Prepared and Written By

RAZEEN AHMED

CSE-440

[NATURAL LANGUAGE PROCESSING]

HANDWRITTEN NOTE

Computer Science and Engineering
BRAC University

Github: github.com/razeen
LinkedIn: linkedin.com/in/razeenahmed

We will only focus on: Natural language (the language which grew with time)

→ It grew organically.

Ambiguity :-

Reasons :-

(i) Phonetics. (how we use sound)

(ii) Morphology.

(iii) Syntax → A single sentece may have multiple interpretation.

(iv) Semantics → A single word may have multiple meaning

VARSABILITY :-

(i) Same - meaning sentence written in different way.

Supervised Learning Agents:- The agent who learns using examples.

Data annotation:- The inputs which is given to computing system so that the PC can learn.
(The first step for any)

Data annotation must go through some processes.

Sentence segmentation:-

Learning based Model:-

challeges:-

(i) Non deterministic soln

Rule based Model

* There are some information we need to memorize. we can't generalize that.

challeges:-

(i) hard to cover all the rules.

(ii) maintenance

Tokenization:-

Each language requires its own tokenization.

Lematization:- A possible soln to understand the PC that these are same words.

(Never use stemmer)

Vectorization :- A word becomes multi dimensional vector space.

challenges :- contexts of similar words

NLP common libraries

(i) ~~yes~~ spacy ⊛

(ii) NLTK ⊛

(iii) Core NLP

(iv) Processors ⊛

06/11/24

AI first coined in 1950s.

⊛ Since advent of AI there was a tendency to manipulate language

→ propositional logics is the base of AI

~~#~~ Parts of Speech (POS) Tagging :-

→ 36 tags ——— are golden ratio.

Recurrent Neural Network

NLP's major challenge is Ambiguity.

(i). Closed class
↳ Categories have fixed set of words

(ii) Open class
↳ Categories have growing set of words.

Challenges:
→ One word can have different POS tag based on its use.

To maximum overall probability we will use global maximum (greedy Algorithm)

* If a word is noun or not noun we can find by its probability

* Identifying names:
Why important? →
Using the technique's of POS tagging we will find out names.

→ Name identifying is multi word task →
Rita Tasnim
split
"Rita" "Tasnim"
multiple person in brac university with this name.

Ambiguity
Washington DC
George washington
→ 1 word can have multiple names.

words are multidimensional.

.multiword → challenges → where to start & where to end.

✱ Co Reference Resolution:

when we face a pronoun which noun we are referring

✱ Parsing: → Allows machine to understand the grammatical structure of language.
→ analyze g. structure of a sentence to understand its components and the relationships between them

Challenge → Same sentences can be parsed in multiple ways.

→ Attachment Ambiguity

→ Coordination "

Session-2:

Supervised L. → entire learning is supervised by examples.

→ Identify the feature

→ map the features into a output.

Classifications 1st step is → "which feature we need"

	f_1	f_2	f_3	...	f_m	Y / N
1	Y	Y	N		N	
2	N	Y	Y		Y	
3						
⋮						
N						

and
WNE
WE

A technique

Bag of words:-

creating a vocabulary and creating column

example:-

(1) pretty good, not bad

(2) pretty bad, not good

	f_p	f_g	f_b	f_{not}	f_{comma}
(1)	1	1	1	1	1
(2)	1	1	1	1	1

Exercise

	f_{can}	f_{call}	f_{have}	f_I	f_{me}	f_{ill}	$f_!$	f_{won}	f_U	$f_{unsubscribe}$
(1)	0	1	0	1	0	1	0	0	0	0
(2)	1	0	1	0	0					
(3)										
(4)										

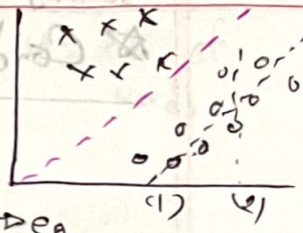
Machine classifiers

Find out the probability

$P(Y/O)$

Discriminative:

Creates an optimal boundary between two different class



eg. logistic r., SVM,

Generative:

Advantage of Discriminative:-

(i) It's very quick and inference type.

when its useful (discriminative)?

