

DEEP NEURAL NLP

27/7/23

① NLU vs NLP.

② Compositional Semantics

$$\text{NL} \xrightarrow{T(\circ)} \text{FL}$$

\uparrow \downarrow
 T^{-1}

$T(\circ)$ → transformation to convert Natural lang(NL) into Formal lang (FL)

T^{-1} → reverse (FL to NL).

Deep Neural NLP

③ Distributional Semantics (IMP).

$$\text{NL}_1 \xrightarrow{T(\circ)} \text{FL}_1$$

Agenda -

If $\text{NL}_1 = \text{NL}_2$ then $\text{FL}_1 = \text{FL}_2$

$$\text{NL}_2 \xrightarrow{T(\circ)} \text{FL}_2$$

$f\text{sim}(\text{NL}_1, \text{NL}_2) = y$,

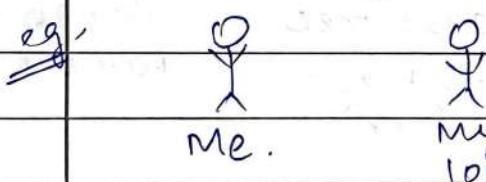
then,

$$f\text{sim}(\text{FL}_1, \text{FL}_2) = y$$

$$y \in [0, 1]$$

exacty
similar

→ Notion of similarity → highly subjective.



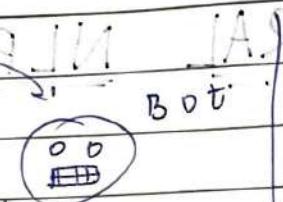
If I have pictures of both, it should be exactly similar (even the mistakes).

→ Syntax → grammar.

Semantics → "literal" meaning.

Pragmatics → intended meaning

— / —

Sentence 

A. Polysemy - no relation b/w words, but same word.

eg. Jaguar
(car & ~~car~~^{animal})

can the root give:-
 i) grammar
 ii) meaning
 iii) intended meaning.

Polysemy - same connection, but diff.

eg. bank.

(wood bank)

→ context window → how much window size to consider.
 + c

Form: S V O
 ↓ ↓ ↓
 Subject Verb Object

28/7/23

Syntax

↓	↓	↓	↓
Form (SVO)	Part of speech (POS) - "sequence"	Parse tree (context-free)	Dependency parsing.

→ If something is grammatically correct, doesn't necessarily mean, it is meaningful.

John loves Icecream

NP (Noun phrase) VP (Verb phrase)
 ↓ ↓
 NN (John) loves NN icecream

VBZ - third person singular form of verb

— / —

lexicon - has entry in chosen dictionary
↳ vocabulary.

- Parse tree.
- Chomsky Normalized Form (CNF)

11/8/23

try displacy. → to visualize dependency parse.
(by spacy)

⑥ NLP pre-processing :-

① Stopword removal (depends on what should really be removed.
Sometimes we may want to keep some).

② Punctuation removal (including hyphens)

③ de-abbreviation

④ Numerical normalization

⑤ Lemmatization/ stemming

⇒ Named Entity Recognition (NER)

⇒ Vector Space Modeling → one method of NL → PL

(distributional semantics)

claim → Meaning = context
(ugly)

LL

3|8|23

↳ Similarity → sharing common properties

$w_{i-2}^o \rightarrow w_{i-1}^o \rightarrow w_i^o \xrightarrow{\text{rent}} w_{i+1}^o \rightarrow w_{i+2}^o$

$w_{j-2}^o \rightarrow w_{j-1}^o \rightarrow w_j^o \xrightarrow{\text{house}} w_{j+1}^o \rightarrow w_{j+2}^o$

might happen
that
these
words
are equal

Distributional semantics → Linguistic semantics

Three condⁿ for similarity :-

(i) words occur in same loc.

(ii) same surrounding words

(iii) freq. of surrounding words is same.

→ Drawback:- model can get confused.
(eg. home & house).

→ also house & event are related, but not similar.

(similarity means computing relatedness)

④ Vector Space Model

↳ Represent a "linguistic unit" as a vector.

character,

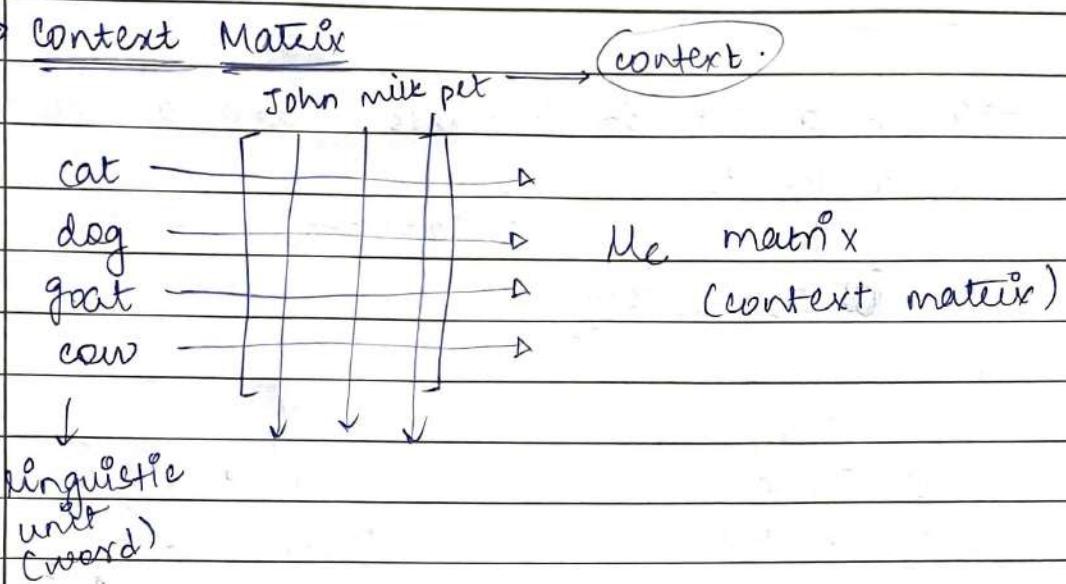
word,

clause,

sentence,

sheet page, long para,
document

⇒ Context Matrix



- always look at matrix in terms of column vectors
- describe columns (features) using neural
- features ⇒ basis (used to construct a particular data point)

John-ness → unique property of John

⇒ not equal to John (vector)

eg:

$$\begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix} = 2 \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 1 \times \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

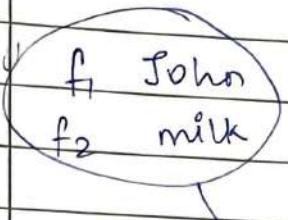
$$= \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad (\text{linear transformation})$$

Topic: Why Context Matrix is Important?

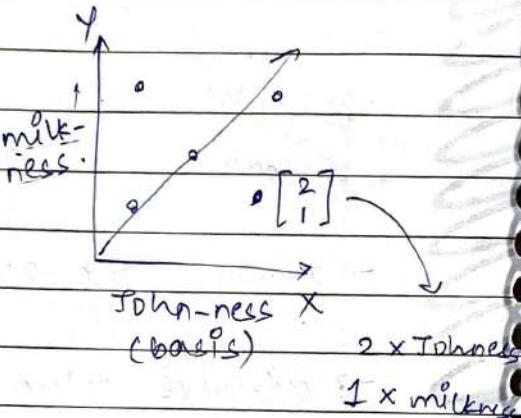
4/8/23

11

- Given a set of features and a set of data points.
- \downarrow $\{f_1, f_2, \dots, f_n\}$
~~feature vector~~
 $\{w_1, w_2, \dots, w_n\}$



denote the words associated with each of the data points.



