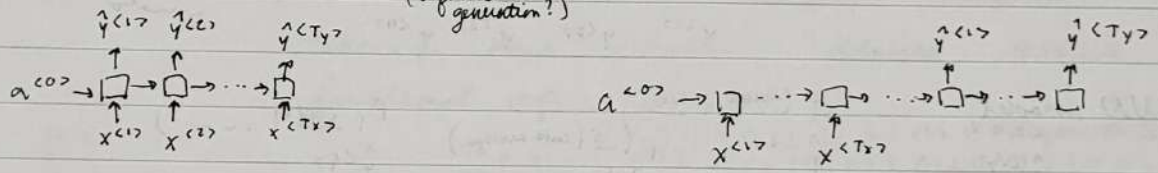
T = 41



one to one

one to many (sequence generation?)

many to one



many to many (where $T_x == T_y$)

many to many (when $T_x != T_y$)

Week One 6/12 Language model and sequence generation

What is a language model?

Speech recognition

"The apple and pair salad." vs "The apple and pear salad"

$P(\downarrow) = 3.2 \times 10^{-13}$ $P(\downarrow) = 5.7 \times 10^{-10}$

$P(\text{Sentence}) = ?$, Language Model should take any sentence and output probability of that sentence

$P(y^{<1>}, y^{<2>}, \ldots, y^{<T_y>})$ i.e. sequence of words

Language model w/ an RNN

Training set: large corpus of English text

Tokenize sentences, (opt) append <EOS> token (end of sentence), for sentences in your dataset

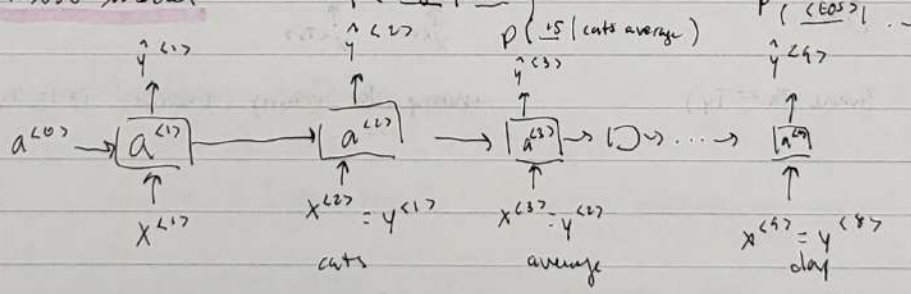
e.x. Cats ~~aren't~~ are cute.
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$

or

Cats are cute. <EOS>
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$

<UNK> = token for unknown or uncommon words

RNN model



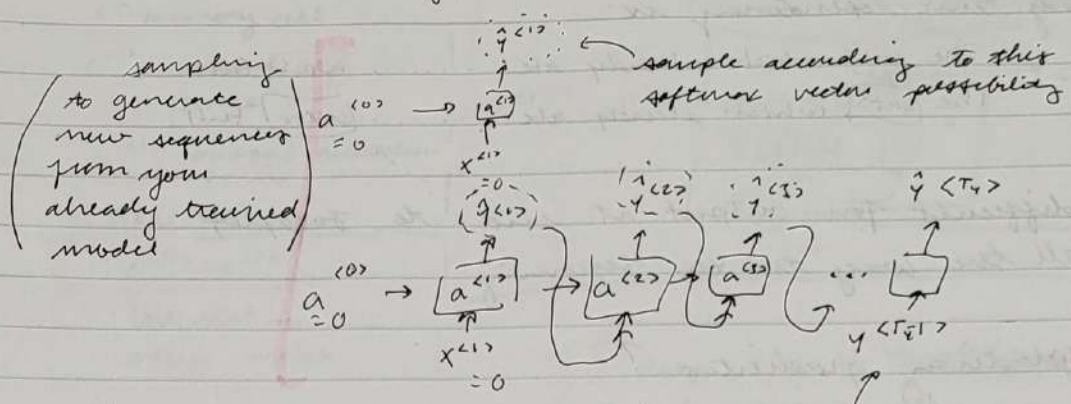
"Cats average 15 hours of sleep a day. <EOS>"

Cost function (Softmax loss)

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

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

Week One 7/12 Sampling Novel Sequences (in a RNN) 3/23/25



* model could output/generate unknown word token.

