NLP ⇒ Introduction

DATE

⇒ NLP ⇒ Natural Language Processing

NLP : All the preprocessing done on raw text before NLU is NLP

Natural Language ⇒ Formal Language (It is Language of Linear Algebra)

FL ⇒ First Order Predicate Logic

Compositional Semantists : $NL \overset{T(.)}{\Rightarrow} FL$ This process is goal in case of compositional Semantists
↳ (Worried abt meaning) $T(.)$

Semantics should have well formed formula.

Distributional Semantists : $NL_1 \overset{T(.)}{\Rightarrow} FL_1$, $NL_2 \overset{T(.)}{\Rightarrow} FL_2$ if $NL_1 \backsimeq NL_2$ then $FL_1 \backsimeq FL_2$ } or $f_{sim}(NL_1, NL_2) = Y$ } $f_{sim}(FL_1, FL_2) = Y$
↳ (Worried about similarity scores)

FL is something that can be easy interpreted by the computer / BOT

How come we can do $f_{sim}(NL_1, NL_2) = f_{sim}(FL_1, FL_2) = Y$
NL can't be understood by computer. So it is done by human touch but it is highly subjective. To overcome this we will keep large no of people as subject for computing $f_{sim}(NL_1, NL_2)$.

One way is take avg of scores of subjects.

In case of Distributional Semantists the $T(.)$ can be loose transformation or fuzzy transform of NL. Here meaning capturing is not important. Everything should be consistent whether it is mistake or correct take.

⇒ Language ⇒ Syntax + Semantics + pragmatic
(Grammar) (Literal Meaning) (intented meaning)

Homonymy → Same word different meaning (Jaguar)

Polysemy → Same word slight difference in meaning (root meaning is same)
(Bank)

Phrase :- Sequence of words that have collectively a meaning in dictionary.

DATE

For pragmatics, we need more context to understand the intended meaning.

Context Window : The size of window to understand context of the sentence.

C=20 mean ±20 from current sentences.

Form : S V O
Subject Verb Object

Syntax

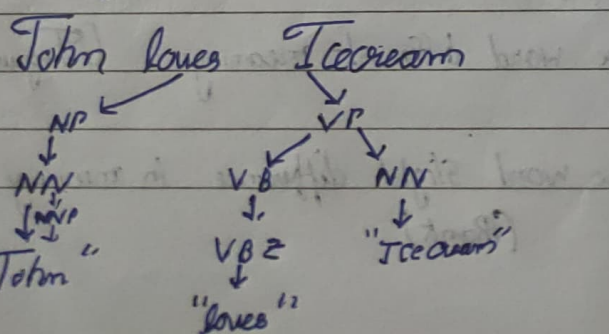
- Form (SVO)
- Part of speech "Sequence" (POS)
- Parse tree (context free)
- Dependency Parsing (DP)

is it valid or not ? => There are some sentences which are grammatically correct but has no meaning.
e.g colourless dreams sleep furiously.

Here the Compositional syntactists transform the sentences & get back to original sentence after considering proper grammar parsing but it is still of no use.

Any sentence is composed of : Noun phrase & Verb phrase

|V| = Σ lexicons
[cardinality of Vocabulary]



VBZ → Singular Verb

DATE

If parse tree goes wrong then everything goes wrong

Slang ⇒ When there is usage of any kind of slang & it maintains the same structure as in training data then it will consider similar pos tag for the unknown phrase.

DP ⇒ In dependency parsing the verb is made the pivot & the rest of the things are figured out

If we can assign truth value to a sentence then we can understand that it is meaningful or not even though it is grammatically correct.

DATE

⇒ If two things are similar they should share some similar properties characteristics.

$[w_{j-2} \rightarrow w_{j-1} \rightarrow \text{house}(w_j) \rightarrow w_{j+1} \rightarrow w_{j+2}]$ } might happen

$[w_{i-2} \rightarrow w_{i-1} \rightarrow \text{rent}(w_i) \rightarrow w_{i+1} \rightarrow w_{i+2}]$

Similar frequency distribution

- If there is overlapping of terms to great extent then it can be claimed that two terms are similar otherwise we will call it related.

- Same places ≈ Wherever the term A is occuring term B should occur
- Same friends ≈ Words in context window A are same as of B
- Same frequency ≈ The histogram of frequency of A & B are same

If these 3 conditions are met then they are similar.