- Catness & Dogness not same as Cat & Dog

	John	milk
cat	2	1
dog	3	4
cow		

↑
cowness

starting point of dataset

$$\begin{array}{l}
 \text{John milk} \\
 \text{cat } \begin{bmatrix} 2 & 1 \end{bmatrix} \times \begin{bmatrix} 2 \\ 1 \end{bmatrix} \times \text{Johnness} \\
 \text{dog } \begin{bmatrix} 3 & 4 \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \end{bmatrix} \times \text{milkness.} \\
 \text{cow } \begin{bmatrix} 1 & 4 \end{bmatrix} \times \begin{bmatrix} 3 \\ 3 \end{bmatrix} \times \text{catness.}
 \end{array}$$

$$= 2 * \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} \times \text{centress} + 1 * \begin{bmatrix} 1 \\ 0 \\ 6 \end{bmatrix} \times \text{dogness} = \begin{bmatrix} 5 \\ 6 \\ 6 \end{bmatrix} \times \text{catness}$$

cat

? : $v_1 \rightarrow v_2$

1/1

$\begin{bmatrix} 5 \\ 6 \\ 6 \end{bmatrix}$ ← representation of cat.

(moving cat from one location
to another)



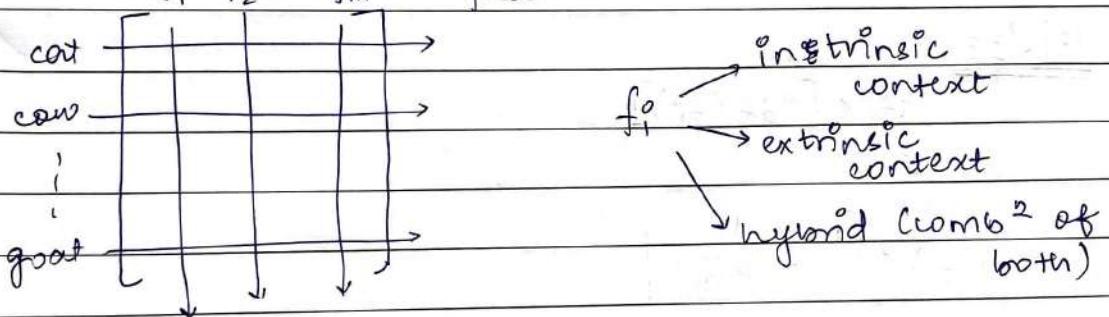
why change location?

→ Matrix multiplication = linear transformation
(bcz it's a modification of same entity)

8/8/23

Context Matrix

Lang exptl give context
 $f_1 f_2 \dots f_m$ features \Rightarrow basis.



e.g. cat

Intrinsic $\rightarrow f_1 f_2 f_3$
"c" "a" "t"

e.g. for phrase,
intrinsic context
can be words,
characters.

extrinsic $\rightarrow f_1 f_2 f_3$
John love milk

for sentence, \rightarrow words,
phrases

⇒ Intrinsic style.

Intrinsic
content
engineering
(like feature
engineering)

How to represent
linguistic units

- Eg. while querying a document during cosine similarity, context is intrinsic.
 → creating feature vector using TF-IDF.
 → words inside document → intrinsic
 → Context Engineering. (?)
 → Are query & document similar or semantically similar?
 (or related)

17/8/23

$$tf * [idf(f_i)]$$

$$P(f_i) * \left[\log \frac{1}{P(f_i)} \right]$$

↑
how "unique" (rare)
this feature is

is
an approximation
of $P(f_i|w_j)$

if f_i : tail
 w_j : cat

$$P(f_i|w_j)$$

$$= \frac{c(f_i, w_j)}{c(w_j)}$$

linguistic
unit: words

context unit: words
(extrinsic)

c → count

Topic^o (TF-IDF revisited)

- / /

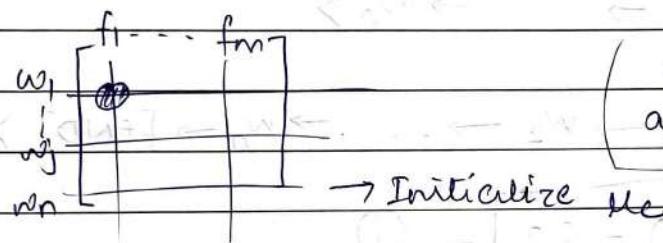
$$\log \frac{1}{P(f_i^o)}$$



$$P(f_i^o) * IC(f_i^o) = E[IC(f_i^o)]$$

↑
(expectation)

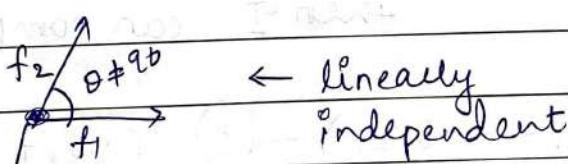
→ We try to understand the average info. content



(data point can be
any linguistic model)

We try to initialize context matrix
using tf-idf.

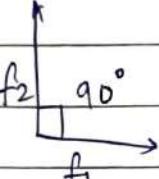
→ all features are independent of each other
(orthogonal). J (understanding orthogonality)



but we don't want shadow also
(i.e. projection)

→ Inductive bias. (when we create context
matrix)
into model
(good)

* LANGUAGE MODELS

 → assumption (ugly)
→ BOW (Bag of words assumption)

⇒ What is LM (Language Model)?

sequence.

$w_1 \rightarrow w_2 \rightarrow \dots \rightarrow w_n$

$\langle [\text{START}] \rightarrow w_1 \rightarrow w_2 \rightarrow \dots \rightarrow w_n \rightarrow [\text{END}] \rangle$

(prob.) $\hat{P}(w_1 \rightarrow w_n) = ??$ —①

↳ This is what any LM does

$\hat{P}(w_n | w_1 \rightarrow w_{n-1})$ —② (Next word prediction)

If I can compute ①,
then I can compute ②, ③

$\hat{P}(w_i \rightarrow w_n | w_1 \rightarrow w_{i-1})$ —③ (Next sequence prediction)

IMP * $P(w_1 \rightarrow w_2 \rightarrow w_3)$ (here seq. matters)

$\neq P(w_1 w_2 w_3)$ (free order not reqd.)
(just they must appear together) } Joint prob.

Chain rule to joint prob. $\rightarrow P(w_1, w_2, w_3)$

$$= P(w_1) * P(w_2 | w_1) * P(w_3 | w_1, w_2)$$

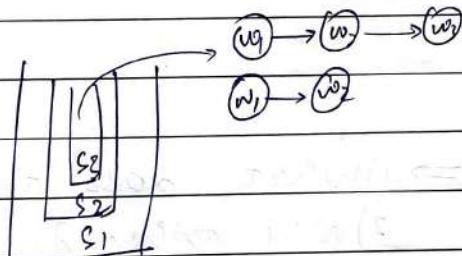
or

$$P(w_3) * P(w_2 | w_3) * P(w_1 | w_3, w_2)$$

For sequence $\rightarrow P(w_1 \rightarrow w_2 \rightarrow w_3)$

$$= P(w_2 | \leftarrow w_1) * P(w_3 | \leftarrow w_2, \leftarrow w_1)$$

(slight catch to the chain rule)



$$P(w_3 | \leftarrow w_2, \leftarrow w_1)$$

↳ gram modeling

$$P(w_3 | \leftarrow w_2)$$

↳ bi-gram modeling

$$P(w_3)$$

↳ unigram modeling

(bag of words)

(Assumes that
no influences
of words on
each other)

from
prob.
point of
view)

Topic:- N-gram modeling & Skip-gram Modeling

18|8|28

N-gram modeling

(as a particular case of language Model)

$$\hat{P}(w_m | \leftarrow w_{n-1} \leftarrow \dots \leftarrow w_1 \leftarrow \text{START})$$

too long

↳ computationally expensive.

