DEEP NEURAL NLP.

27/7/23

⊗ NLU v/s NLP.

① Compositional semantists

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

T^{-1}

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

T^{-1} → reverse (FL to NL).

Deep Neural NLP.

② Distributional semantists (IMP).

$NL_1 \xrightarrow{T(\circ)} FL_1$

$NL_2 \xrightarrow{T(\circ)} FL_2$

Agenda :-

If $NL_1 \equiv NL_2$ then $FL_1 \equiv FL_2$

$f_{sim}(NL_1, NL_2) = y$,

then,

$f_{sim}(FL_1, FL_2) = y$.

$y \in [0, 1]$.

exactly similar

→ Notion of similarity → highly subjective.

eg.

Me.

My look-alike.

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

→ Syntax → grammar.

Semantics → "literal" meaning.

Pragmatics → intended meaning

Sentence → Bot

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

Homonymy - no relation b/w words, but same word.
↳ eg. Jaguar. (car & animal)

Polysemy - same connection, but diff.
eg. bank.
(blood 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) VBZ (loves) NN (icecream)

VBZ - third person singular form of verb

__/__/____

lexicon - has entry in chosen dictionary
↳ vocabulary.

→ Parse tree.
→ Chomsky Normalized Form (CNF)

1/8/23

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

(*) 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

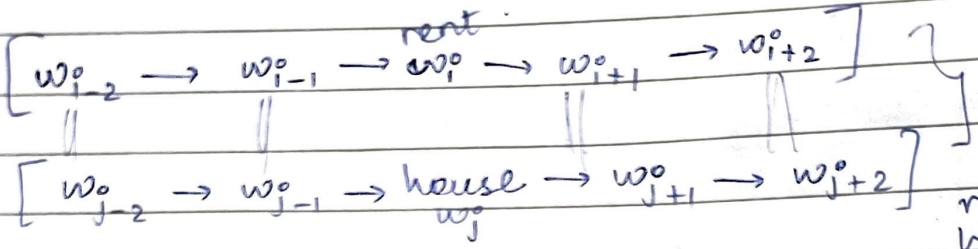
⇒ Named Entity Recognition (NER)

⇒ Vector Space Modeling → one method of NL → FL
(distributional semantics)

Claim → Meaning = context.
(ugly)

3/8/23

↳ similarity → sharing common properties



might happen that these words are equal

Distributional semantics → linguistic semantics
Three conds 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 & rent are related, but not similar.

(similarity means computing relatedness)

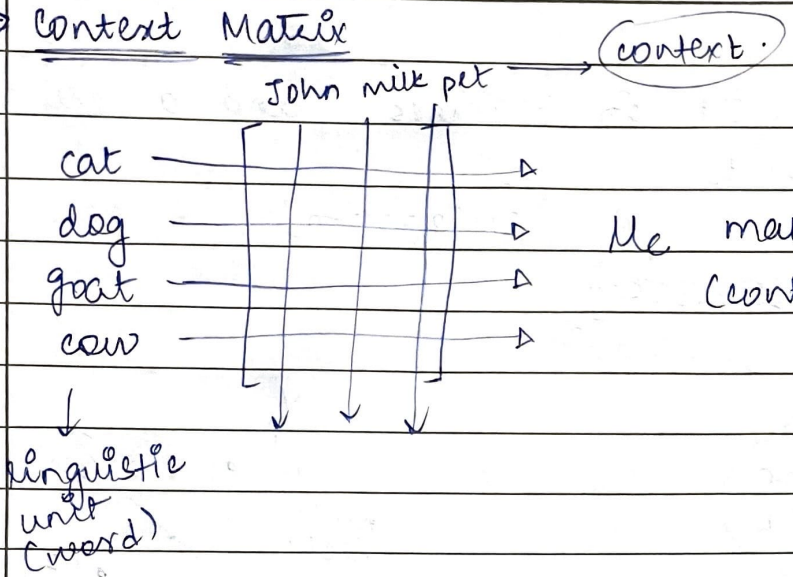
⑧ Vector Space Model

↳ Represent a "linguistic unit" as a vector.

↓
character,
word,
clause,
sentence,
short para, long para,
document

__/__/___

⇒ Context Matrix

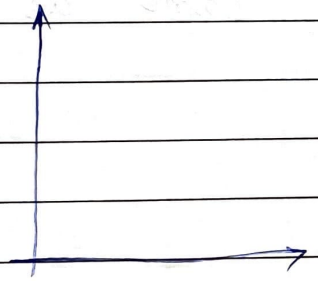


We matrix (context matrix)

→ always look at matrix in terms of column vectors

→ describe columns (features) using nouns

→ features ⇒ basis (used to construct a particular data point)



John-ness → unique property of John

↓

≠ not equal to John (vector)

eg:

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

$$= \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

(linear transformation)

Topic: Why context Matrix is important?

__/__/__

4/8/23

→ Given a set of features and a set of data points.

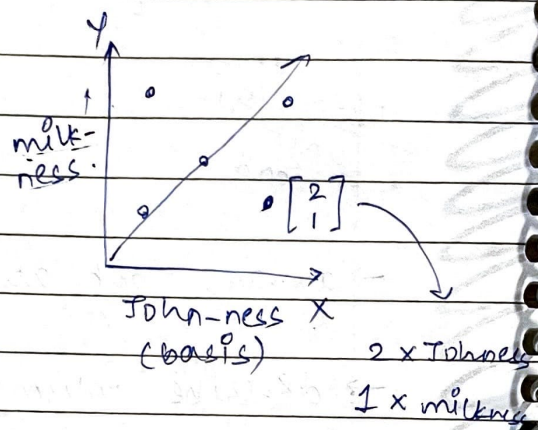
↓ ↓

$\{f_1, f_2 \cdots f_n\}$

$\{w_1, w_2, \cdots, w_n\}$

f_1 John
f_2 milk

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

↑ catness

← starting point of dataset

	John	milk
cat	2	1
dog	3	0
cow	1	4

$\times$ [2 x Johness; 1 x milkness.]

cat vector.

$= 2 * \begin{bmatrix} 2 \to \text{catness} \\ 3 \to \text{dogness} \\ 1 \to \text{cowness} \end{bmatrix} + 1 * \begin{bmatrix} 1 \times \text{catness} \\ 0 \to \text{dogness} \\ 4 \to \text{cowness} \end{bmatrix} = \begin{bmatrix} 5 \to \text{catness} \\ 6 \to \text{dogness} \\ 6 \to \text{cowness} \end{bmatrix}$

cat

$f: v_1 \to y \quad v^2$

__/__/____

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

$\hat{cat}$

(moving cat from one location to another)

↓

why change location?