- The problem is eventhough A & B are similar but the model (Distributional semantic Model) won't consider it similar due to the above 3 conditions are not fulfilled in training data. it will tell A&B are related.

→ Now onwards similarity is releatedness.

⇒ Vector Space Modelling

- Represent a "linguistic unit" as a vector.

linguistic unit : character, word, phrase, clause, sentences, short paragraph, long paragraph, document.

Context Matrix :

name of the matrix is set on basis of columns.

DATE

Context Matrix

features →

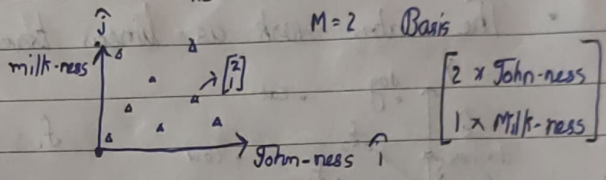
cat
dog
goat
cow

The features / context comes from context window.

- features form basis.

⇒ given a set of features ($f_1, f_2, \ldots f_M$) & set of datapoints $\{w_1, w_2, \ldots w_N\}$

f_1 : John
f_2 : milk
↳ denotes the words associated with each one of datapoints.



Initially: John ≡ Johness

Context - Matrix

	John	milk
cat	2	1
dog	3	0
cow	1	4

John is in 3D
Cat is in 2D

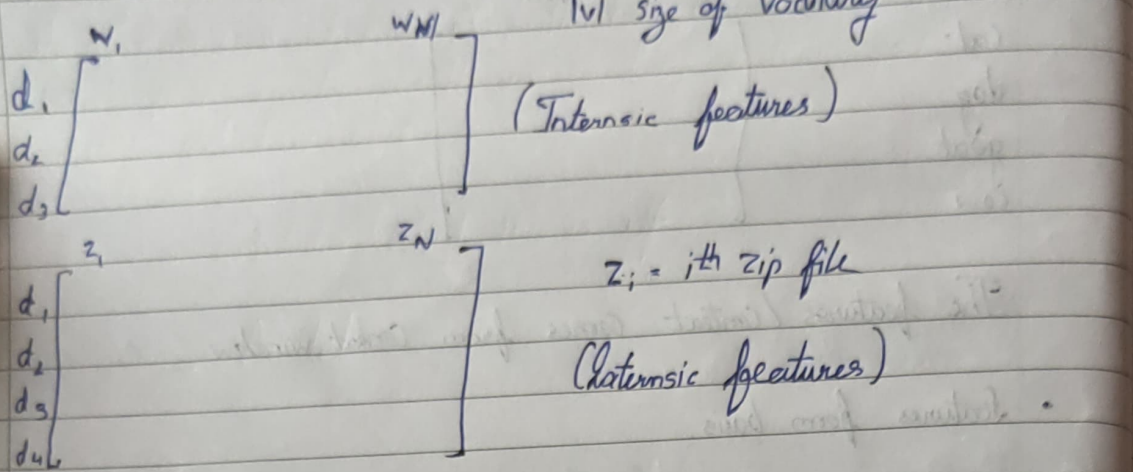
$$\text{cat} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} = 2\begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} + 1\begin{bmatrix} 1 \\ 0 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \\ 6 \end{bmatrix}$$