2 options

1) reject all output samples w/ unknown word token

2) leave it in if you don't mind

So far we have built a 'word level model' but you can also build 'character level model'

- tokens are chars, think ASCII, instead of words

e.x vocabulary: a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9 0 _ <(a space)
, ! @ # $ % ^ & * ( ) - _ + = [ ] { } ? . . . . .
(so on & so forth)

Pros & Cons

⊕ no unknown word tokens

⊖ much longer sequences

↓

char level models not as good at capturing how early parts of sentences affect latter parts

⊖ more computationally expensive

Week One 8/12 Vanishing Gradients w/ RNNs

long term dependency ex:

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

The cats which already ate ..... were full

- difficult for output at end to backprop all the way to the beginning

Exploding gradients ?

↓

gradient clipping.

- rescale some gradient vectors to be 'clipped' according to some max value

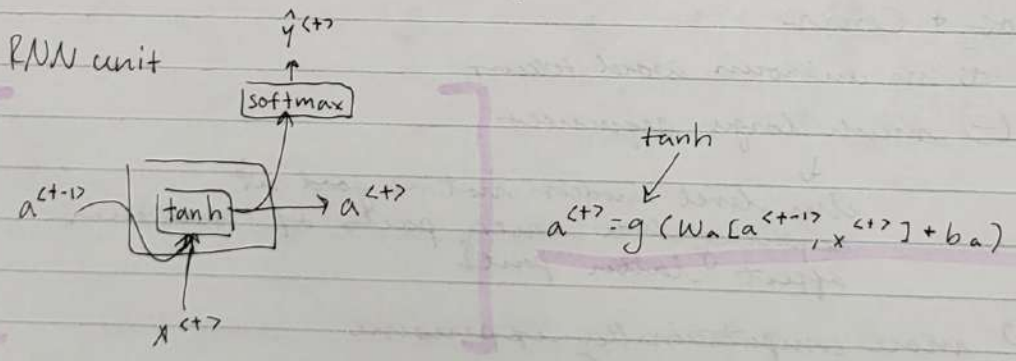
Vanishing gradients ?

↓

harder to deal w/ ∴

Week One 9/12 Gated Recurrent Unit (GRU)

- a mod to RNN hidden layer
  - helps capture long-range connections & deal w/ vanishing gradient issue



GRU ~~unit~~ (simplified)

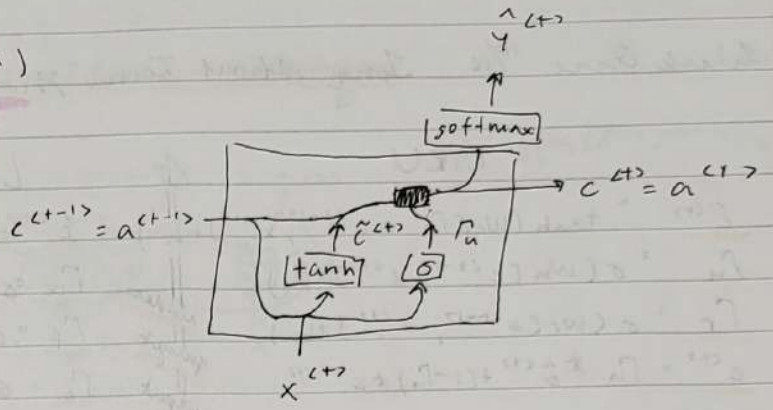
c = memory cell

$c^{\langle t \rangle}$ = value of c @ time t

$a^{\langle t \rangle}$ = output activation value

in GRU

- $c^{\langle t \rangle} = a^{\langle t \rangle}$, but not in other units.



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

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

Γ_u = gate Γ = Gamma "update"

$\hat{c}$ = candidate to replace $c^{\langle t \rangle}$

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

$c^{\langle t \rangle}$, $\hat{c}^{\langle t \rangle}$, & Γ_u can be vectors these are element wise multiplication in that case