→ Matrix multiplication = linear transformation (because its a modification of same entity)

8/8/23

Context Matrix

John | eats | love | context features ⇒ basis.

f_1 f_2 ... f_m

cat
cow
:
goat

f_i → intrinsic context
→ extrinsic context
→ hybrid (combo of both)

eg. cat

Intrinsic → f_1 f_2 f_3
"c" "a" "t"

extrinsic → f_1 f_2 f_3
John love milk

⇒ Intrinsic style.

eg. for phrase, intrinsic context can be words, characters.

for sentence, → words, phrases

⇓

Intrinsic ~~context~~ 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

$$P(f_i|w_j) = \frac{C(f_i, w_j)}{C(w_j)}$$

C → count

linguistic unit: words.

context unit: words (extrinsic)

Topic: (TF-IDF revisited)

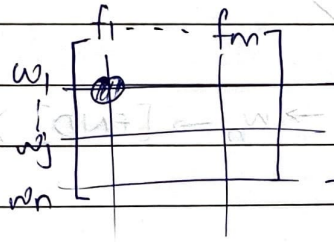
__/__/__

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

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

(expectation)

→ We try to understand the average info. content

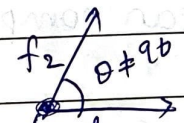


→ Initialize the

(data point can be any linguistic model)

We try to initialize context matrix using tf-idf.

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

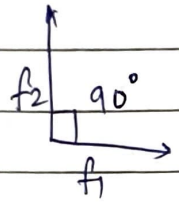


← linearly independent

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.

$\langle w_1 \rightarrow w_2 \rightarrow \ldots \rightarrow w_n \rangle$.

$\langle [START] \rightarrow w_1 \rightarrow w_2 \rightarrow \ldots \rightarrow w_n \rightarrow [END] \rangle$

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

↳ This is what any LM does

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

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

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

IMP ⊛ $P(w_1 \rightarrow w_2 \rightarrow w_3)$ (Here seq. matters)

$\neq$

$P(w_1\ w_2\ w_3)$ (Here order not imp.) (just they must appear together) } Joint prob

__/__/____

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

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

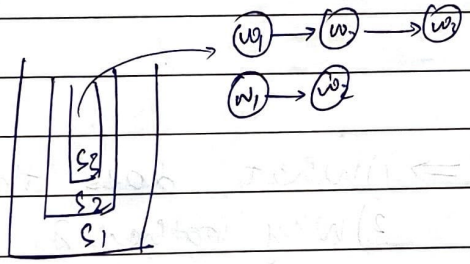
or

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

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

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

(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 from prob. point of view)

(Assumes that no influences of words on each other)

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

18/8/23

(*) N-gram Modeling
(as a particular case of Language Model)

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

too long
↳ computationally expensive.

↳ Markov Assumption ⇒
approximate $\approx$ $\hat{P}(w_n | \leftarrow w_{n-1})$ (bigram model)
degree = 1 (past only 1)

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

$\hat{P}(\text{END} | w_n)$ ⇒ 1) What does this mean?
2) Why bother?

↓
what's prob. of w_n being last one.

$P(A \rightarrow B \rightarrow C) = P(A) * P(B | \leftarrow A) * P(C | \leftarrow B \leftarrow A)$

↓ approx
↳ $P(C | \leftarrow B)$

⇒ Problem with considering short sequences:-
eg. John who lives in NY and works at Apple loves icecream

18/8/23

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.

⇒ Maximum likelihood →

likelihood

$$P(\text{Icecream} \mid \text{loves}) = \frac{L(\text{IC, loves})}{L(\text{loves})}$$

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

(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 ⇒ when N=0 in N-gram modeling.

(NP) Language Models ⇒ Skip-gram Modeling

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

↓

its influenced by both past & future context

Strong influence → suppress/boost

__/__/____

22/8/23

⑨ Skipgram Model (word2vec)

$P(w^c | w^t)$ w^t: target word (center word)

w^c: context word

n-gram → focuses on moving forward

$\prod_{j=m}^{+m}$

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 diff techniques (n-gram / skipgram)

all mutually independent = unigram model

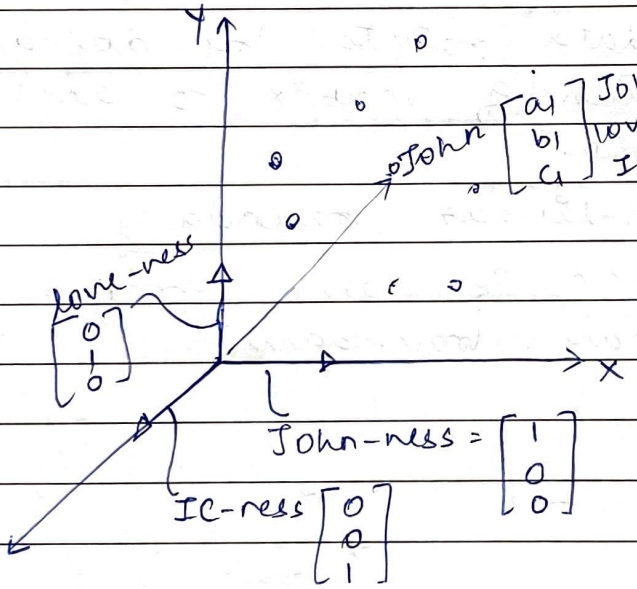
$\prod_{w^t=1}^{T}$ — T: all the unique target words

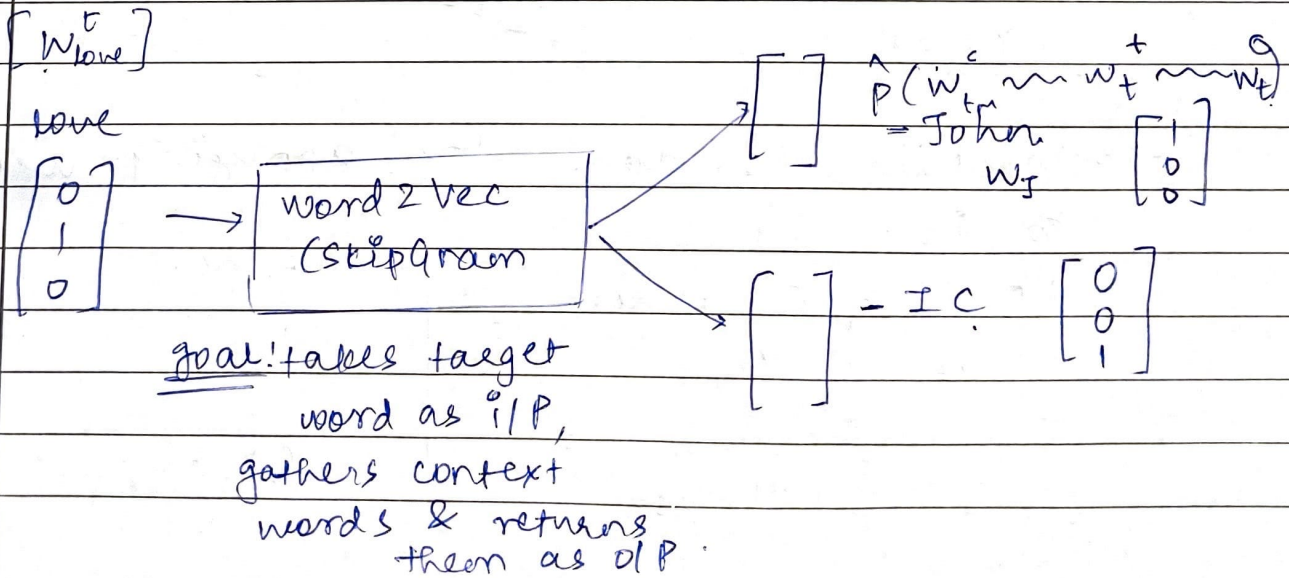
$$\prod_{w^t=1}^{T}\left[\prod_{j=-m}^{+m} P\left(w_{t+j}^{c} \mid w^t\right)\right]$$

↑ partition index effect

$w_i \rightarrow$ 2 avatars

(a) Word2Vec :-





24/8/23

vocab size $|W| = w_n$

features $f_1\ f_2\ \dots\ f_n$

Context matrix (M_c)

data points	f_1	f_2	...	f_n
w_1	0	0	...	0
:	:	:		:
w_n	1	1		1

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.