Generative classifiers:-

example:- Naive bayes

working idea :- -: falls
+: not falls

- ☆ Conditional probability
- ☆ Bayes theorem
- ☆ Prior probality
- ☆ Joint probability
  - The event which may occur together.

$P(+|O)$ $P(-|O)$

$P(+|f_1(O)\ f_2(O) \ldots f_m(O))$

$= \frac{P(f_1(O)\ f_2(O) \ldots f_m(O)|+)\,P(+)}{P(f_1(O)\ f_2(O) \ldots f_m(O))}$

$= P(f_1(O), f_2(O), f_m(O)|+)\,P(+)$

~~☆ Condi~~

$= P(f_1(O)|+)\ P(f_2(O)|+) \ldots P(f_m(O)|+)\ P(+)$

can calculate when they are independent of each other.

Use more training data ~~than the probability~~ to increase ~~decrease~~ the ~~probability~~ probability of getting unknown words | Test data must be completely unknown.

★ Conditional Independence:

$$P(A, B|C) = P(A|C)\,P(B|C)$$

$$= P(A_1 A_2 \ldots A_n | B)$$

$$= \prod P(f_i | B)$$

Classifiers

step → make vocab table. (Convert each e.g. to feature vector)

+s = 10

training data	$f_!$	$f_{I'll}$	f_I	f_{sorry}	f_U	f_{call}	f_{can}	f_{have}	f_{later}	f_{msg}
+ Sorry I'll call later	0	1	1	1	0	1	0	0	1	0
+ U can can call msg now	0	0	0	0	1	1	1	0	0	1
− U have call now!	2	0	0	0	1	0	1	0	0	0
~~Test~~ Test										
Sorry! U can not unsubscribe	1	0	0	1	1	0	1	0	0	0

~~P(+|0)~~ $P(+|0)$

$P(!|+) = \frac{0}{10} = 0$ | $P(!|+) * P(sorry|+) \ldots P(call|+)$

$= 0$

$= 0$ ← problem

Total words → 12

Add 1 normalization

Solve:-

$$P(!|+) = \frac{0+1}{10+M} = \frac{1}{10+12}$$

M = no. of words

← large constant

→ $P(f_1(0)\, f_2(0) \ldots f_{12}(0) \mid +)\; P(+)$

$= P(f_1(0)|+)\, P(f_2(0)|+) \ldots P(f_{12}(0)|+) \times \frac{2}{3}$ | $P(+) = \frac{2}{3}$ ← 2 (+)

← no. of exclamation present in (+) is 0 (check the 1st one)

$$= \frac{0+1}{10+12} \times \frac{12}{22} \times \frac{2}{22} \times \frac{2}{22} \times \frac{2}{3}$$

! sorry V car

$= 0.000023$

$P(-|0) = P(!|-)\, P(V|-)\, P(=)\, P(\text{sorry}|-)\, P(\text{car}|-)$

$= \frac{2+1}{7+12} \times \frac{1+1}{7+12} \times \frac{1}{3} \times \frac{1}{7+12} \times \frac{1}{7+12}$

$= \frac{1}{7+12} \times \frac{1}{7+12} \times \frac{1}{3}$

no. of words

f_vocab ← 12

* lexicon → set of relevant words which is related to our tasks.

Morphemes

NOTES

TOKENIZATION:- It is the process of breaking down text into smaller meaningful units.

Challenges :- (i) Ambiguity
(ii) Handling punctuation
(iii) Compound words. (Hyphemation)
(iv) Contractions

No, general solution

"Similar words look different"

Solution :-

(i) Stemming : chopping of suffixes & prefixes from words based on set of rules, without understanding context or meaning.
- Cut out common substrings.
- Strip words to a base form

(ii) Lemmatization : it reduces the word to their base form and considers parts of speech and meaning.

(iii) embeddings. Vector representation of words, phrases.
→ similar meaning word have similar vector representation.

→ Faster, inaccurate. Resulting base word may not be a real word in the language, causing loss of meaning

→ lemmatizers rely on hard built lexicon to ensure the base form is a real word. which will be meaningful and grammatically correct

→ slower, accurate. [words with multiple meaning]

[More complexity & processing time as it relies on dictionary or linguistic rules]

Errors in POS

POS TAGGING :-

The process of assigning a part of speech to each word in a sentence based on its context and definition. [Assigning grammatical categories to words]

- → Context of similar words
- → Requirement of high computational resources & memory.