Now cat in 3D

The features you take due to co-occurance then it is an extrinsic features

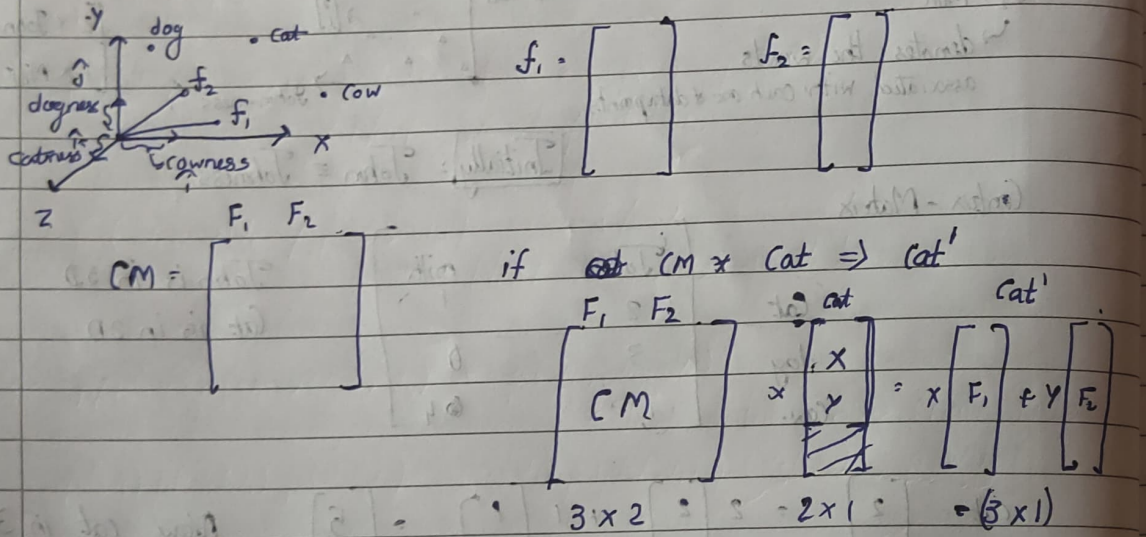
When we use representation of contents inside them are intrinsic features.

DATE



When modelling of documents is done based on extrinsic features most of times it is not ~~*~~ useful.

- The matrix work as linear transformation



- If you are not able to get better performence then the culprit is the Content Matrix as it can't transform things efficiently.
- Therefore, the content matrix is needed to be learnt.
- You can randomly initialize CM (May be values from gaussian dist) then start training & check output formation.

DATE

- TF-idf ⇒ Term Frequency - inverse document frequency.

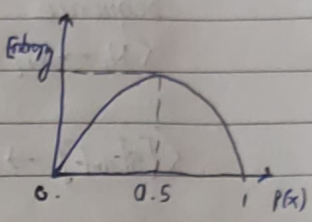
If we know that a particular feature will surprise you / or help you to classify similar & non-similar content.

Q How to compute information content?

⇒ $I_C(f_i) = \log\left(\frac{1}{p(f_i)}\right)$ = $I_C(x) = \log\left(\frac{1}{p(x)}\right)$

Most surprising state when $p(x)$ reaches 0.5

$H(x) = \mathbb{E}(I_C(x)) = \sum_x p(x) \cdot \log\left(\frac{1}{p(x)}\right)$: Entropy



- idf = $\frac{1}{\frac{\text{d}_i \text{ in } N\text{docs}}{N}} = \frac{N}{\text{d}_i \text{ in } N\text{doc}}$ ∝ $\frac{1}{p(x)}$ now if we take log it is similar.

- TF: we get I_C of (f_i) but we want expected value for f_i (Surprise value of F_i knowing we have cat) $= \frac{\sum_{d_i} W_{cat \in d_i}}{\sum_i W_{E d_i i}}$

⇒ Initializing a content matrix

$$CM = \begin{matrix} \\ d \downarrow \end{matrix} \begin{matrix} f \rightarrow \\ \begin{bmatrix} \theta & * \\ \amalg & \square \end{bmatrix} \end{matrix}$$

Calculation of each cell is independent of row & column (i.e assuming all the features are independent of each other) [orthogonal] ↘ (BOW = Bag of Words)

But it is false as the language models depends on the assumption that words are dependent so are the features which will help in predicting that the missing words.

John loves ____ ← (Ice-Cream) → Prediction.

There is a flow in the language otherwise those modelling techniques won't exist.

DATE

We can initially initialize CM with help of TF-idf it is better than random. If you are doing so than you are adding a inductive bias.

The main contributor behind this is idf.

⇒ Language Model

Sequence

$\langle w_1 \to w_2 \to w_3 \to \ldots \to w_n \rangle$

$\langle [start] \to w_1 \to w_2 \to w_3 \to \ldots \to w_n \to [End] \rangle$

$\hat{p}(w_1 \to w_n) = ??$ → This estimation is what language model does

$\hat{p}(w_n / w_1 \to w_{n-1}) = ??$ (Yes can be computed)

↳ Next word prediction.