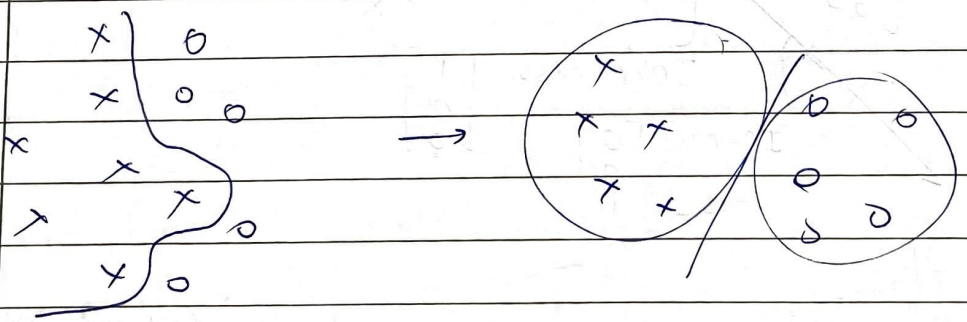
$$\hat{p}(start_0^c \rightarrow w_1^c \rightarrow w_2^c \rightsquigarrow w_t^c \rightsquigarrow w_n^c)$$

$$= \hat{p}(starts^c) \times p(w_1 | start) \times p(w_2^c | \leftarrow w_{1-}^c \leftarrow start)$$

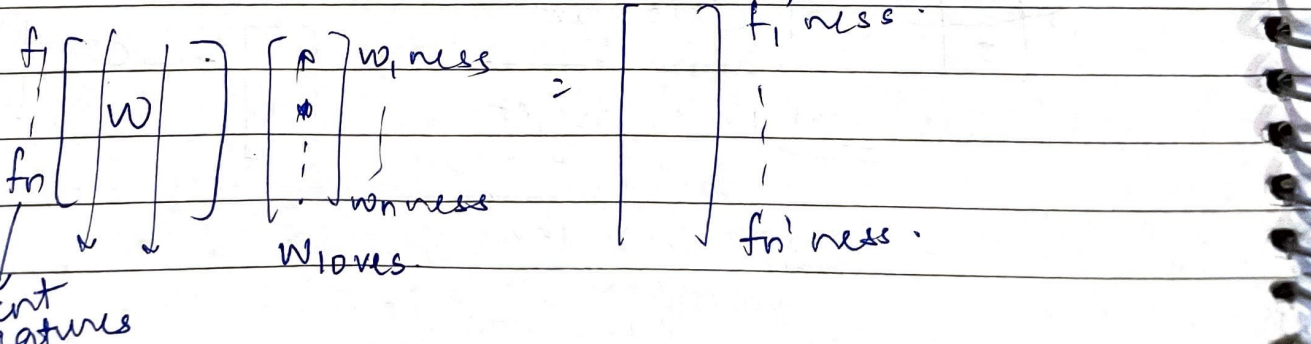
$$\times \dots \times p(w_n^c | \leftarrow w_{n-1}^c \leftarrow \dots \leftarrow w_1^c \leftarrow start^c)$$

→ 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.



Topic: Word2Vec

__/__/____

25/8/23.

① LM can be approximated using skip-gram modeling.

→ $\hat{p}(\text{START} \to w_1 \to w_2 \rightsquigarrow w_n \to \text{END}) = ?$

② Word2Vec follows skip-gram modeling.

③ Word2Vec does not attempt LM.

④ We can extend Word2Vec to do LM.

⑤ $\hat{p}(w^c_{t-m} \to w^c_{t-(m-1)} \rightsquigarrow w^c_{t-1} \rightsquigarrow w^c_{t+1} \to w^c_{t+2} \rightsquigarrow w^c_{t+m} \mid w^t_t)$

(sequence)

Word2Vec attempts to estimate this.

(just a join)

$\approx \hat{p}(w^c_{t-m}, w^c_{t-(m-1)}, \dots, w^c_{t-1}, w^c_{t+1}, w^c_{t+2} \dots w^c_{t+m} \mid w^t_t)$

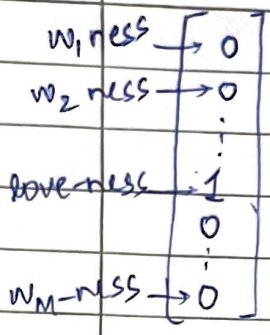
joint distribution of the context of words (w^c) given w^t_t.

⑤ is the first assumption that Word2Vec has

words (w^c) given w^t_t

$\approx \prod_{j=t-m}^{t+m} \hat{p}(w^c_j \mid w^t_t)$

__/__/____



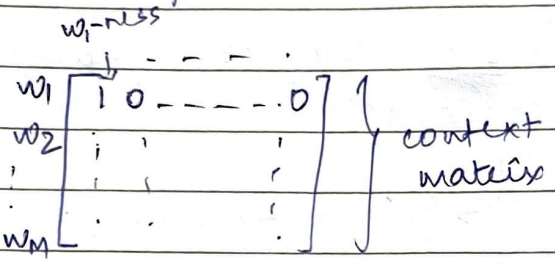
w_{love}

↳ one-hot vector.

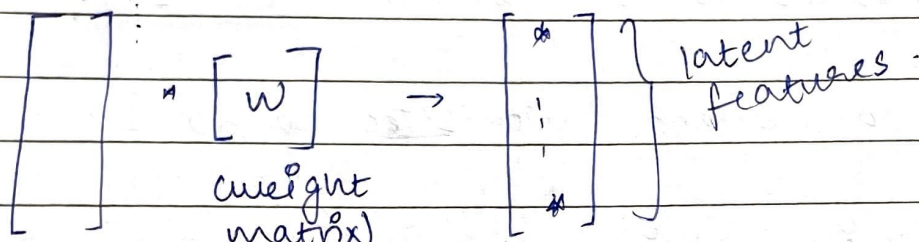
(Input)

Row → data points.

col → features



(w_1 only contains w_1-ness & so on)



w_{love} → input.

w^t_{love} → target

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

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

↓

side-effect

↓

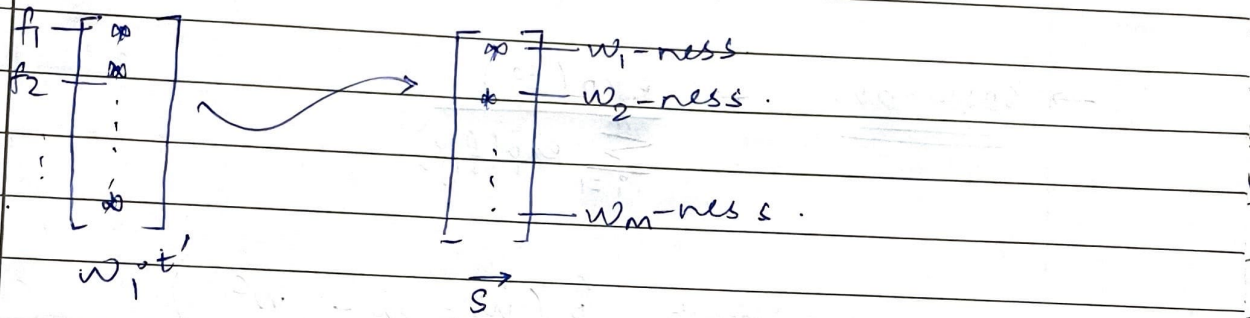
reach at the right location

(info contained in training data)

28/8/23

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

$\prod_i \prod_j P(w_j^c \mid w_i^t)$



has to be related to similarity fun^n.