Full GRU

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

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

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

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

↑ alt. notation

Week One 10/12 Long Short Term Memory (LSTM) unit T=94

GRU

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

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

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

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

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

update
target
output

LSTM

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

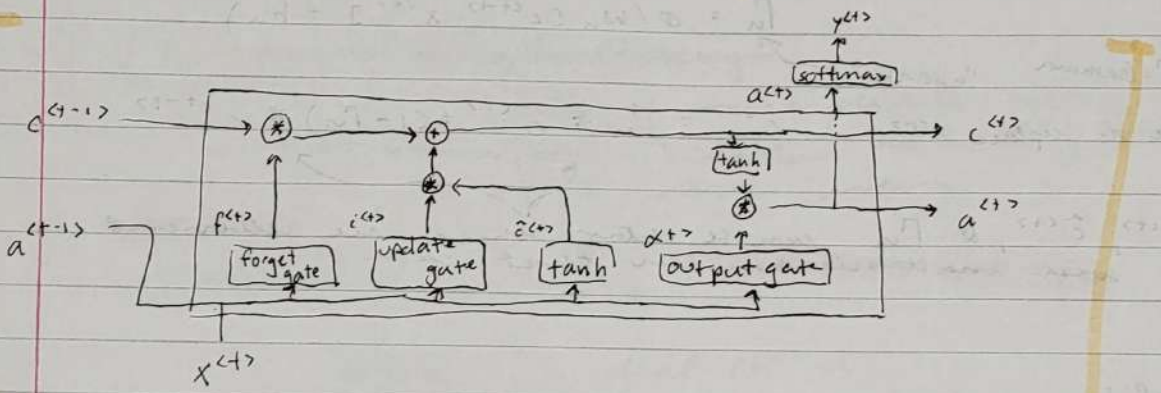
$\Gamma_u = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$

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

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

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

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



"peephole connection"

where in $\Gamma_u, \Gamma_f, \Gamma_o$, $W[a^{\langle t-1 \rangle}, x^{\langle t \rangle}, c^{\langle t-1 \rangle}]$,

$c^{\langle t-1 \rangle}$ used to affect gate value

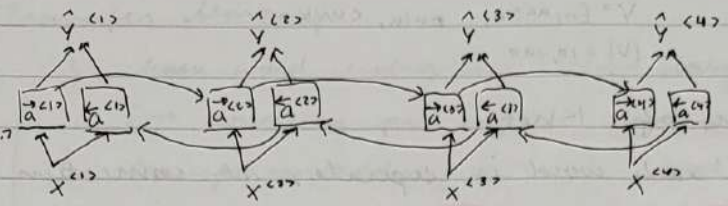
GRU v. LSTM

neither is unanimously superior

- GRU is lighter, easier to scale
- LSTM more powerful, flexible

Week One 11/12 Bidirectional RNN (BRNN) T=106 3/24/25

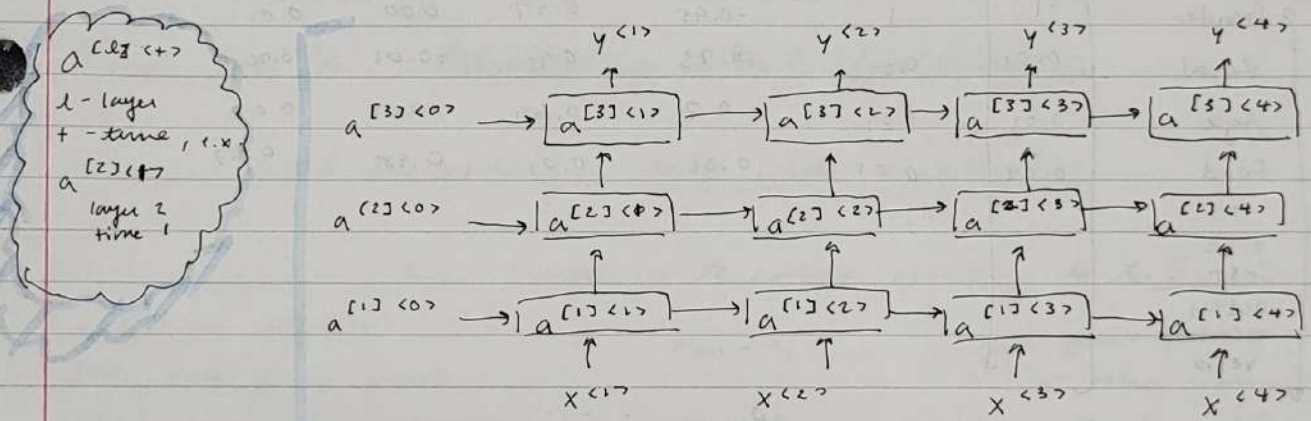
$$\hat{y}^{<t>} = g(W_y[\vec{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$$



RNN blocks
or
GRUs
LSTM blocks ← commonly used in BRNNs

note: need entire sequence before BRNN can use it, ex. not suited for rolling speech transcription

Week One 12/12 Deep RNNs



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

- 3 layers is kinda already deep for RNN
- computationally expensive
- could connect y's to standard neural network layers (not horizontally connected)

Intro to Word Embeddings

Week Two 1/4 Word Representation

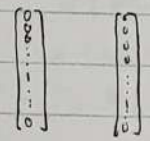
T = 121

1-hot representation

$V = [a, aaron, \ldots, zulu, \langle UNK \rangle]$

$|V| = 10{,}000$

Man (5391) Woman (9853)