↳ Markov Assumption \Rightarrow

$$\text{approximate } \approx \hat{P}(w_n | \leftarrow w_{n-1}) \quad (\text{bigram model})$$

degree = 1 (past only)

$$\hat{P}(w_1 | \text{START})$$

$$\hat{P}(\text{END} | w_n) \Rightarrow 1) \text{What does this mean?}$$

2) Why bother?

↓
what's
prob. of

w_n
being
last one.

$$P(A \rightarrow B \rightarrow C) = P(A) * P(B | \leftarrow A) * \underbrace{P(C | \leftarrow B \leftarrow A)}_{\text{approx}}$$

↳

$$P(C | \leftarrow B)$$

\Rightarrow Problem with considering short sequences:-

e.g. John who lives in NY and works at Apple loves ice cream,

For eg. If I consider only prev 2 words, then Icecream to be predicted only based on loves & Apple. (doesn't make sense).

∴ choose n properly in n -gram modeling. But if n is very large, no point of Markov assumption.

$$\Rightarrow \text{Maximum likelihood} \rightarrow \hat{P}(\text{Icecream} | \text{loves}) = \frac{L(\text{IC}, \text{loves})}{L(\text{loves})}$$

$$\hat{P}(\text{IC} | \leftarrow \text{loves}) = \frac{L(\text{loves} \rightarrow \text{IC})}{L(\text{loves})}$$

(occur together
in such
a way that
'loves'
comes
first)

↑
classical
NLP way of
computing a sequence.

→ Out of vocabulary situation

→ Bag-of-words model \Rightarrow when $N=0$ in n -gram modeling.

④ Language Models \Rightarrow Skip-gram Modeling

$$\hat{P}(w_1 \rightarrow w_2 \rightarrow \dots \rightarrow w_i \rightarrow \dots \rightarrow w_n)$$

its influenced

by both past & future context

Strong influence \rightarrow suppress / boost

22 | 8 | 23

- / -

④ Skipgram Model (Word2Vec)

$$P(w^c | w^t)$$

w^t : target word

(center word)

w^c : context word

n-gram → focuses on moving forward

+m

||

j=m



context window = 2m.

why
target
word
is
center?

(Bcoz it
will always
be at
center in
context
window)

$$\{w_{t-n}^t, \dots, w_t^t, \dots, w_{im}^c\}$$

↓
center
word

→ Conditional Independence vs Total Independence

$$P(\text{John} \rightarrow \text{love} \rightarrow \text{IC})$$

$$= P(\text{John}) * P(\text{love} | \leftarrow \text{John}) * P(\text{IC} | \leftarrow \text{love} \leftarrow \text{John})$$

we are approximating this based on all mutually
diff techniques (n-gram/ skipgram) independent

= unigram
model

$\prod_{w_t=1}^T$

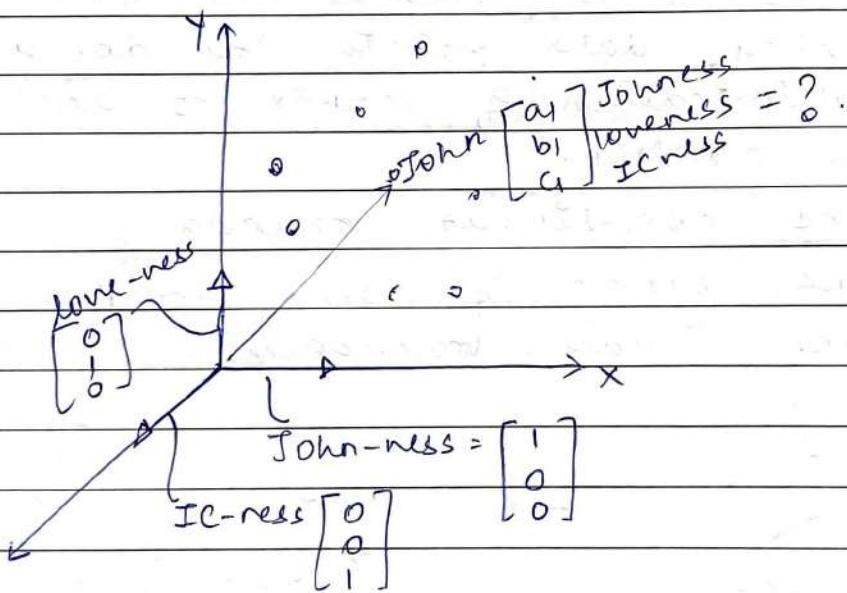
T: all the unique target words

$$\prod_{w_t=1}^T \left[\prod_{j=-m}^{+m} P(w_{t+j}^c | w_t) \right]$$

partition
index
effect

$w_i \rightarrow 2$ avatars

⑩ Word2Vec



$[w_{\text{love}}^t]$

love

$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

word2vec
(skipgram)

goal: takes target

word as o/P ,

gathers context

words & returns
them as o/P

$$\hat{P}(w_{t+j}^c | w_t, w_{t+1}, \dots, w_T) = \text{John}$$

$$- I.C. \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

24
25/8/23

content matrix (M_c)

vocab
size
 $w_1 = w_n$

features
 $f_1 f_2 \dots f_n$

data points	w_1	w_2	\dots	w_n
:	0	0	\dots	0
	1	1	\dots	1
	.	.	\dots	.

for initialization of

context matrix, we start from taking unique features of data points. i.e. -ness features of data.

(we have to look at matrix in feature perspective)

→ Estimate.

$$\hat{p}(\text{Start}_0^c \rightarrow w_1^c \rightarrow w_2^c \rightarrow \dots \rightarrow w_t^c \rightarrow w_n^c)$$

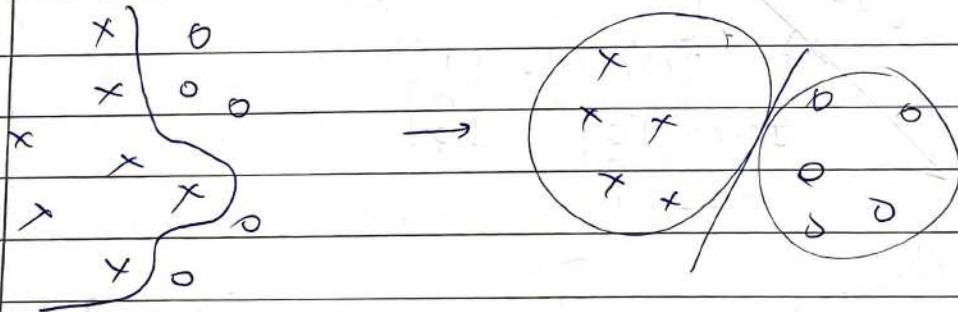
$$= \hat{p}(\text{Start}_0^c) \times p(w_1 | \text{start}) \times p(w_2^c | w_1^c \leftarrow \text{start}) \times \dots \times p(w_n^c | w_{n-1}^c \leftarrow w_n^c \leftarrow \text{start})$$

→ FNN (Feedforward Neural Network)

- making data points by doing multiplication of matrix to arrange them as bunches.

- make non-linear boundary

- make bunch of same group & then make linear boundary).



→ relocation is done by applying weights to it.

$$\begin{array}{c}
 \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} f_1' \\ f_2' \\ \vdots \\ f_n' \end{bmatrix} \\
 \text{Latent features} \quad \text{Wines} \quad \text{finesse} \\
 \downarrow \quad \downarrow \quad \downarrow \\
 \text{Wines} \quad \text{finesse}
 \end{array}$$

Topic: Word2Vec

25/8/23.

- ① LM can be approximated using skip-gram modeling.
 $\hat{P}(\text{START} \rightarrow w_1 \rightarrow w_2 \cdots w_n \rightarrow \text{END}) = ?$
- ② Word2Vec follows skip-gram modeling.
- ③ Word2Vec does not attempt LM.
- ④ We can extend Word2Vec to do LM.