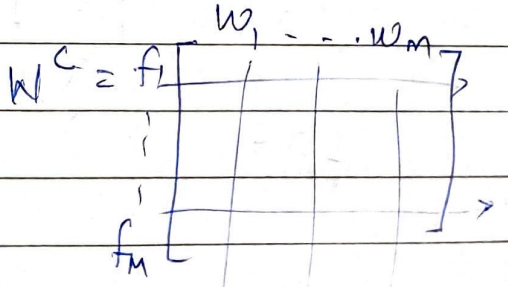
↳ $\cdot(\sigma)$ → Word2Vec: dot uses) product

$$w_i^{t'} \cdot w_j = s_{ij}^{oo}$$ (similarity using dot product)

$\Rightarrow w_i^{t'} * w_j^T = s_{ij}^{oo}$.

$\Rightarrow w_i^{t'} * W C^T = \vec{s}$

$W C^T * w_i^{t'} = \vec{s}$



→ Each word is unique.

→ From the world of vector space to probability space.

↓

$p(w_j | w_i^t)$ ← probability vector

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

Estimated prob: $\hat{p}(w_j = w_j^c \mid w^{ct})$

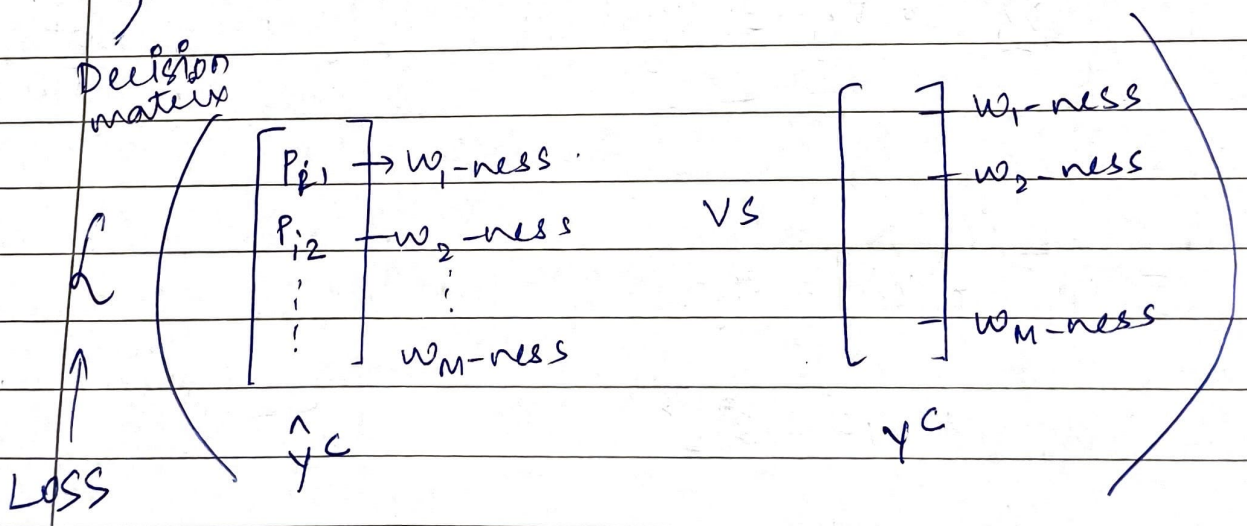
29/8/23

$w_i^{t'} = W^t * w_i^t = L(w_i^t)$ ——① (Linear transformation) → hot vector

$s_i^t = (W^c)^T * w_i^{t'} = f_d(w_i^{t'})$ ——② (Decision func^2)

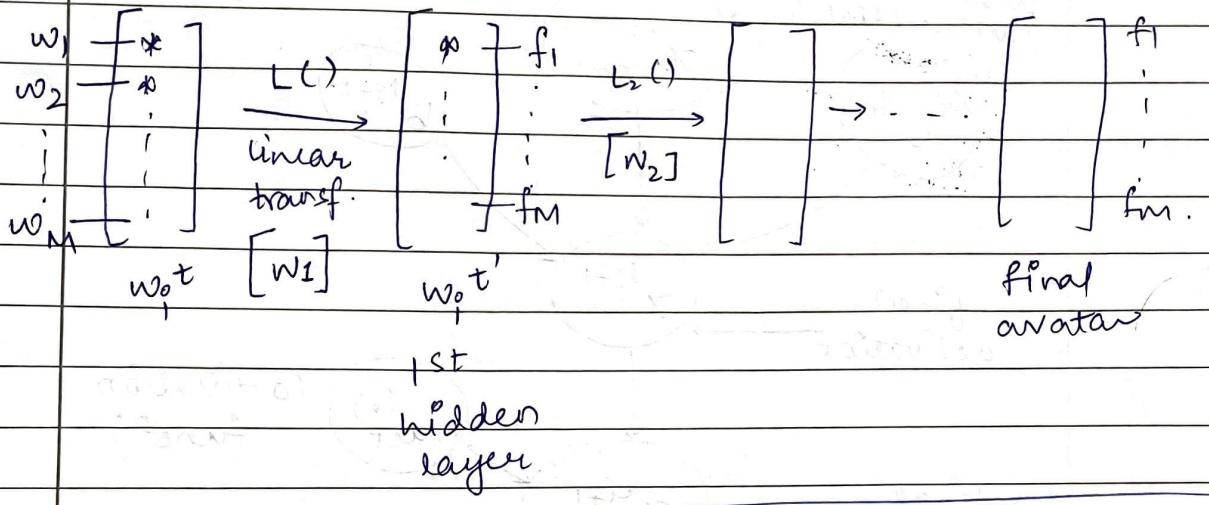
$\hat{y}^c = \hat{p}(w_j^c \mid w_i^t) = \text{softmax}(s_i^t)$ ——③