She reads book

PROBLEM :-
One word can have diff pos tag, eg → I painted the room
He " room

she → pron
reads → verb
book → noun
room → adj or verb

NAME ENTITY RECOGNITION (NER)

Identify and classifies entities into predefined categories

Challenges :- (i) Ambiguity → apple (company & fruit)
Jordan (country & person) need context to be correctly classified.

Solution :- (i) sequence tagging.

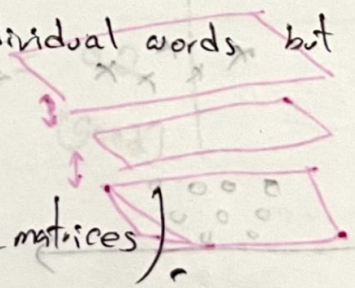
PARSING :-

understanding → sentences to determine → grammatical structure / syntactic

Challenges :
→ many sentences have multiple possible interpretation or structure

Bag of words :-

Simplifies text into collection of individual words but keeping track of their frequency.



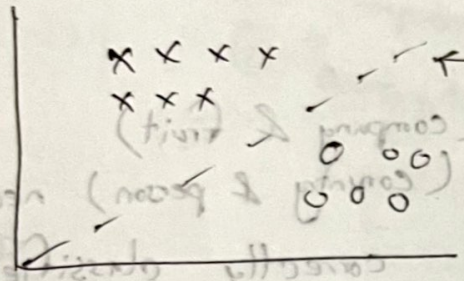
Problems :-
(i) One feature function per word (large, sparse matrices).
(ii) Completely ignores word order and context.

13/11/24

Sentence level representation does not work.

→ How to make ML Algorithm better?

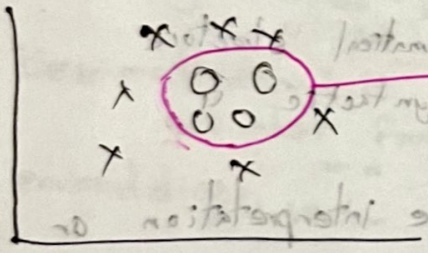
∴ we will use Discriminative classifiers.



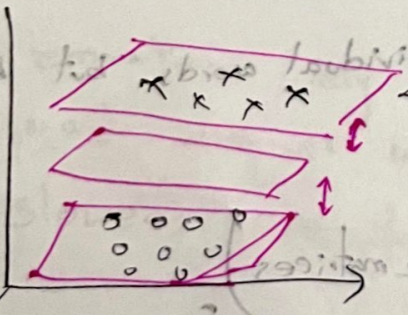
we have to identify this line.

1D in 2D

$y = mx + c$



↓ what if upward projection in 3D



Target → to create a hyperplane in a dimension to get a optimal solution

Loss :- → To identify the data wrongly | why needed?

→ The amount of error.

We will use "CROSS ENTROPY Loss".

Sigmoid Function :-

~~Sigmoid~~

$$\frac{1}{1+e^{-y}} = \frac{1}{(1+e^{-(mx+c)})}$$

↓ the more closer to 0 more better.

The more closer to '0' the more beffe (For '0' class)

→

Gain Function → $\hat{y}^{y}(1-\hat{y})^{(1-y)}$

→ $-\log(\hat{y}^{y}(1-\hat{y})^{(1-y)})$

→ $-y\log\hat{y} - (1-y)\log(1-\hat{y})$

(Binary Cross entropy loss function)

$x_m = 1$

$\hat{y}_{x_m} = 0.2$ (bad)

~~$x_m = 0$~~

$x_n = 0$

$\hat{y}_{x_n} = 0.2$ (Good)

$x_n = 0$

$\hat{y}_{x_n} = 0.9$

Every 10 example we will update!

↑
This 10 is batch.