⑤ $\hat{P}(w_{t-m}^c \rightarrow w_{t-(m-1)}^c \rightarrow \cdots \rightarrow w_t^c \rightarrow w_{t+1}^c \rightarrow w_{t+2}^c \cdots \rightarrow w_{t+m}^c | w_t^+)$

(sequence) word2vec attempts to estimate this.

(just a join) $\approx \hat{P}(w_{t-m}^c, w_{t-(m-1)}^c, \dots, w_{t-1}^c, w_t^c, w_{t+1}^c, w_{t+2}^c, \dots, w_{t+m}^c | w_t^+)$

joint distribution of the context of words (w^c) given w_t^+ .

- ⑤ is the first assumption that Word2Vec has

words (w^c) given w_t^+
 $\approx \prod_{j=t-m}^{t+m} \hat{P}(w_j^c | w_t^+)$

w ₁ -ness	→ 0
w ₂ -ness	→ 0
:	
soveness	→ 1
0	
:	
w _M -ness	→ 0

w_{love}

↳ one-hot
(Input) vector.

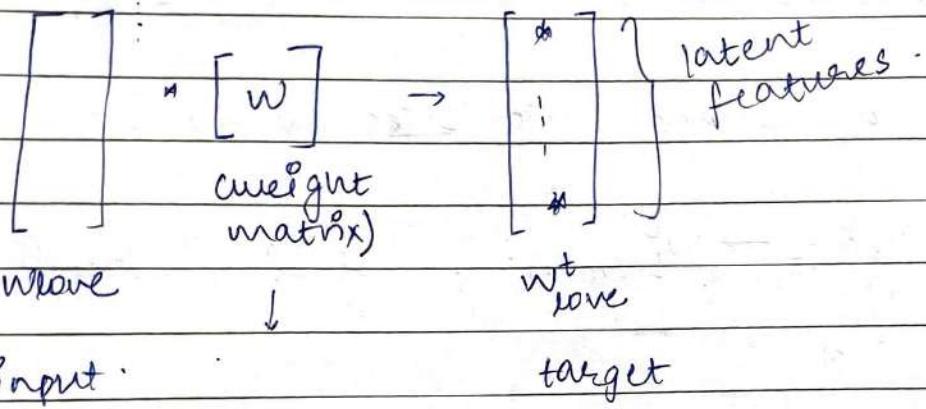
Row → data points.

col → features

w ₁ -ness	↓	- - -	0
w ₂	↓	↓	↓
:	↓	↓	↓
w _M	↓	↓	↓

content
matrix

(w₁ only contains
w₁-ness & so on)



(we don't
pass entire
context
matrix -
for word2vec)

→ self supervised (eg: relocate until friends
are found)

↓
side-effect

↖ search at the
right location

(info
contained

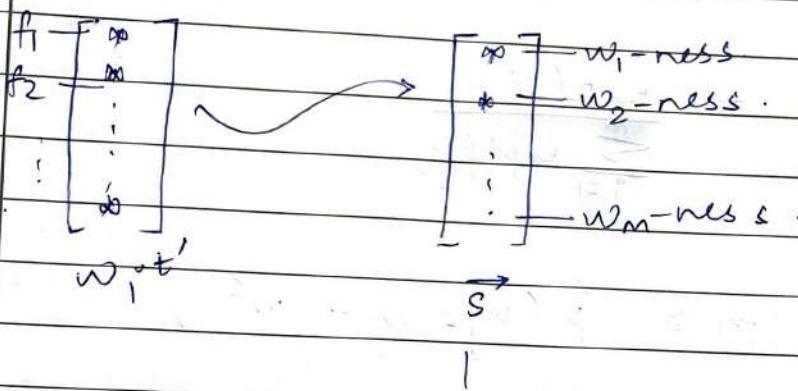
in training data)

28/8/23

11

Input: $[0 \ 0 \dots 1 \ \dots 0 \ \dots 0]$ w_i^t .
 Desired output: $\{w_j^c\}$.

$$\prod_{i=1}^n P(w_j^c | w_i^t)$$



has to be related to similarity
func².

$\hookrightarrow (6) \rightarrow$ Word2Vec: dot
 (use) product

$$w_i^t \cdot w_j^c = S_{ij}^{oo}$$
 (similarity using dot product)

$$\Rightarrow w_i^t \cdot w_j^c = S_{ij}^{oo}.$$

$$\Rightarrow w_i^t \cdot w_c^T = \vec{s}$$

$$w_c^T \cdot w_i^t = \vec{s}$$

$$w_c^T = f^T \begin{bmatrix} w_1 & \dots & w_m \end{bmatrix}$$

→ Each word is unique.

→ From the world of vector space to probability space.

$$p(w_j | w_i^t) \leftarrow \text{probability vector}$$

→ softmax. $\rightarrow \frac{\exp(f_i^o)}{\sum_{j=1}^m \exp(f_{ij}^o)}$

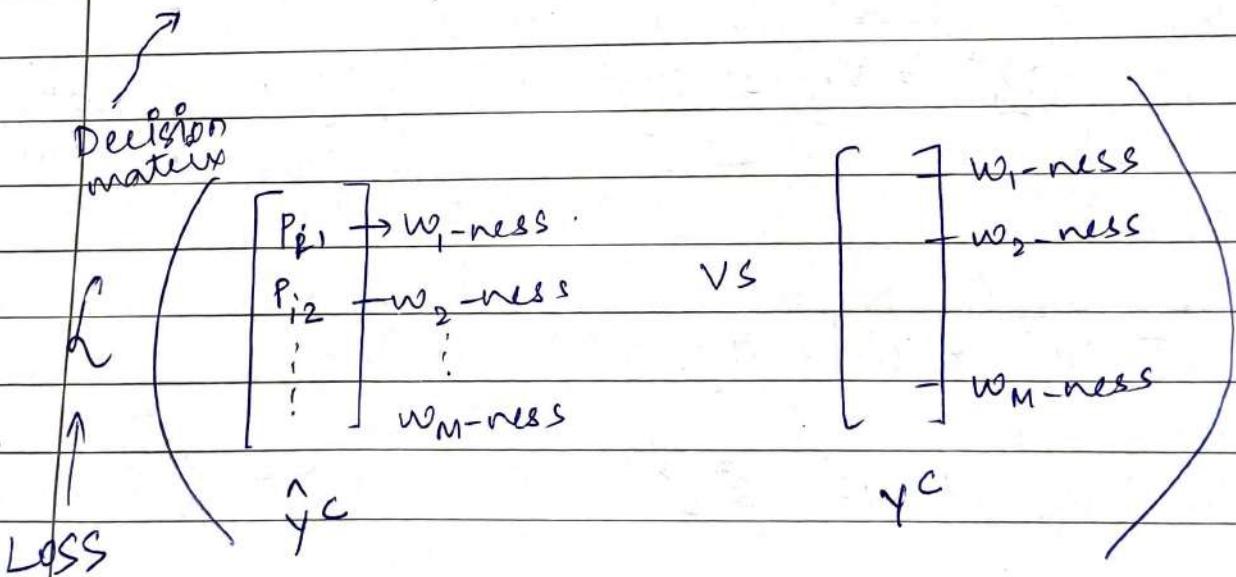
Estimated prob: $\hat{p}(w_j = w_c^c | w_i^t)$

29/8/23

$$w_i^t' = w^t * w_i^{o^t} = L(w_i^{o^t}) \xrightarrow{\text{hot vector}} \text{① (Linear transformation)}$$