Decision matrix

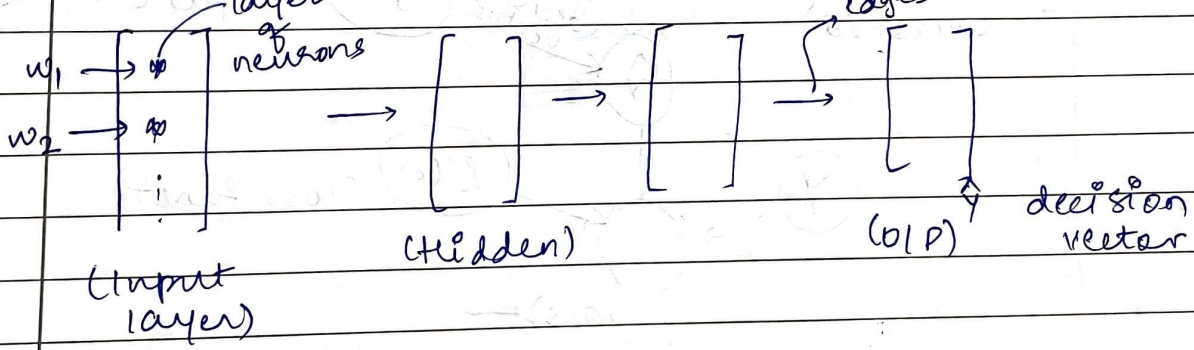


$$w^c \cdot w_i t' = (w^{cT} * w_i t')$$

FFA - Feed Forward Network

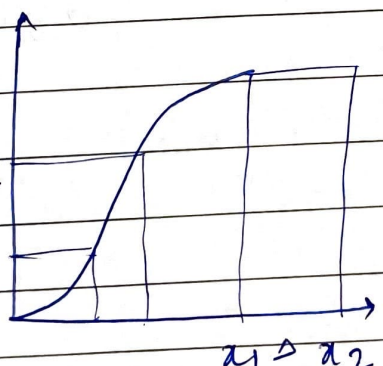


31/8/23 Input layer → Hidden layer → Output layer.



$$w_i t'' = w_j * w_i t' + b_j$$

⇒ logistic "activation" (squashing) function

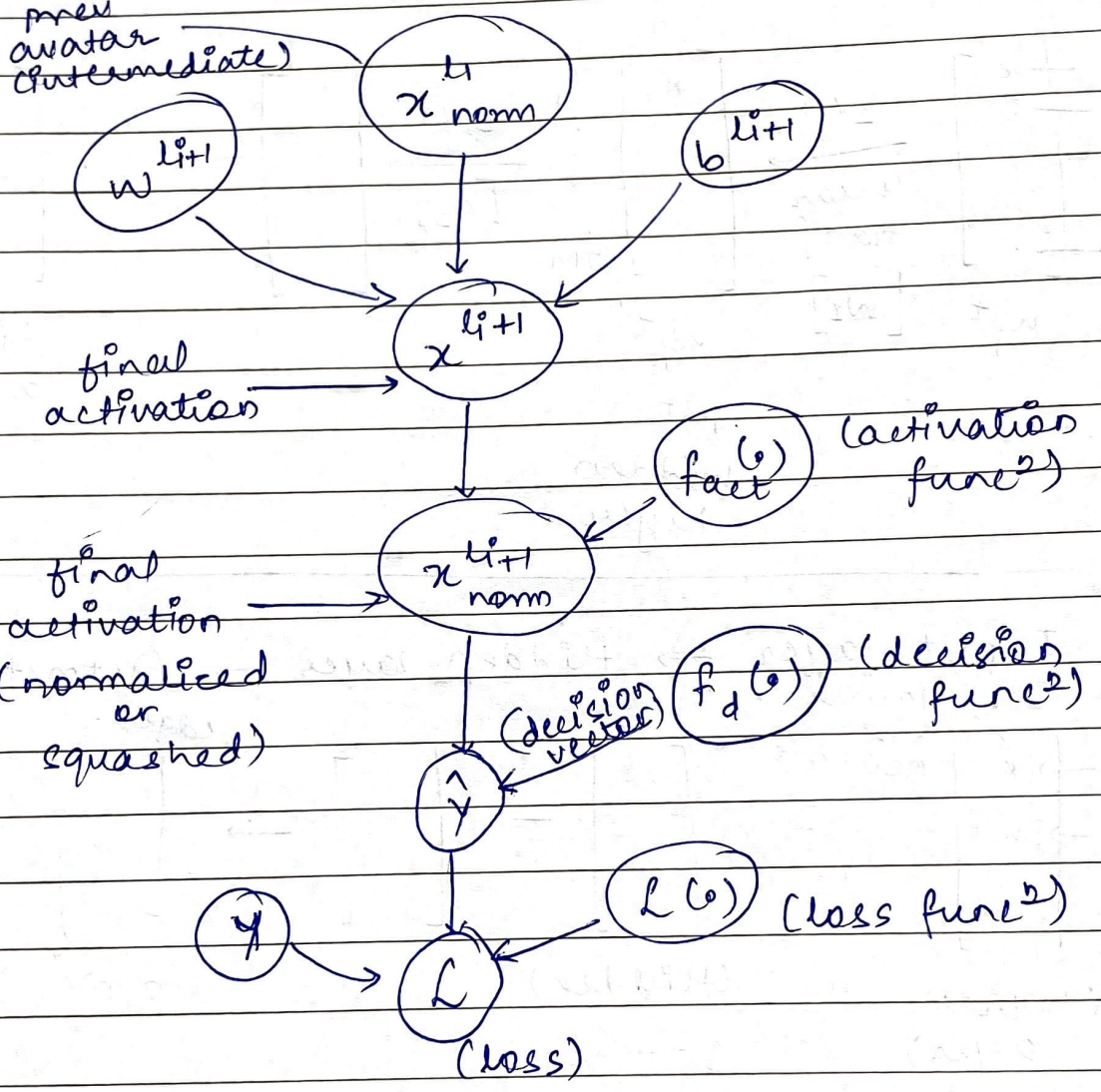


activation of neuron

(1st neuron - 1st axis :))

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

1|9|23 Topic:- Loss & Backpropagation



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

→ Error propagate top-to-bottom.

⇒ Why partial differentiation? $\frac{\partial L}{\partial w^{li+1}} = ?$

__/__/____

Chain rule.

$$\frac{\partial x^{li+1}}{\partial w^{li+1}} * \frac{\partial x^{li+1}_{norm}}{\partial x^{li+1}} * \frac{\partial \hat{y}}{\partial x^{li+1}_{norm}} * \frac{\partial h}{\partial \hat{y}} = \frac{\partial h}{\partial w^{li+1}}$$

Updation rule:-

$$w^{li+1} = w^{li+1}_{old} - \alpha \frac{\partial h}{\partial w^{li+1}}$$

[(stochastic) gradient descent]