$\hat{p}(w_i \to w_e / w_1 \to w_{i-1}) = ??$ (Yes)

↳ Next Sequence prediction

Note : $p(w_1 \to w_2 \to w_3) \neq p(w_1, w_2, w_3)$

↳ Joint distribution ⇒ Chain Rule.

$p(w_1)\, p(w_2/w_1)\, p(w_3/w_2) = p(w_2)\, p(w_2/w_2)\, p(w_3/w_1) = p(w_3)\, p(w_2/w_3)\, p(w$

Therefore, in RHS Order doesn't matter

$p(w_n / \leftarrow w_{n-1} \leftarrow w_{n-2} \leftarrow \ldots w_1 \leftarrow start)$

N-gram Modelling

But if we consider Markov's chain Rule & use $p(w_n / \leftarrow w_{n-1})$ then this is bi-gram model.

We also have to compute $p(End / w_n)$. It is needed to be calculated because we need to ensure that the sentence is ending there and there is no extension to it.

DATE

⇒ Why not bigram? It can't sustain longer dependencies.

Eg John, who lives in NY and works at Appl, loves Ice-cream.

Bi-gram ⇒

Tri-gram ⇒

still context is not understood.

But if we consider large degree for consideration then there is no meaning to consider Markov's rule.

$\hat{P}(IC / \leftarrow loves) = \frac{C(loves \rightarrow IC)}{C(loves)}$ { This is something We are interested in finding

⇒ Bag of Words model can be used when we don't have to consider sequence

⇒ Skip-gram Modelling (Word2Vec)

If we have to estimate $\hat{p}(w_1 \rightarrow w_2 \rightarrow \ldots \rightarrow w_n)$.

In skip gram modelling it says if we have to figure out $w_1 < w_i < w_n$ then we have to consider sequence $w_i \rightarrow \ldots \rightarrow w_{i-1}$ as well as $w_{i+1} \rightarrow \ldots \rightarrow w_n$.

Problem with n-gram: John → loves → Ice-cream

$p(IC / \leftarrow loves \leftarrow John) = \frac{C(John \rightarrow loves \rightarrow Ice\text{-}cream)}{C(John \rightarrow loves)}$

- There might be few John → loves in the data so denominator problem
- There are some John → loves → Ice-cream counts so the prob decrease.
- In some cases we can say directly from John that ice-cream should come but due to n-gram we are penalizing it.

$p(w^c / w^t)$ $w^t \Rightarrow$ target word w^c: context word.

$w^c = w^c_{j+t}$ the context word is at j offset from target word.

classmate PAGE

DATE

In case i want to take bigger context window then...

$\prod_{j=-m}^{+m} p(w_{t+j}^c / w^t)$ ⇒ Context window = 2m

$\{ w_{t-m}^c \; w_{t-2}^c \; w_{t-1}^c \; \underbrace{w_t^t}_{\text{target word}} \; w_{t+1}^c \; w_{t+2}^c \ldots w_{t+m}^c \}$

p(sequence)

So we are trying to find probability of $p(w_{t-m}^c \rightsquigarrow w_t^t \rightsquigarrow w_{t+m}^c)$

Here we are taking assumption that they are conditionally independent otherwise if they are totally independent then it would be Bag of words

Conditional independence of all w^c's

If you have multiple target words then

$\prod_{t=1}^{T} \left[\prod_{j=-m}^{+m} p(w_{t+j}^c / w^t) \right]$ → Maximize this

⇒ Word2Vec takes target word as input & gives set of context words as output. While learning this it learns the weight values for target words [Embeddings]

→ Word2Vec doesn't attempt LM

→ We can extent Word2Vec to do Language Model

classmate

PAGE

DATE

→ Similarity function function (Word2Vec) : Dot product

$W_t^{t'} \cdot W_j^T = S_{ij}$: Similarity Score

Input hot-vector $[0, 0\ 0\ 0\ 1\ 0\ 0]$ W_i^t

desired output : $\{W_j^c\}$ → this will be found by calculating similarity with W^c

Now we have similarity matrix $[\ ]$ so we will convert it into $\begin{bmatrix} \frac{s_i}{z_a} \\ \frac{s_2}{z_{s_j}} \end{bmatrix}$ probability matrix

but there is a chance that one similarity can dominate rest we may get skewed matrix, so instead of this we can use softmax $\left[\frac{\exp(s_i)}{\sum_i \exp(s_i)}\right]$