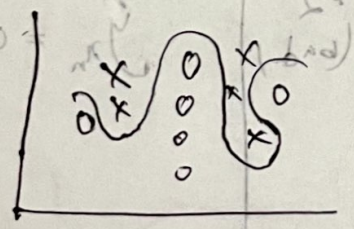
Full interation in one entire dataset is 1 epoch.

when we stop:-

→ At one point the graph will hit a plateau then we will stop.

we will redce the loss function (gradient descent & Boraboor)

until we hit plateau.



→ It will be a very complex solution. So, its not recommended.

GOAL:-

Minimum loss as simple as possible decision boundary

we want to have a simple solution to reach a optimal soln.

DATA SPLITTING:

- Training data ← train model
- validation / Evaluation data ← evaluate
- Test case

parameters :-
(i) Type : hyper parameters
(ii)

good / Not good

Change parameters

Change learning environment / updates hyper parameters.

to get the validation data :-

- Cross validation
  N fold validation.

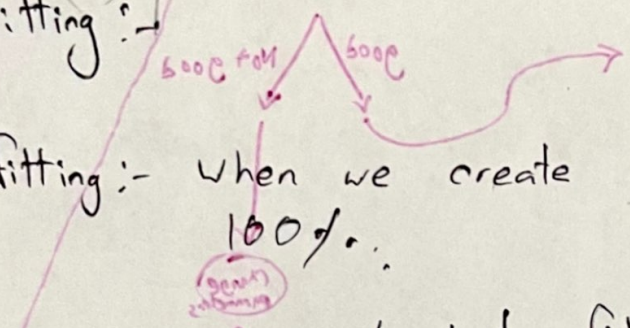
Try to split the data as early as possible.

strategy :-
- how many splits
- when to split

18/11/24

What is fit? | We want to fit our model to the training data.

Underfitting :-



Overfitting :- When we create something fit our data 100%..

☆Idea is to have to best fit not perfect fit.

development / evaluation / validation.

Solution :-

→ Nonlinear equation

→ Increase dimensions.

→ The problem should fit our solution or else it will be overfitting.

PROBLEM :-

☆Providing too much data with high amount of noise & outliers

$$\text{Accuracy} :- \frac{\text{no. of correct predictions}}{\text{All examples}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy is not a good metric when the goal is to discover something rare.

When accuracy score is not a good choice?

→ if the classes are inbalanced.
→ if we want to focus on one sigle classes.

Undersampling:-

problem- data are reducing

Over sampling:-

problem:- we are losing the real life information.

Precision:- we have to reduce the FP in case of

$$P_+ = \frac{TP}{TP+FP}$$

$$P_- = \frac{TN}{TN+FN}$$

2 cars are produced → 2 cars are flawless.

The amount of operations we are performing should be more precise.

High recall:-

$$R_+ = \frac{TP}{TP+FN} \qquad R_- = \frac{TN}{TN+FP}$$

Better Solution :-

Harmonic mean :- F_1 score $= \frac{2}{\frac{1}{P}+\frac{1}{R}} = \frac{2PR}{P+R}$

Which metric should be presented?

☆ Can we claim our model is better?

→ model performance have to be significantly better, to significant no. of times.

P value :- sensitivity value.

Hypothesis testing :-

Prog 1 → reject the null hypothesis

End of session-3

Dimension

→ If the root word are same.

→ Capture synonyms.

→ " antonyms

→ Words sense ↳ looks same, different meaning

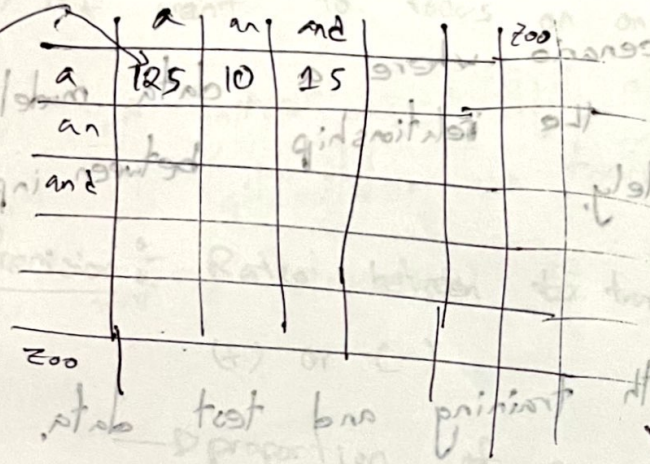
Sentence to convert to number is not good idea.