α → learning rate (how much to update)

high (full trust on partial diff.) → faster convergence if there is convergence

disadv: overshoots

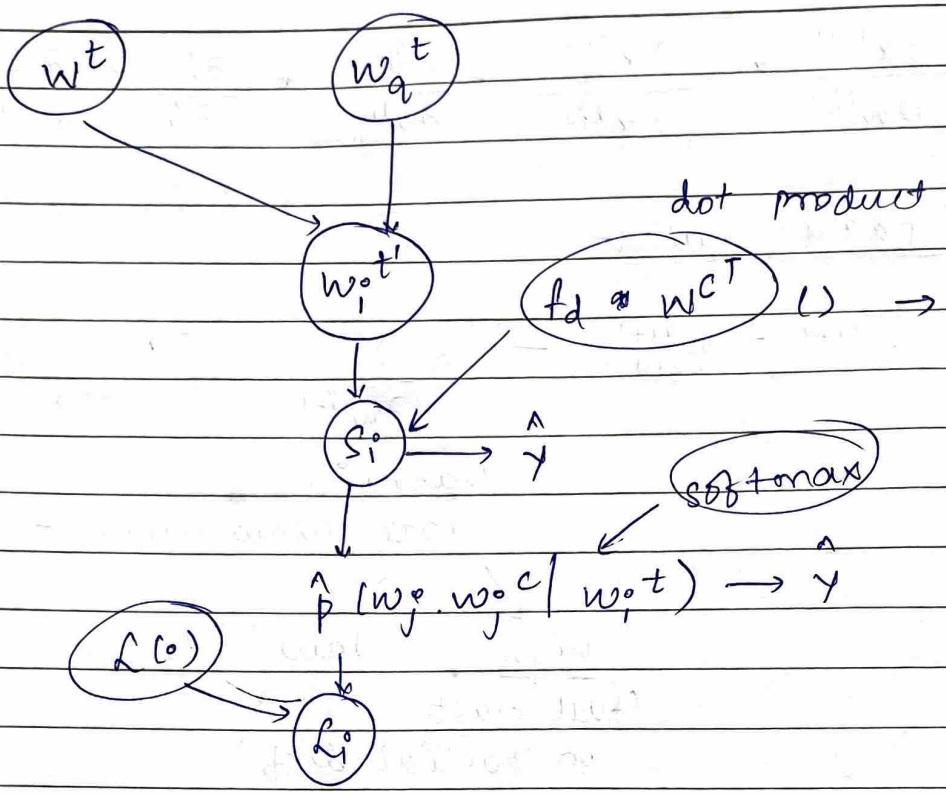
low

$$x^{li}_{norm} = x^{li}_{norm\,(old)} - \alpha \frac{\partial h}{\partial x^{li}_{norm}}$$

$$b^{li+1} = b^{li+1}_{old} - \alpha \frac{\partial h}{\partial b^{li+1}}$$

What's missing in word2vec:-

- No squashing
- No bias
- No activation func^{n}



loss func^2 → cross-entropy
(minimize the expected "surprise" of $P(w_j^c | w_i^t)$ → info content

TF-IDF – high info content.

2/9/23 Topic:- NN Update rule & Word2Vec.

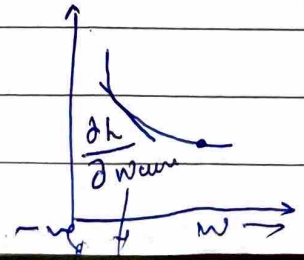
(#) Update rule:- on basis of G.D., S.G.D.

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

α → learning rate

Why subtract? → convex func^2

Cross entropy
-ve log likelihood

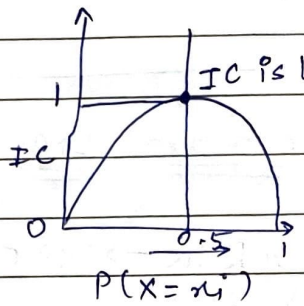


__/__/____

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

Surprise = Information content (IC)

$$IC(X = x_i^\circ) = \log \frac{1}{P(X = x_i^\circ)}$$



$$\therefore IC(X) = \sum_{X = x_i^\circ} \log \frac{1}{P(X = x_i^\circ)}$$

(X takes discrete values (RV))

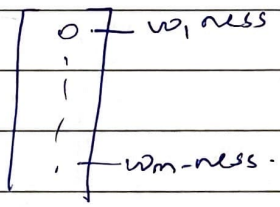
$$\text{Min } IC(X) = \min \sum_{X = x_i^\circ} \log \frac{1}{P(X = x_i^\circ)}$$

$$= - \min \sum_{x_i^\circ} p \log P(X = x_i^\circ)$$

eg. If it is a classification problem, $x_i^\circ \rightarrow$ classes.

In word2vec, $x_i^\circ \rightarrow$

$\hat{y} \Rightarrow P(w_j^\circ = w^c \mid w_i^t)$ ← Input



$\sum P(w_i^\circ \mid w_i^t) = 1$ (bcoz of softmax)

↑ Minimize the surprise given the context

Cross entropy (no ignoring)

Entropy → expected surprise.

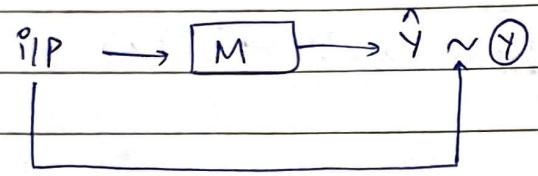
$$H(Y, \hat{Y}) = \sum_i \left[y_i^\circ * \log \frac{1}{P(\hat{Y})} + (1 - y_i^\circ) * \log \frac{1}{P(1-\hat{Y})} \right]$$

(what I want) (what I don't want)

__/__/____

14/9/23 Topic:- n-gram LM using RNN.

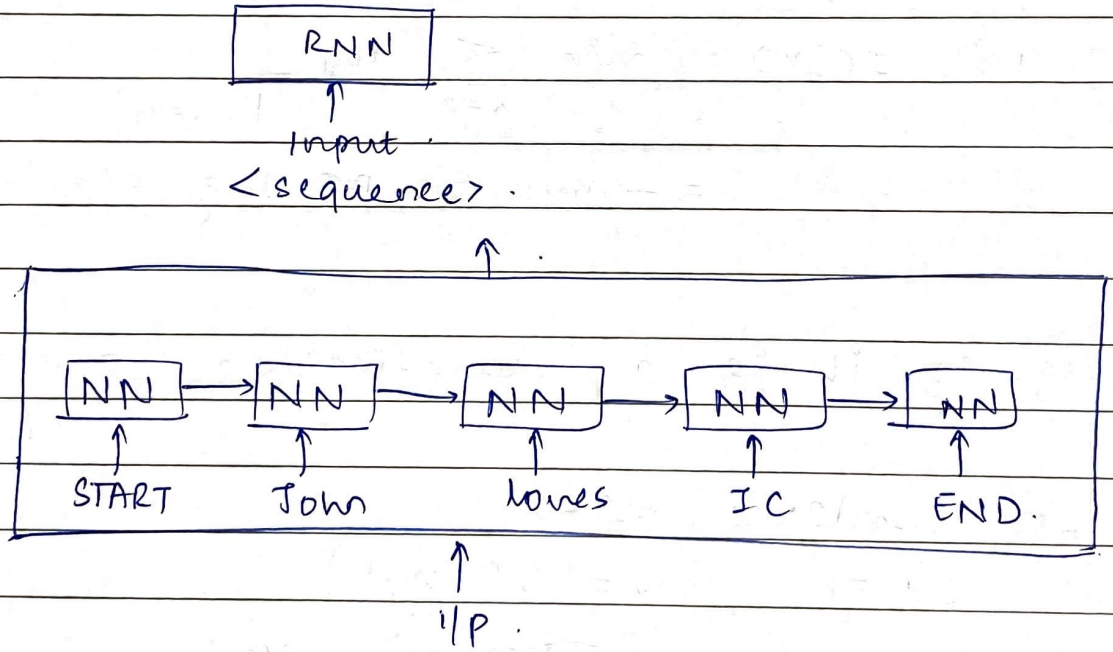
→ self-supervised setup.