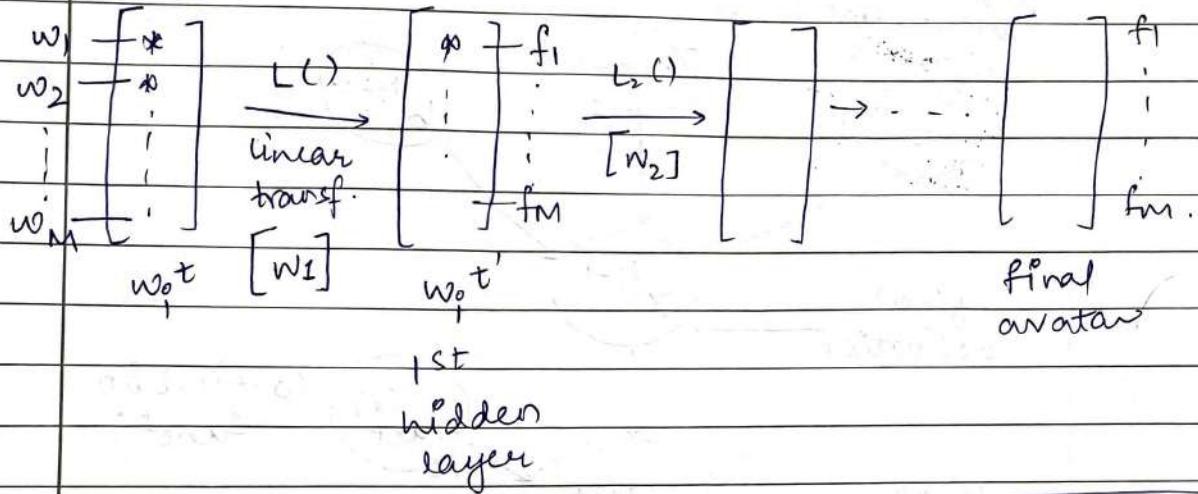
$$s_i^t = (w^c)^T * w_i^t' = f_d(w_i^t') \xrightarrow{\text{② (Decision func^2)}}$$

$$\hat{y}_c = \hat{p}(w_j^c | w_i^t) = \text{softmax}(s_i^t) \xrightarrow{\text{③}}$$

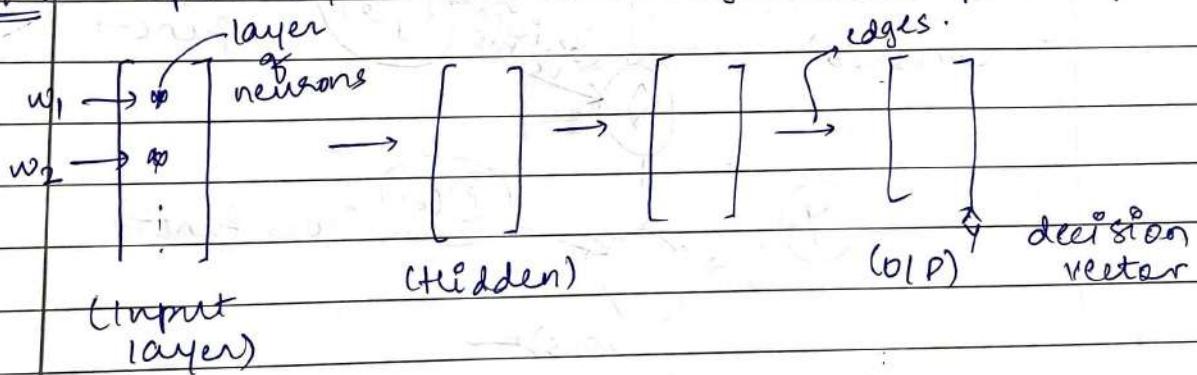


$$w^c \cdot w_i t' = (w^c \otimes w_i t')$$

FFA - Feed Forward Network

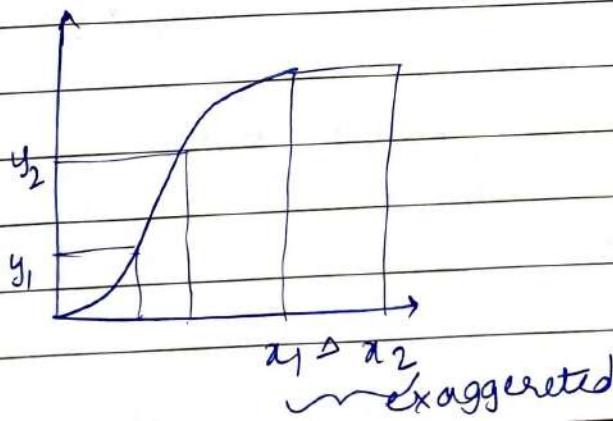


31/8/23 Input layer \rightarrow hidden layer \rightarrow output layer.



$$w_i t'' = w_0 \otimes w_i t' + \vec{b}_j$$

\Rightarrow logistic "activation" (squashing) function



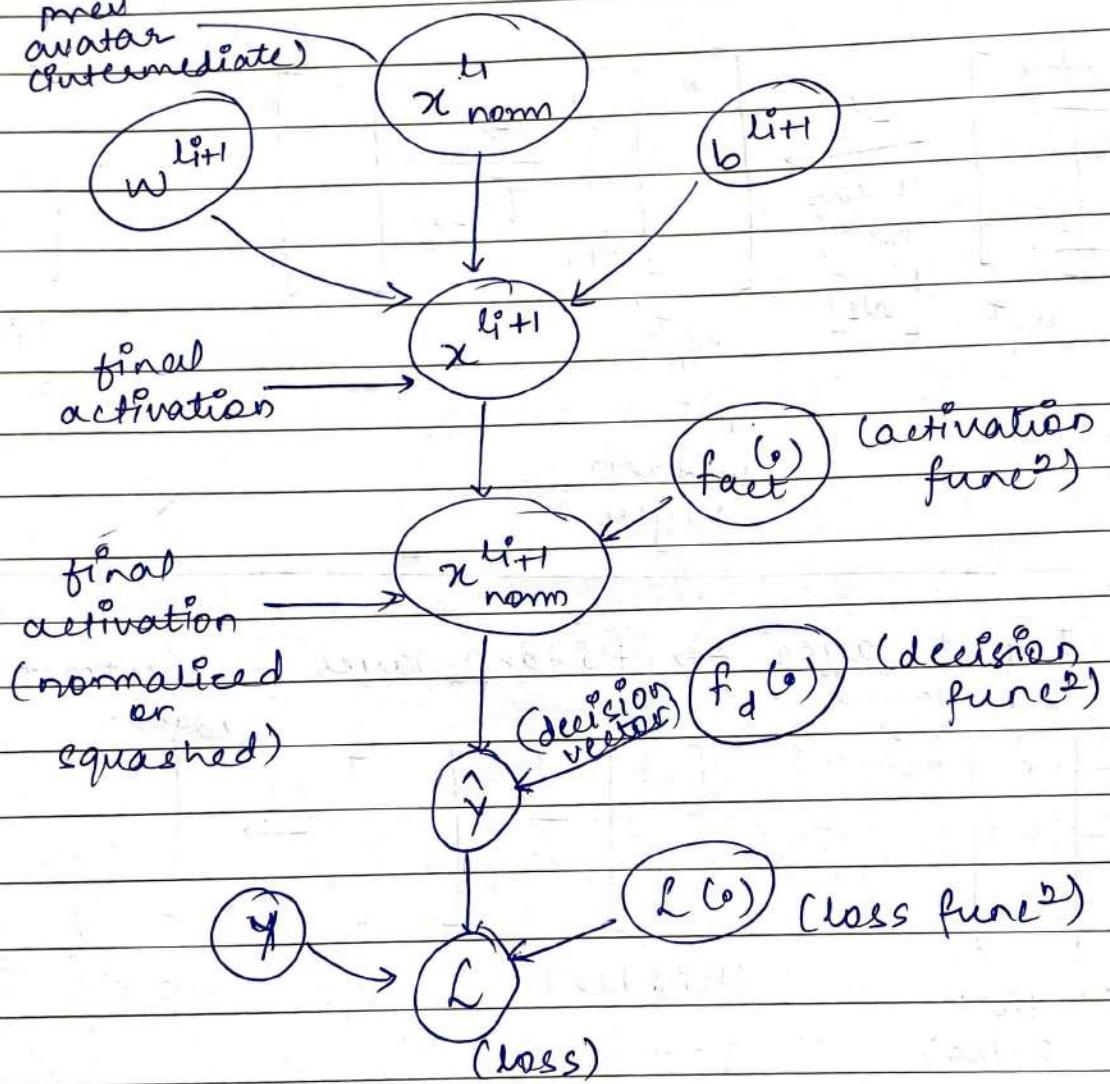
activation of neuron

1st neuron - 1st axis

$$w_i^M = (w_i^M * w_i^T + \vec{b}_i)$$

11/9/23

Topic :- Loss & Backpropagation



→ Blaming from bottom-to-top (who caused error)

→ Error propagate top-to-bottom.