Distributional semantics:-

Term Term Co occurence semantics:-

$|V| \times |V|$ — matrix

vocab (unique words)



how many times both the words came together in a sentence or paragraph

Individually present in different sentace or in a dataset.

connectioned with bag of words

Underfitting:- When a model has not learned the patterns in the training data well and is unable to generalize well on the new (given) data.

☆ Poor generalization

→ when a model is too simple to capture data complexities.

→ It is a scenario where a data model is unable to capture the relationship between input and output accurately.

PROBLEM:-

(i) Poor performance in both training and test data.

Overfitting:- Overfitting occurs when a model learns the training data too well, perfectly ~~indeed~~ including noise and irrelevant ~~path~~ patterns rather than the underlying patterns/ generalization trends.

Problem:-

poor performance on unseen/new data.

☆ Excellent performance on training data, but poor performance on unseen test data

e.g. →

(1) Healthcare Diagnosis (Detecting life

(2) Search and rescue operation

lexicon :- list of words that have been annotated for a task

Performance metrics :-

Accuracy :- It measures the proption of the correctly predicted instances out of the total no. of instances.

$$A.S = \frac{TP+TN}{TP+FP+TN+FN}$$

Problem :-

(i) The distribution of classes or categories are inbalanced.
(ii) If want to focus on one single classes.

e.g. → Balanced dataset
→ detecting spam in text message.

Precision :- Ratio between true positive or (+) and predicted (+) or (-)

→ proportion of correctly predicted instances out of all instances predicted as positive.

$$P_+ = \frac{TP}{FP+TP}$$

$$P_- = \frac{TN}{FN+TN}$$

Problem :-

→ Not ideal for all scenario's
e.g. → cancer detection,
→ ignores false negatives.

(ii) Sentiment analysis e.g. —
(i) Spam detection
(ii) Targeted advertising

fraud detect, Spam spam classificati

Recall :- Ratio between true (+) or (-) and actual (+) or (-)

→ That measures the ability of a mode to correctly identify all relevent instances.

Problem :-

Un balanced focus
↳ does not ensure that the predicted positives are correct

→ not ideal for all scenario's
→ does not account how many incorrec (+) prediction it makes.

Scanned with CamScanner

F1 score:- Harmonic mean of precision and recall.

→ combines p. and r. score

Eg → weapon detection in airport.

$$\frac{2PR}{P+R}$$

Statistical Significance:

It is a measure used in hypothesis testing to determine whether the observed effect or relationship is likely to have occured by chance or represents a true underlying effect.

→ why need?

⇒ Avoid random errors

⇒ Decision making

⇒

P value:-

probability of obtaining the observed results if the null hypothesis is true.

test value ≤ p value (reject null hypothesis)

|V| x |V|

Vocabulary

	any	big	cat	dog
any	1	1	1	0
big	1	2	2	0
cat	1	2	3	1
dog	0	0	1	1

Individual Sentence → any big cat
→ big cat
→ cat dog cat.

	any	big	cat	dog
S1	1	1	1	0
S2	0	1	1	0
S3	0	0	2/1	1

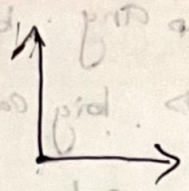
not how many times! but if there is present or not

transpose

	S_1	S_2	S_3
any	1	0	0
big	1	1	0
cat	1	1	1
dog	0	0	1

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

$$= \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 2 & 2 & 0 \\ 1 & 2 & 3 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$



⇒ They are not connected at all.

0	2	2	1
1	2	2	1
1	1	0	0

Cosine similarity:- It is better function to calculate the seperation between two matrix.

$$\frac{\vec{v} \cdot \vec{w}}{\sqrt{\sum v_i^2} \cdot \sqrt{\sum w_i^2}}$$

To check whether our vector length

Cosine are normalized. In NLP, does not matter.