⇒ n-gram LM using RNN:-

↗ RNN → 1990.
↘ Word2Vec → 2013

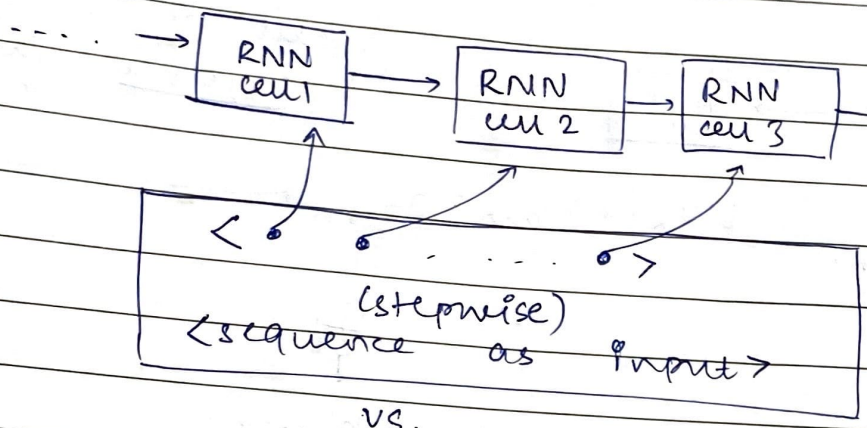
(R)NN
recurring.



follow i.i.d. assumption
(independent & identically distributed)

15/9/23

RNN.



vs.

<set as an input> → word2vec (CBOW)

vs.

<parts of seq as i/p> → word2vec (skipgram)

Conditional independence.

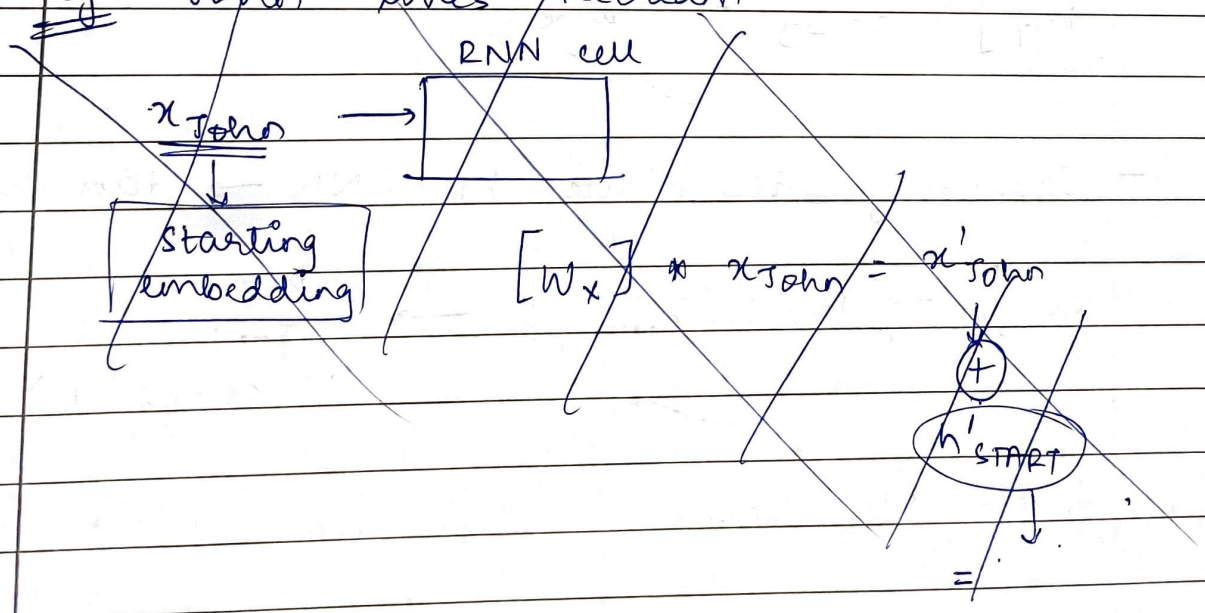
skipgram :- $\prod_j P(w_{j \in c} | w^t)$

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

c: context set of w^t.

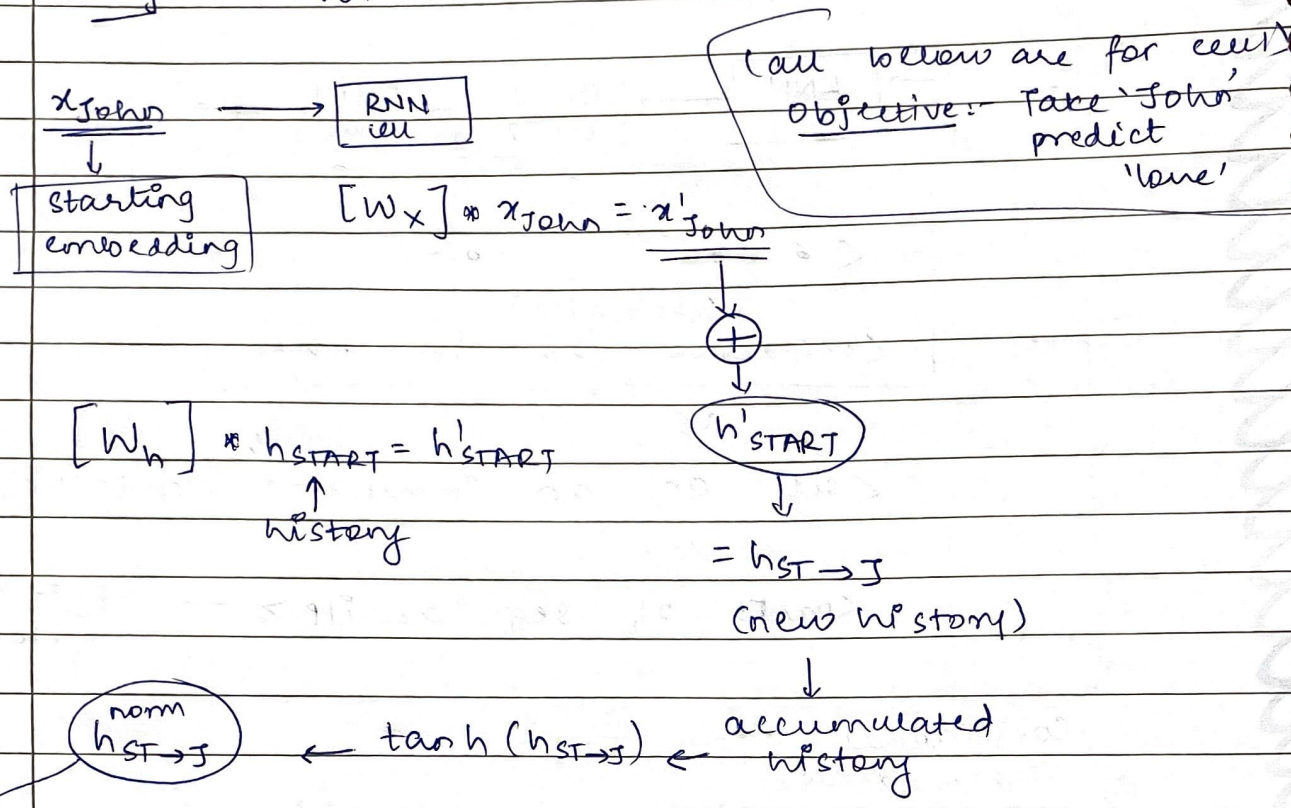
[what are key takeaway of word2vec?]