$W_i^{t'} = W^t * W_i^t = L(W_i^t)$ — ①

W^t → are embeddings in word2vec

$S_i^t = (W^c)^T * W_i^{t'} = f_d(W_i^{t'})$ — ②

- W^c → Context matrix

$\hat{y}^c = \hat{P}(W_j^c / W_i^t) = \text{softmax}(S_i^t)$ — ③

decision vector

$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{|v|} \end{bmatrix}$ ← Winess, vs, ← Winess

$\hat{y}^c$ y^c

- You know how many 1's should be there at which position in y^c

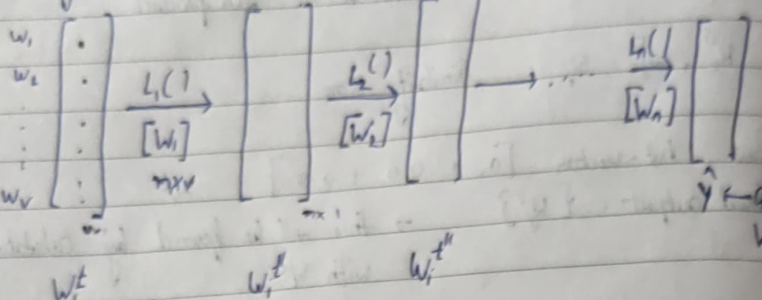
Now we have two vectors so we will decide about loss, $L(\hat{y}^c, y^c)$

$[W_{final}^t] * W_i^t = \mu^t$ loss is cross entropy

DATE

=> discon
Gradec
Jrmetri
Consider

=> Some goal with FFN



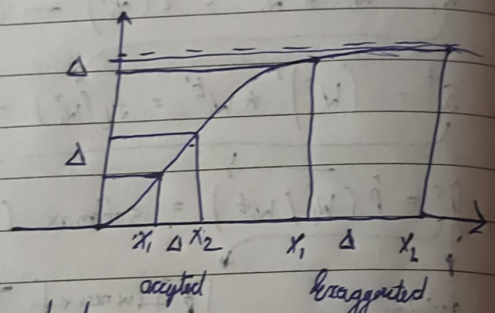
$w_i^{t''} = w_j * w_i^{t'} + \vec{b_j}$: bias : safety parameter

=> What is happening at hidden layers

- By applying activation function we are squashing the vectors

$w_i^{t''} = f_{act}(w_i * w_i^{t'} + \vec{b_j})$

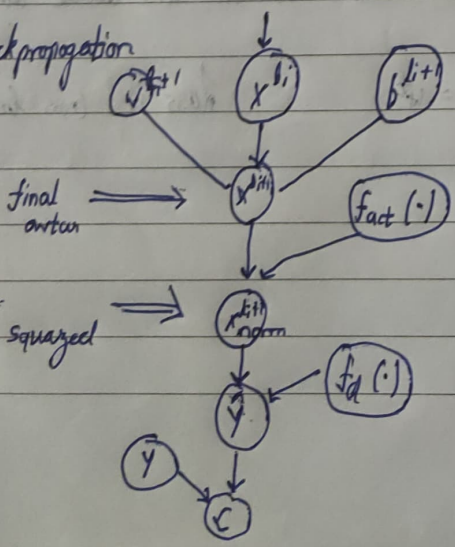
$f_{act}\left(\begin{bmatrix} \cdot \\ \cdot \end{bmatrix}\right) = \begin{bmatrix} f(\cdot) \\ f(\cdot) \end{bmatrix}$ ⊬



- it will squash more the unimportant features & squash less the important features (this is why activations are important)

all 5 type of activation can take care of this exaggeration Sigmoid, tanh & softmax

=> • Loss & Backpropogation



classmate

PAGE

DATE

Q How to know about the role of each component in giving wrong decision?

A So we will partially derivate the loss function w.r.t all components.