O_{5391} O_{9853}

Cons of 1-hot

- each word is seperate, no connection between them

Featurized representation

features	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						
action						
noun						
verb						
⋮						

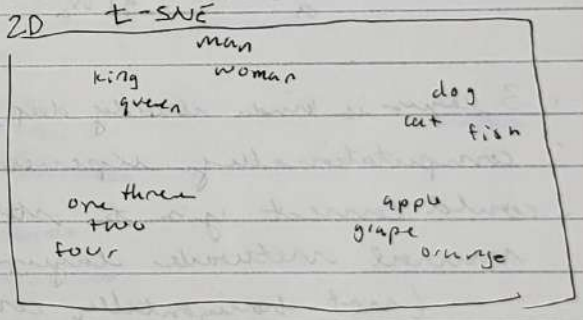
300

e_{5391} e_{9853} e_{4914} . . .

t-SNE - visualizing word embeddings

3D

300D → 2D



300D orange vector

Week Two 2/4 Using Word Embeddings

T=130

Transfer learning

(A) 1. learn word embeddings from large text corpus (1-100B words) (or download pre-trained embedding)

(B) 2. transfer embedding to new task w/ smaller training set (e.x. 100k)

3. optional: fine-tune embeddings w/ new data (only in practice when training set is larger)

most helpful when tons of data for (A) and less for (B)

T=141

Week Two 3/4 Properties of Word Embeddings

word embeddings help with analogies

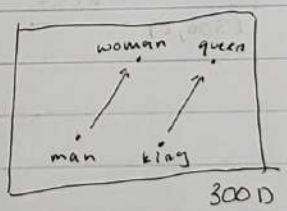
e.x. "man is to woman as king is to ?"

model can calculate → then look for a word that creates same result for $e_{king} - [\ ?\ ]$

$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ \vdots \\ 0 \end{bmatrix}$ ← gender feature; ← other features

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

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



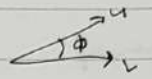
find word w that maximizes similarity of $e_{king} - e_w$ & $e_{man} - e_{woman}$

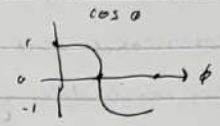
$\arg\max \text{sim}(e_w, e_{king} - e_{man} + e_{woman})$

visual / parallelogram relationships likely won't hold true t-SNE

Cosine similarity

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





or

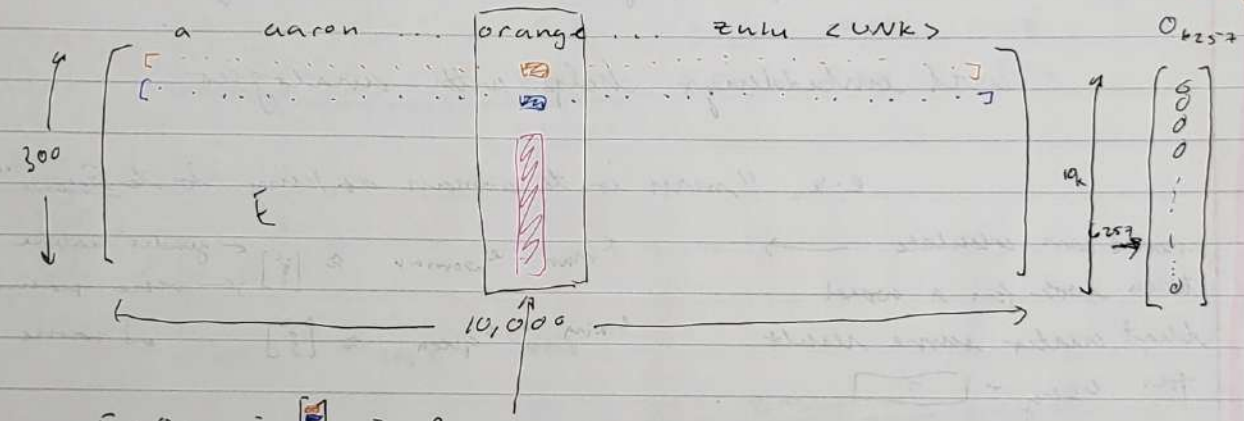
Euclidean distance / square distance

$$\|u - v\|^2$$

Week Two 4/4 Embedding Matrix

T = 145
~ 2.4 h

When you implement an algo to learn word embeddings, what you learn is an embedding matrix



$$E \cdot O_{6257} = e_{6257}$$

(300, 10k) (10k, 1) (300, 1)