⇒ Why partial differentiation?

$$\frac{\partial L}{\partial w^{l+1}} = ?$$

Chain rule.

$$\frac{\partial x^{l+1}}{\partial w^{l+1}} \Rightarrow \frac{\partial x^{l+1}_{norm}}{\partial x^{l+1}} \Rightarrow \frac{\partial y}{\partial x^{l+1}_{norm}} \Rightarrow \frac{\partial L}{\partial y} = \frac{\partial L}{\partial w^{l+1}}$$

Updation rule:-

$$w^{l+1} = w^{l+1}_{old} - \alpha \frac{\partial L}{\partial w^{l+1}} \quad [\text{(stochastic) gradient descent}]$$

↓ learning rate (how much to update)

↓ ↓
high low
 (full trust
on partial diff.)

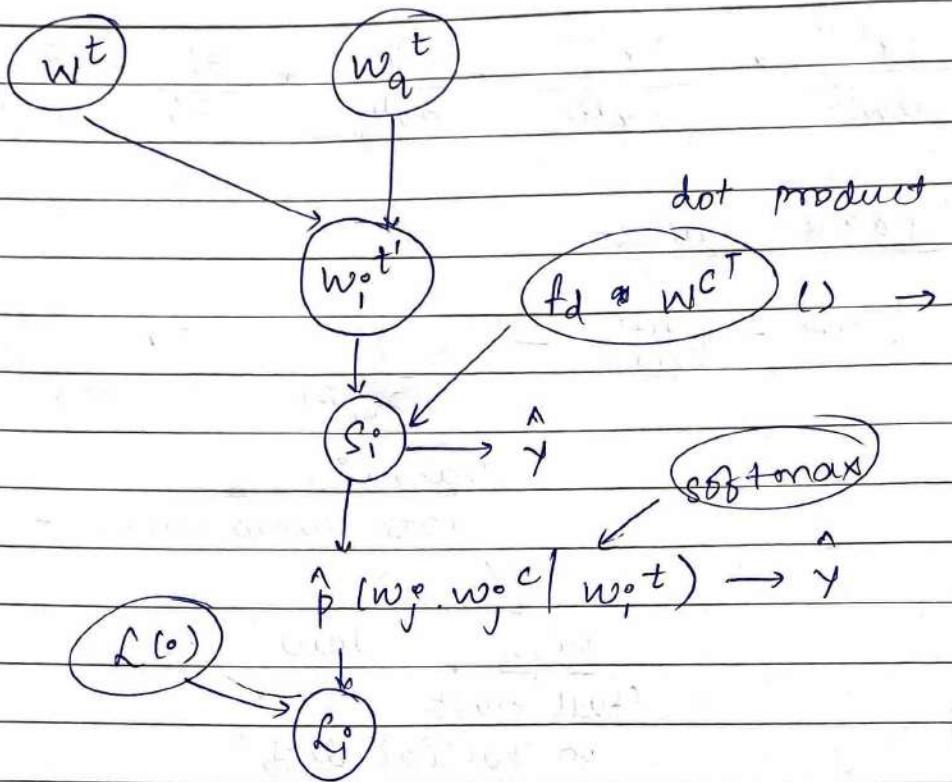
↓
 faster convergence
 if there is convergence
disadv.: overshoots

$$x^{l+1}_{norm} = x^{l+1}_{norm} - \alpha \frac{\partial L}{\partial x^{l+1}_{norm}}$$

$$b^{l+1} = b^{l+1}_{old} - \alpha \frac{\partial L}{\partial b^{l+1}}$$

What's missing in word2vec-

- No squashing
- No bias
- No activation func²



loss func² → cross-entropy

(minimize the expected "surprise" of $P(y|w_i^t)$)

info content

TF-IDF - high info content

2/9/23

Topic:- NN Update rule & Word2Vec.

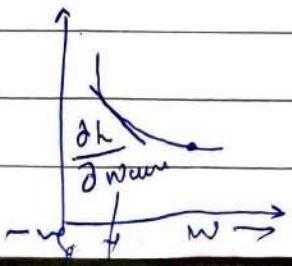
④ Update rule :- on basis of G.D., S.G.D.

$$w_{\text{new}} \leftarrow w_{\text{curr}} - \alpha \frac{\partial L}{\partial w_{\text{curr}}}$$

learning rate

Why subtract?
↓ convex func²

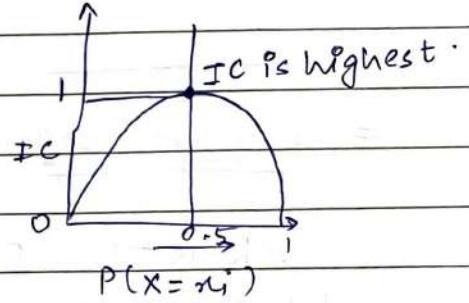
Cross entropy
-ve log likelihood



Chain rule. (Bcoz h & w don't have direct relationship)

Surprise = Information content (IC)

$$IC(x=x_i) = \log \frac{1}{P(x=x_i)}$$



$$\therefore IC(x) = \sum_{x=x_i} \log \frac{1}{P(x=x_i)}$$

(x takes discrete values (RN))

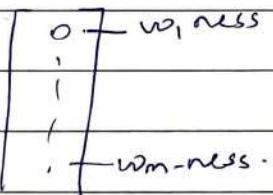
$$\begin{aligned} \min IC(x) &= \min \sum_{x=x_i} \log \frac{1}{P(x=x_i)} \\ &= - \min \sum_{x_i} \log P(x=x_i) \end{aligned}$$

e.g. If it is a classification problem,
 x_i → classes.

In word2vec, $x_i \rightarrow$

$$\hat{y} \Rightarrow P(w_i^o | w_i^t)$$

↑
Input



$$\sum P(w_i^o | w_i^t) = 1 \quad (\text{bcuz of softmax})$$

Minimize the surprise given the context

Cross entropy (no ignoring)

Entropy → expected surprise.

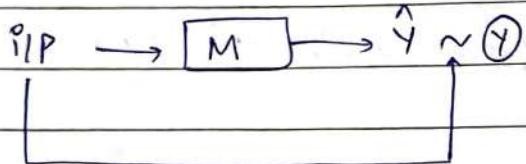
$$H(y, \hat{y}) = \sum_i \left[y_i \log \frac{1}{P(y_i)} + (1-y_i) \log \frac{1}{P(1-\hat{y}_i)} \right]$$

what I want what I don't want

14/9/23

Topic:- n-gram LM using RNN.

→ self-supervised setup.