In NLP, because its scalar multiplication

How many words are important

★ $v = [0, 1, 0, 1]$ → cantonym → $[0, -1, 0, -1]$

$v = [1, 0, 1, 0]$

$w = [3, 0, 3, 0]$

$[-5, \ldots]$

have to normalize with the big vector

We want something more semantically make sense.

TF-IDF vectorizing:-

= Assigns important values of the word in a certain context.

e.g. →

messi → football

putin → politics

Probate. →

raw count is not a smart idea.

→ Gives more informative vector of a certain sentence.

(term frequency)

TF → calc.

to get a small val

$$tf(w,d) = \log(\Delta + f(w,d))$$

raw frequency

	any	big	cat	dog
S1				
S2				
S3				

IDF multiplier :-

$$\log\left(\frac{N}{f(w, D)}\right)$$

ekta goto kom context e present hoy tar mane oi particular context e ei word ta important

example :-

$N = 1000$ sentence, $D = \{ d_0, d_1 \ldots\ldots d_{999} \}$

$f(\text{sorry}, d_{783}) = 2$ | $f(\text{and}, d_{783}) = 10$

$f(\text{sorry}, D) = 100$ | $f(\text{and}, D) = 900$

For sorry :

~~log(.~~

$tf(\text{sorry}, d_{783}) = \log(1+2)$

$= \log 3$

$= 0.477$

$IDF(\text{sorry}, D) = \log\left(\frac{1000}{100}\right)$

$= 1$ ~~$= 0.477$~~

~~$\log 3 = 0.477$~~

Multiplying $= 1 \times 0.477 = 0.477$

For and :

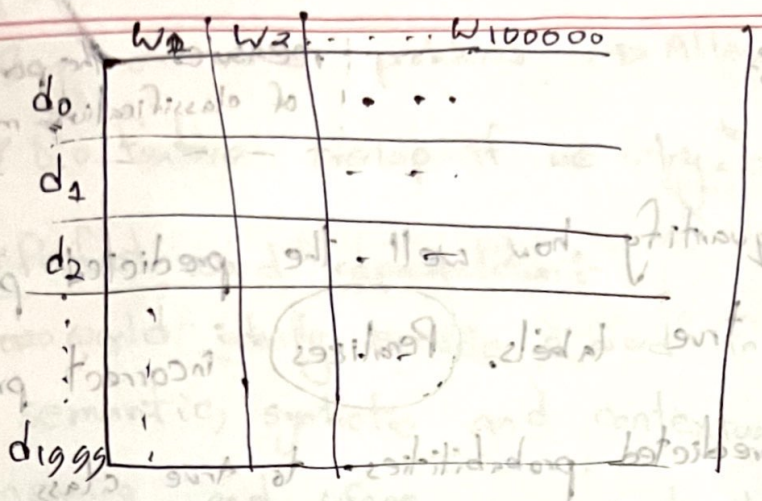
$tf(\text{and}, d_{783}) = \log 11$

$IDF(\text{and}, D) = \log\left(\frac{1000}{900}\right)$

$= \log \frac{10}{9}$

Multiplying $= \log 11 \times \log \frac{10}{9}$

$= 0.047$



PMI → Book
→ Mid

weighted form of term-term co occurrence.

Q. Pattern

straight

Three q. → theory question
→ analytical question → mathematical
→ complex q. → real life situation

find out which is better in this particular scenario.

CROSS VALIDATION

It is a technique in ML used to evaluate the performance of a model and prevent overfitting.

It splits the available datasets into multiple subsets to train and test the model iteratively.

Goal:- Not overly reliant on specific portion of the training set.

CROSS ENTROPY Loss:

measures the performance of classification model

* loss function

→ It is used to quantify how well the predicted probabilities match the the true labels. Penalizes incorrect predictions by comparing predicted probabilities to true class labels.

→ C.E.L. is a mathematical loss function used to evaluate the performance of classification models.

It measures how well the predicted probability distribution aligns with the true labels by quantifying the uncertainty of the predictions.

Formula:

$$L = -[y \log(\hat{y}) + (1-y) \cdot \log(1-\hat{y})]$$

y → true label

$\hat{y}$ → predicted probability

* we will reduce the loss function until we hit a plateau

BATCH

Subset of the training data used during one iteration of training a model.

EPOCH:

One full iteration through the entire training dataset.