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

This is backpropogation of error

So now error correction or updation rule

$$W^{li+1}_{(New)} = W^{li+1} - \alpha \frac{\partial L}{\partial W^{li+1}} \quad -(1) \Rightarrow \text{Stochastic Gradient Descent}$$

$$x^{li}_{norm} = x^{li}_{norm} - \alpha \frac{\partial L}{\partial x^{li}_{norm}} \quad -(2)$$

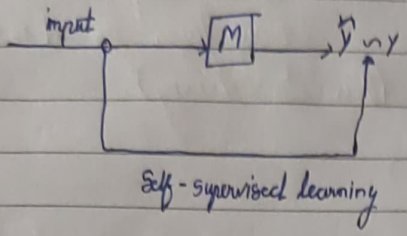
$$b^{li+1} = b^{li+1} - \alpha \frac{\partial L}{\partial b^{li+1}} \quad -(3)$$

Similarly

$$W^{li} \rightarrow \frac{\partial L}{\partial W^{li}} = \frac{\partial x^{li}}{\partial W^{li}} * \frac{\partial x^{li}_{norm}}{\partial x^{x_i}} * \frac{\partial L}{\partial x^{li}_{norm}}$$

DATE

⇒ How Word2Vec uses skipgram modelling?



• We can train Word2Vec model to work as language model

☆ n-gram Language Model using RNN

RNN → 1990 Word2Vec → 2013

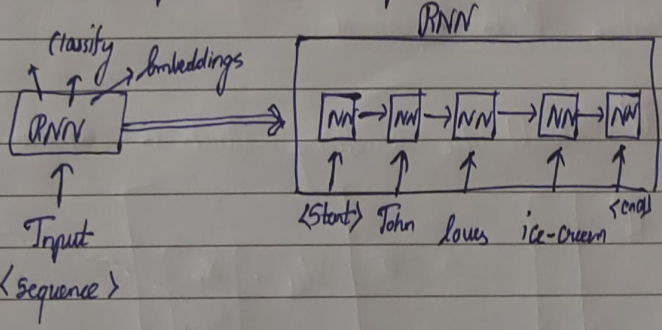
RNN is not a fixed or particular type of model. It considers sequential data.

We can create our version's of RNN. Mostly used RNN is LSTM

R → Recurrent (Recurring) Neural Networks

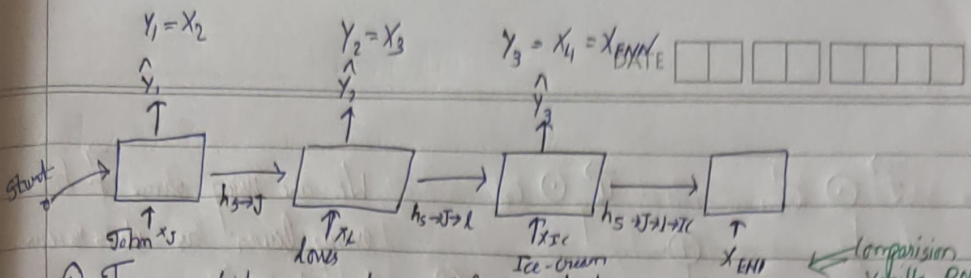
Word2Vec considers part of the sequence at a time as input. So next loss updates is independent of previous loss update.

It assumes that the words follow IID (Independent & Identical Distribution)



There will be different embeddings for mary in mary loves jonh & jonh loves marie in RNN where as in Word2Vec it will be same.

Structure architecture will be come throughout the sequence.



comparision with Vahilla RNN

① Incoming history needs modification before can be used

② Basic structure remains same, knock off structures which can confuse the model. The past can be carried to certain point continuing further will confuse the model. This the job of forget gate

Forget gate : It is a kind of vector, with which we are doing element-wise multiplication of Incoming history.

$\vec{f}$ ⊙ $h'_{S \to J}$ = C'

C = Content inside previous cell
C' = " " modified cell

Forget gate Incoming history

(Hum shadow Reduce kar rahe h kuch messes ki)
↳ We are forgetting certain messess & we are keeping the rest as it is

Therefore, $\vec{f}$ should lee like a reducing agent i.e squashing function.