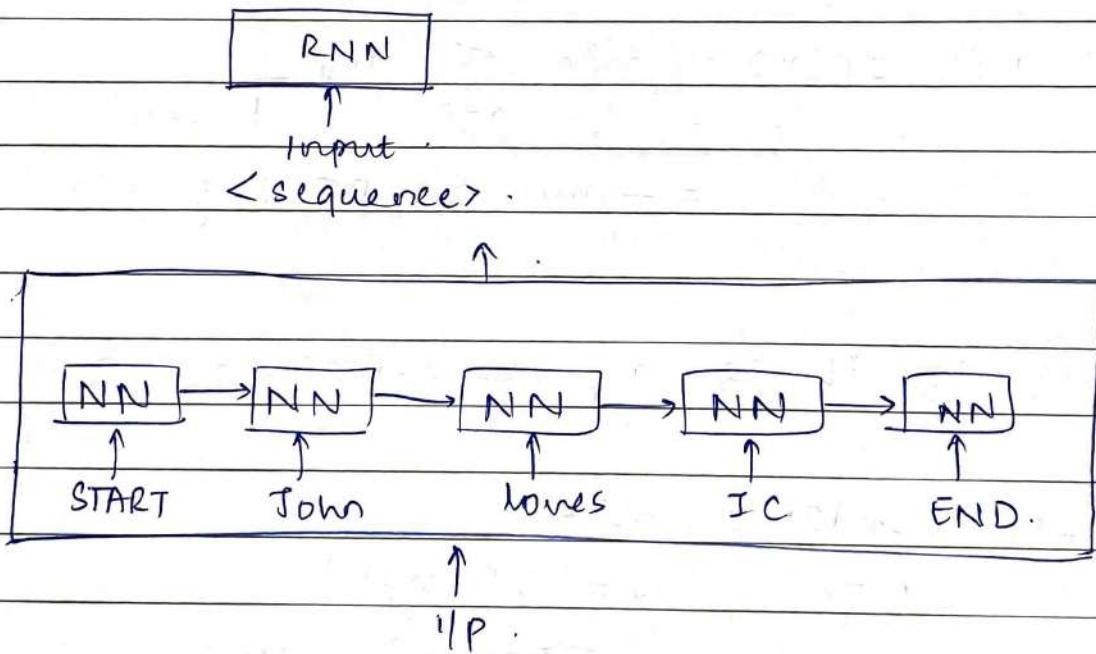
⇒ n-gram LM using RNN :-

RNN → 1990.

Word2Vec → 2013

(RNN)

recurrent.



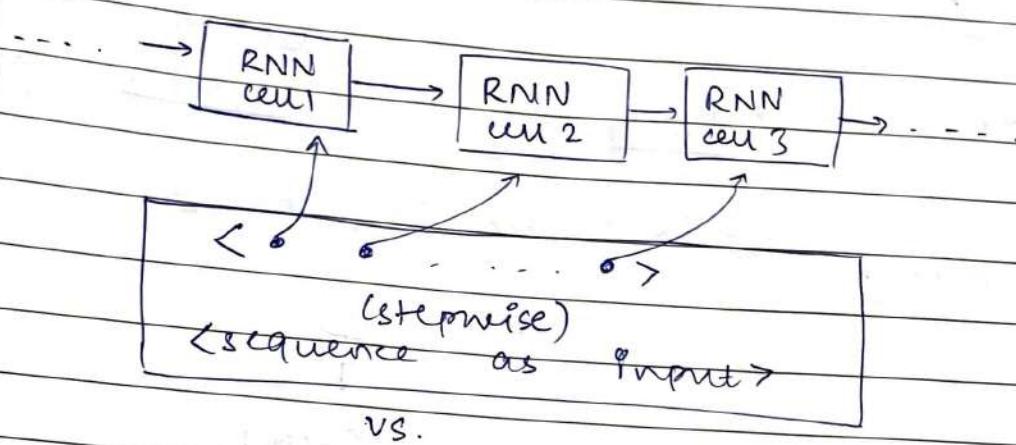
follow i.i.d. assumption

(Independent & Identically distributed)

15/9/22

RNN

— / —



<seq as an input> → word2Vec (CBOW)
vs.

<parts of seq as ip> → word2vec
(skipgram)

Conditional
independence.

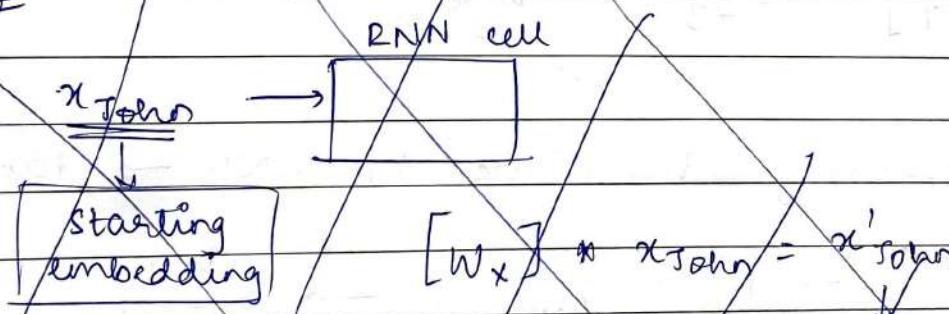
skipgram :- $\prod_j P(w_{j \in c} | w^t)$

c: context set
of w^t .

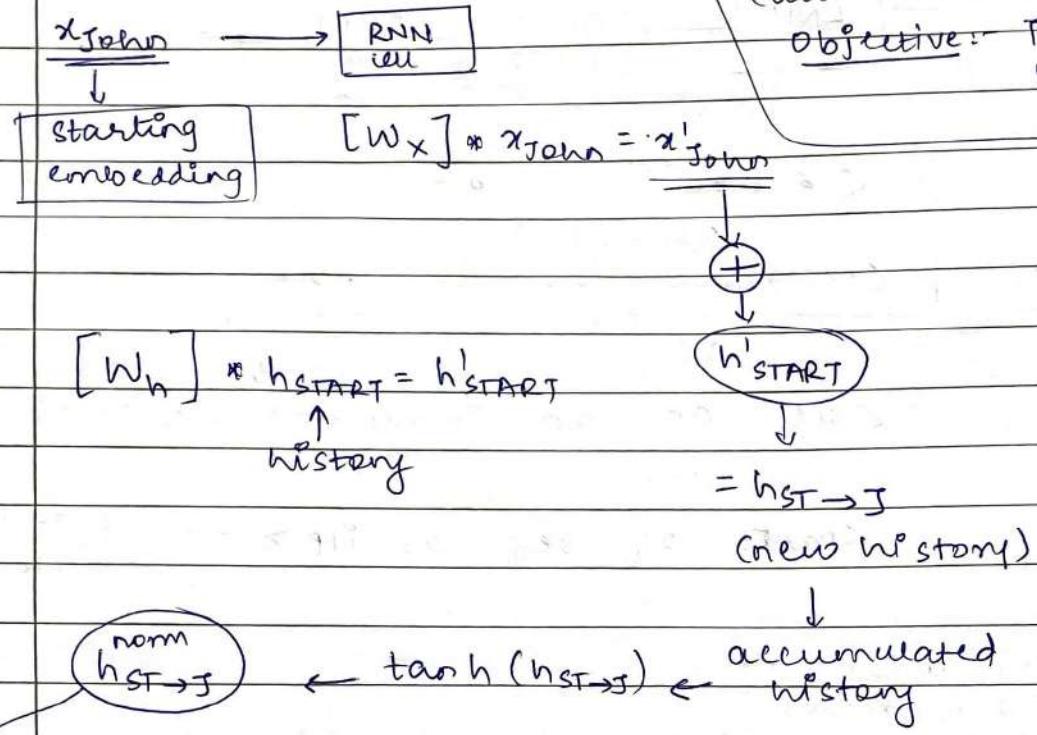
CBOW :- $\prod_j P(w^t | w_{j \in c})$

[what are
key takeaways
of word2vec?]

e.g. John loves icecream



eg. John loves icecream.



decision func²

$$f_d(h^{\text{START}} \rightarrow j) \mapsto \hat{y} \rightarrow \text{hopefully is 'love'}$$

$$[w_y] * h^{\text{START}} \rightarrow j = \hat{y}_{\text{love}}$$

→ Squashing technique in RNN $\rightarrow \tan h$

$$h^{\text{START}} \rightarrow j \rightarrow \tan h(h^{\text{START}} \rightarrow j) \rightarrow \text{norm } h^{\text{START}} \rightarrow j \\ (\text{squashed } h)$$

Ques:- we know \hat{y} but we don't know y

↓
go into world
of -nesses

$$\text{softmax } n \quad y_{\text{love}} \quad \hat{P} \cdot \quad \hat{P} = \frac{\text{softmax}(\hat{y})}{\text{argmax}(\hat{P})}$$

21/9/23.