There is

Bag of words problems ⟶ Alternate method.

* No feature overlap if we try to calculate similarity.

Perfect word representation:-

would ideally encode a word in such a way that all the semantic, syntactic and contextual information relevent to its meaning and usage are captured.

How we can do that?

(i) Share lemmas ; e.g ⟶ mouse/mice,

(ii) different word senses ; e.g. ⟶ pc mouse, vs pet mouse,

(iii) Synonyms ; e.g. ⟶ couch/sofa, car/automobile river bank vs finance bank.

(iv) antonyms ; e.g. ⟶ long/short, ligh/dark

(v) word similarity ; e.g. ⟶ cup/coffee, car/road

(vi) word relatedness. ; e.g. ⟶ cat/dog, doctor/nurse.

(vii) word arousal ; e.g ⟶ excited & angry ⟶ high arousal

relaxed & depressed ⟶ low arousal

* Words can be represented as vectors, each entry in the vector represents a dimension.

Term Term Co occurence matrix:

It is a respregentation used in NLP to capture the frequency with which a pair of terms (words) occur together within a defined context such as within a single document, a sentence, ±n words of each other.

It is:

$$X = |V| \times |V|$$

no. of words in vocabulary

Build steps:-

(i) build binary BOW for the sentences

(ii) Transpose it

(iii) Multiply it with the original matrix

Cosine similarity:

$$\cos(v,w) = \frac{\bar{v} \cdot \bar{w}}{\sqrt{\sum v_i^2}\ \sqrt{\sum w_i^2}}$$

PROBLEMS:-

(i) very sparse

(ii) does not carry any contextual info

iii) " " represent how important a word is in a sentence.

$$TF\text{-}IDF_{wt} = tf(w,d) \times idf(w,d)$$

TF - IDF : Assign important values of the word

It is a statistical measure used in NLP to evaluate how important a word is to a document within a collection of documents

#TF :

* TF → occurs many times in specific context.
  → Measures how frequently a term appears in a context.

* High TF vector → informative
* High IDF → uninformative

~~IDF → Does~~ If a term appears frequently in a document, its likely to be important in that document

FORMULA :-

$$tf(w,d) = \log(1 + f(w,d))$$

↑ raw frequency ~~of the~~ count

IDF :

→ Does not occur in every context.

→ Measures how important or unique a term is across the entire document.

→ Rare words carry more information

→ common words are less informative
e.g. → the, and, is,

Formula :-

$$idf(w,d) = \log\left(\frac{N}{f(w,D)}\right)$$

N ← no. of sentence

f(w,D) ← no. of this word in the document

EXAMPLES:-

stemming:- equivalence ⟶ equival

organization ⟶ organ

lemmatization :-

leaves ⟶ leaf

falling ⟶ fall

embedding:- ⟶

dog = [0.5, 0.4, 0.2]

dogs = [0.5 0.4 0.2]

WORD REPRESENTATION : WORD VECTORS

1) Skip gram (word2vec) | 2) Glove

(i) Sparse vectors

(ii) Dense vectors

WORD EMBEDDINGS :-

It transform words or phrases from their raw textual form into continuous vector representation in a high dimensional space.

These representation capture semantic relationships between words.

two types :-

(i) CBOW :- (Continuous Bag of Words)

Predict the target word given the neighbouring words.

(ii) Skip-Gram :-

Predict the neighbouring words given the target word.

→ Skip Gram Classifier :-

i→ its not hardware intensive.

→ its a simple technique.

→

GloVe :- Designed to generate dense vector representation of words

PROBLEMS :

(i) They are symmentically dense.

(ii) How can we combine the word vectors so that we can classify the sentence. — S_1

(iii) The word vectors cannot capture the contextual information in a particular situation. bank — money bank — S_2
bank — sea shore

(iv) Word2Vec cannot deal with unknown words

Solution :-

(S1) → we can calculate the centroid vector

problem :- If we use antonyms both of them will banish each other in to a particular dimension.

(S1) [better solution] → RNN can convert a word vector to a sentence level representation.

(S2) → using language models.

~~Language Modelling~~ :-

LANGUAGE MODELLING :-

→ To assign probabilities to sequence of words or generate plausible word sequence.