$\vec{f}$ used in LSTM for squashing is (Sigmoid)

Q ∘ Why element wise multiplication? ⇒ Beez we need to work selectively

⇒ our $\vec{f}$ is sigmoid over something $= \sigma\left[W_f^t \cdot \begin{bmatrix} X_t \\ h_{t-1} \end{bmatrix}\right]$

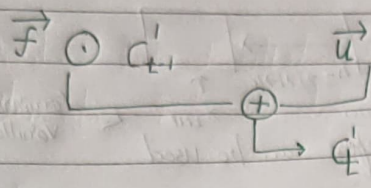
$= \sigma\left[W_f^t * X_t + W_f^t * h_{t-1} + b_f\right]$

$\vec{f}$ needs to be learnt

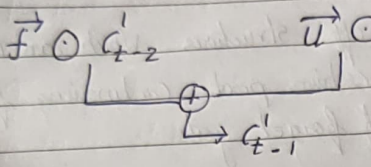
If i get $\vec{f}$, I will element wise multiplication with C'_{t-1}

fo C_{t-1} → modified history at time t

DATE



We are using same $\vec{f}$ for every layer as it is recurrent (Parameter sharing)



In RNN

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

- h_t in RNN $\backsim$ C_t in LSTM (Raw New content or new history)

But we are not happy with naw version, we want some part to be removed

the $\vec{u}$ is important beeze it helps in redudancy removal

To cheeze mene $\vec{f} \odot C'_{t-1}$ pe through dalla wohi same cheeze $\vec{u} \odot C_t$ daalega to wo redudancy ho jayegi isliye $\vec{u}$ is important.

$\vec{u}$ should be something which is similar to

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

→ If our $\vec{f}$ & $\vec{u}$ are identity ~~matrix~~ vectors it will be special case of LSTM.

→ If $\vec{f}$ is $\vec{0}$ & $\vec{u}$ becomes $\vec{1}$ then LSTM will be Vanilla RNN

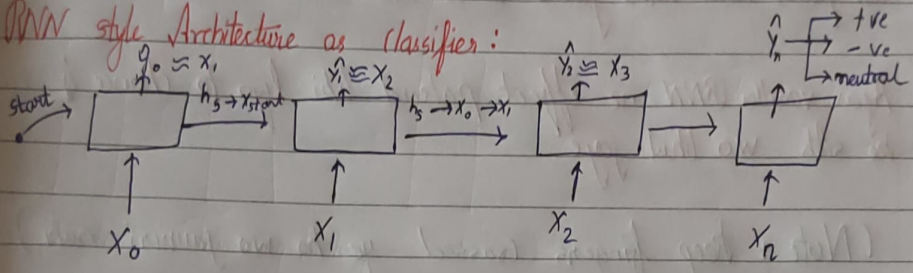
DATE

$\vec{\theta}$: Output filter : It decides kitna shadow rakhna h ya kitna shadow chota kar dena h. Wo dekhega ki noise dobara na aajaye

$$\vec{\theta}\begin{bmatrix} \\ \end{bmatrix} = \sigma\left[W_\theta \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix}\right]$$

$x''_t = \tanh(\vec{\theta} \odot C_t) \quad h''_t = \vec{\theta} \odot \tanh(C_t)$

→ RNN style Architecture as classifier :



first example :

Sentimental Analysis → Class → +ve, -ve, neutral

Hate Speech Analysis → Class → Yes, No

Opinion Analysis / Mining → Support, Oppose, neutral [Twitter]

The degree of some class is a regression problem.

When doing classification the modifications are:

- Removing y_0, y_1, y_2 part. i.e we are skipping $\hat{y} = W_y * h_3 \to x_0$ thy
- We just want the y in the last cell but we want it to predict classes instead of predicting the next word.

classmate

PAGE

DATE

- We don't look at all elements but actually there are certain prominent elements which will define the outcome for the classification task

[Certain vectors ka zyada contribution hoga snowball banane me]

Training ke waqt, non essential elements ka contribution kam hona hi chahiye ye hame chahiye

- We prefer other RNN structure rather than Vanilla RNN bcoz use pata nahi kitna jaane dena hai kitna nahi
- Agar vanilla RNN me effect pakad bhi liya to chance h time ke sath wo bhul jaye

(Note: Keep fingers crossed ye ho wo jaruri nahi)

=> Conditional LM

$\hat{P}(w_t | \leftarrow w_{t-1} \leftarrow w_1 \leftarrow w_{start}) = ?$ Plain LM

$\hat{P}(w_t | \leftarrow w_{t-1} \leftarrow w_1 \leftarrow w_{start}\ ;\ C)$ → Condition

Types of condition

i) should be present in context matrix / window [Extrinsic]

(Don't be confused with context as condition)

(You can set condition that it should come before or after something)

Eg: ⟨John → loves → (TC) ; New York⟩

There will be cases the condition supresses the probability

Eg ⟨John → loves → (TC) ; Gandhinagar⟩

less chance of occurance of John with TC or with any other word.