RNN

$$x_t' = W_x \frac{x_t}{T} + b_x$$

\curvearrowright input

$$h_{t-1}' = (W_h h_{t-1} + b_h)$$

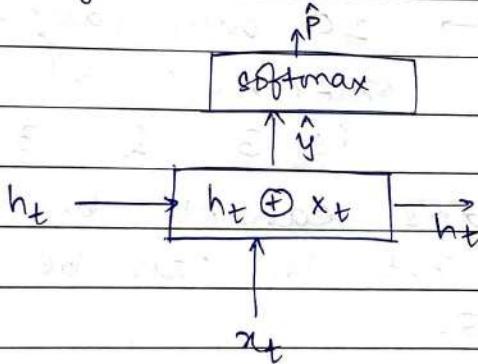
$$h_t = \tanh(h_{t-1}' + x_t')$$

$$\hat{y} = (W_y \frac{h_t}{T} + b_y)$$

\downarrow
decision
func

$$\hat{p} = \text{softmax}(\hat{y})$$

→ gives the next linguistic unit.



→ In word2vec we are going from word q's w's to f's (features)

vocab size |V| = 4

0.7	*	w_1 -ness → "sell"-ness
0.2	*	w_2 -ness → "even"-ness
0.05	*	w_3 -ness → "leave"-ness
0.05	*	w_4 -ness → "sleep"-ness

$p_y \rightarrow$

0	sell
0	even
1	leave
0	sleep

this should be expected.

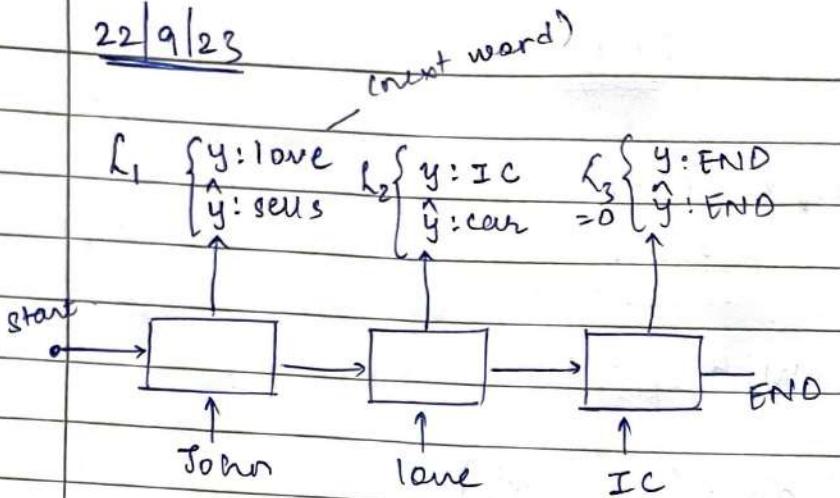
here we are pathetically wrong
(but love has cast)

loss $\rightarrow L_t(\hat{p}_y, p_y)$

hot vector

22/9/23

11



$$L = L_1 + L_2 + L_3$$

(8)

LSTM - elder brother of RNN.

→ In n-gram, sliding window of n

START → John → loves → IC → made → in → NY → END

eg 5-gram model → IC only takes

START → John → loves which
is 3 & $3 \leq 5$.

∴ 5-gram means cannot be more than 5, but it can be less or equal to 5.

RNN follows n-gram modeling in some sense

→ RNN & Word2Vec have different tasks.

↓
('next in sequence'
prediction) ↓
 context pred
 or
 pred next
 word from context)

LSTM \rightarrow long short Term Memory (gated network)
1995-97

- ① do we really need all of the cumulative history? (snowflake)?
 f : forget gate.

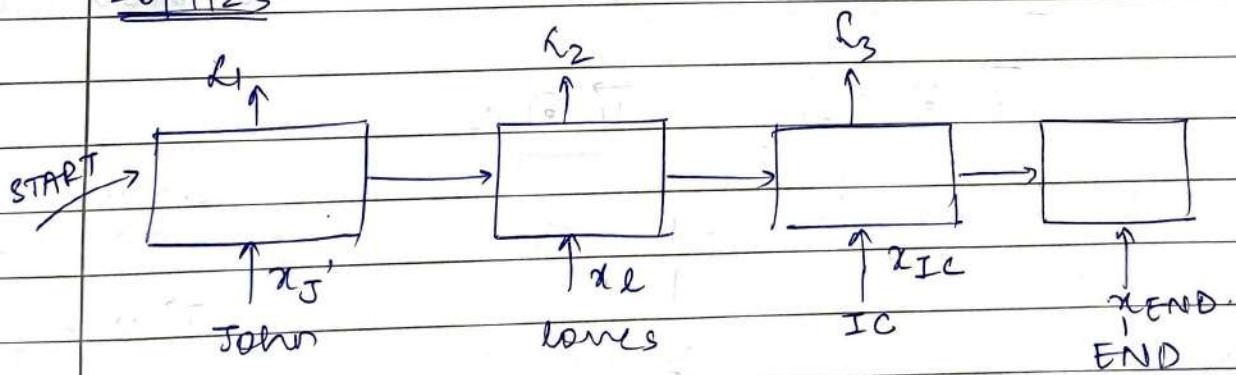
$f \oplus c'_{t-1}$
content (modified history)
how much of past child
to allow in this cell
(c_t)

- ② how much of the new info should be added.
 u_i - (update gate)

$i \oplus c_t$

- ③ Output gate. output filter - redundant addition

26/9/23



$$h = h_1 + h_2 + h_3$$

① Incoming history needs modifications before it can be used.

↓
unwanted
noise

should
be removed

↓
(forget
gate)

② f (func²) → acts as a reduction agent (squashing func²)

→ Why elementwise?

⇒ How f is generated?

$$f = \sigma \begin{bmatrix} w_f^T \cdot [x_t] \\ h_{t-1} \end{bmatrix}$$

(needs to
be
learned)

$$f = \sigma \underbrace{\left[w_f^T * x_t \right]}_{\text{recurrent}} + \underbrace{w_f^T * h_{t-1}}_{\text{recurrent}} + b_f$$

$f \odot c'_{t-1}$ save context in prev cell

$$\overrightarrow{f} \odot c'_{t-1} \quad \overrightarrow{u} \odot c_t$$

+
↓
 c'_t

should be
removed

$$\begin{aligned} & (\overrightarrow{f} \odot c'_{t-2}) \\ & + \\ & (\overrightarrow{u} \odot c'_{t-1}) \end{aligned} = c'_t$$

Vanilla RNN:-

$$h_t = \tanh h (w_h * h_{t-1} * b_h) + (w_x * x_t + b_x)$$

\hookrightarrow (t ("raw" new context or new history))

(→ When ' f ' becomes 0 vector, u becomes $\frac{1}{f}$ vector, then, LSTM becomes Vanilla RNN.)

$$\begin{bmatrix} u \end{bmatrix} = \sigma \begin{bmatrix} w_u * \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} \end{bmatrix}$$

$$(\text{needs to be learned}) = \sigma [w_u^* * x_t + w_u^* * h_{t-1} + b_u]$$

$$\begin{bmatrix} o \end{bmatrix} = \sigma \begin{bmatrix} w_o * \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} \end{bmatrix}$$

$$o = \sigma [w_o * x_t + w_o * h_{t-1} + b_o]$$

output filter