eg. John loves icecream



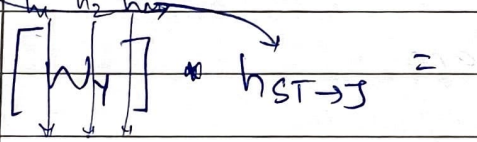
__/__/____

eg. John loves icecream.



decision func^n

$f_d(h_{ST \to J}) \longmapsto \hat{y} \to$ hopefully is 'love'



→ Squashing technique in RNN → tan h

$h_{ST \to J} \to \tanh(h_{ST \to J}) \to h^{norm}_{ST \to J}$ (squashed h)

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

↓

go into world of —nesses

softmax $\hat{y}_{love}$ $\hat{p}$. $\hat{p} = \text{softmax}(\hat{y})$ $\text{argmax}(\hat{p})$

21/9/23.

__/__/____

RNN

$$x_t' = W_x x_t + b_x$$

(input)

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

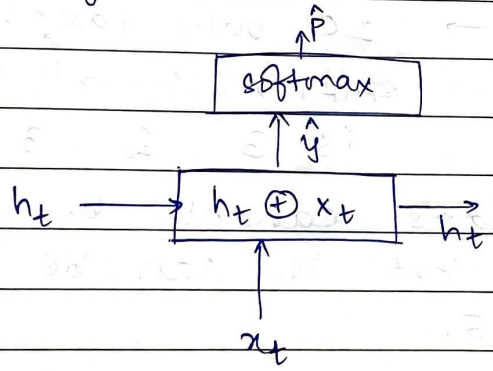
$$h_t = \tanh(h_{t-1}' + x_t')$$

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

(decision func^n)

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

→ gives the next linguistic unit.



→ In word2vec we are going from world of w's to f's (features)

vocab size $|V| = 4$

0.7	w_1	w_1-ness → "sell"-ness	
0.2	w_2	w_2-ness → "oven"-ness	
0.05	w_3	w_3-ness → "love"-ness	
0.05	w_4	w_4-ness → "sleep"-ness.	

$\hat{P}_y$

Here we are pathetically wrong (bcz love has least)

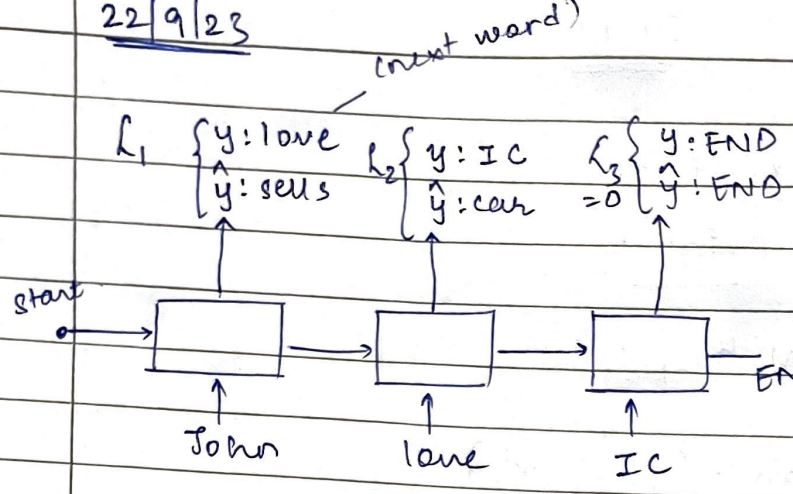
P_y →	0	sell
	0	oven
	1	love
	0	sleep.

hot vector

this should be expected

loss → $\mathcal{L}_t(\hat{P}_y, P_y)$

22/9/23



$$L = L_1 + L_2 + L_3$$

(ŷ)

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 ≤ 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 → 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) of poor child

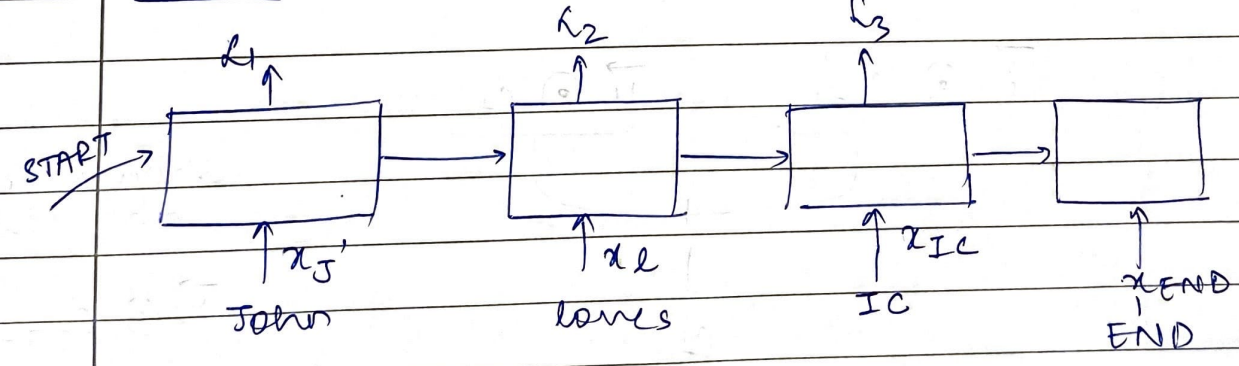
how much to allow in this cell (C_t)

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
↓
unwanted noise should be removed
↓
(forget gate)

② f. (func^2) → acts as a reduction agent (squashing func^2)

→ Why elementwise ?

⇒ How f is generated ?

$$\begin{bmatrix} \\ \\ \\ f \end{bmatrix} = \sigma\left[W_f^t \cdot \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} \right]$$

(needs to be learned)

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

↙ f

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

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

$+$

↓

c'_t

should be removed

$$\left(\vec{f} \odot c'_{t-2}\right) + \left(\vec{u} \odot c_{t-1}\right) = c'_{t-1}$$

Vanilla RNN:-

$h_t = \tanh(W_h * h_{t-1} * b_h) + (W_x * x_t + b_x)$

↳ C_t ("raw" new context (or new history))

→ When 'f' becomes 0 vector, u becomes 1 vector, then, LSTM becomes vanilla RNN.

$$\begin{bmatrix} \\ \\ \end{bmatrix}_u = \sigma\left[W_u^{**} \cdot \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} \right]$$

u (needs to be learned)

$$= \sigma\left[W_u^{*} * x_t + W_u^{*} * h_{t-1} + b_u \right]$$

$$\begin{bmatrix} \\ \\ \end{bmatrix}_O = \sigma\left[W_O^{**} \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} \right]$$

O output filter

$$\sigma\left[W_O * x_t + W_O * h_{t-1} + b_O \right]$$