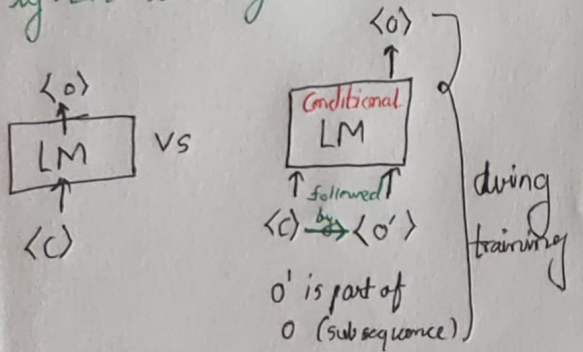
ii) The condition should occur somewhere inside the sequence [Intrinsic] ← Rare

- happens in certain generative LMs

iii) The condition is the input itself & the sequence is part of output & we have to complete it further

[Generating sequence of output given C input (prompt)]

Eg: text summarization



⟨C⟩ → Question

⟨O⟩ → A

⟨O'⟩ → subsequence of A

O' → can just be start (in case of minimum subsequence)

Eg of (ii) Translating sentence from 1st language to 2nd but you want to translate particular part of sentence

English → Hindi

Artificial Intelligence will take over the world -

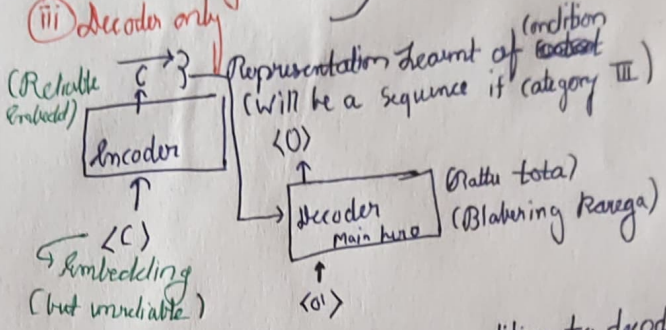
Artificial Intelligence duniya pe kabza kar lega.

C

Three types of LM

(i) Encoder - Decoder
(ii) Encoder only
(iii) Decoder only

→ How to make it Conditional LM

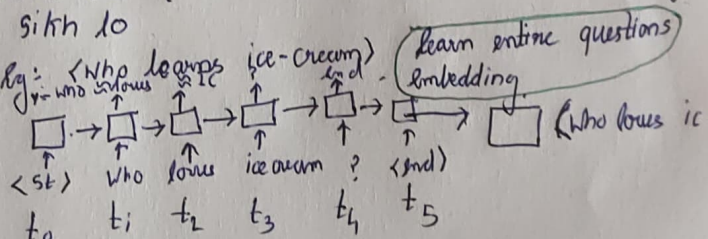


We are injecting question or condition to decoder through encoder.

In toughest case scenario the ⟨O'⟩ should be start (We want to train it perfect)

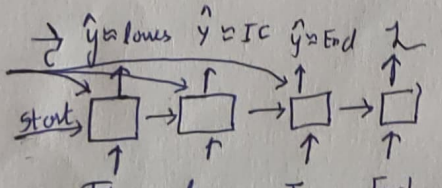
- Condition narrets how the decoder flavers the information

- We can use vanilla RNN as Encoder & decoder
  → first train (learn) Encoder. while ⟨C⟩ ka embedding sikh lo



You will be passing a lot of question learning W_x^c, W_y^c, W_h^c to Encoder

→ Once trained, you will plug it into decoder.
(It will be similar to encoder but with two inputs)



C → dhakka maar rahe h.

$h'_t = W_h * h_{t-1} + b_h$

$x'_t = W_x * x_t + b_x$

$h_t = \tanh\left((h'_{t-1} + \vec{C}) + \vec{x}'_t\right)$

Thaya dhakka