→ It is a statistical model that learns the likelihood of a sequence of words occuring in a particular language.

why do we need it?

(i) Understanding context & meaning. (iii) Text generation.
(ii) sentiment & emotion analysis (iv) Speech recognition.

Sematic properties of embeddings:

(i) Different types of similarity ~~or~~ / ~~words with similar no.~~ sematic similarity

→ Based on the context window words similarity changes.

⇒ Hogwarts — harry, harmione, potter

(ii) Analogy / relational similarity.

→ encode meaningful relationships between words.

ex — man is to king
woman is to Queen.

(iii) Historical context.

→ The meaning of a certain word may vary based on what time period we are using it.

Imp for exam :-

- relational similarity vector given
- prove that here relational similarity vector isn't working

There is no engineering technique which can remove the bias from a model

SEQUENCE LEARNING.

05/12/24

It is very important to capture the sequence

How? "Using POS tagging"

Fully	furnished	condo	in	the	beautiful	catalina	foothills
ADV	ADJ	NOUN	PREP	DET	ADJ	PROP	

Have to identify that ADJ comes befor noun

have to identify that this forming an connection with ADJ

Challenge → ambiguity

Solution → dictionary → but there is a problem

WSJ

1 tag	45%
2+ tag	55%

we will assign tags for each 1 tag word

we might get only 70% accuracy at best

Solution 2 → We will create a big table for the ambiguous words and store the most commonly used tag.

This simple solution will provide a 92% accuracy

MAXIMUM ENTROPY HIDDEN MARKOV MODEL :-

$ used for sequence modding predict sequences of hidden state

probability → For a certain % event to occur at the present only depends on what happened before"

For word prediction we will look at the past immediate words

If we want predict the present words tags → look at past words → tag, lemma, casing info

we will also look at late words?

STEPS OF ME MM :-

(i) If we want to predict the tag of a word we will make feature vector of the past words. We will name it $x_1, x_2, x_3 \ldots x_n$

(using casing info, suffix, prefix)

(ii) then use $y = m_1 x_1 + m_2 x_2 \ldots m_n x_n$ → linear regression classifier.

Revised MEMM ~~set~~ steps:

(i) Make feature of every words of the sentence.
Vector
↳ (Using casing info, suffix, prefix)

(ii) Then train the model.

(iii) Input a new sentence now predict the tags of the next words.

we will predict bidirectionally. ~~For~~ From left → right, right → left then compare both prediction and create a final prediction.

[Because, there is a less chance to make the same error from both direction]

Solution

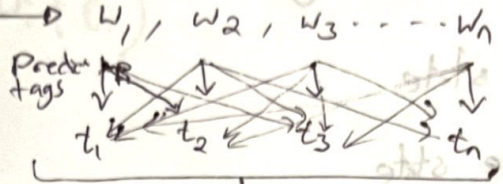
PROBLEM:

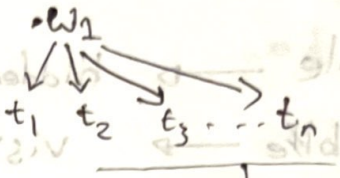
If we incorrectly predict the previous word then it will heavily effect the prediction of the next words

HIDDEN MARKOV MODEL:-

words → $w_1, w_2, w_3 \dots w_n$

Predict tags → $t_1, t_2, t_3 \dots t_n$





PROBLEM →

It will create exponential time complexity

we will calculate the probability of every assignments

↓

the probality which will provide ~~a~~ max out that will be our final tag assignment

we will predict the ~~tags~~ probability of every 1 tag assignments (possible)

↓

Select the maximum tag probabil

↓

Final tag selected

approximation :-

Ds are independant each other

2nd approximation :-

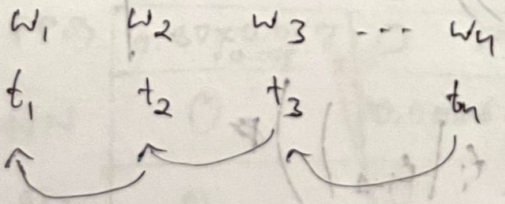
☆ WORDS Depend only on their POS tags.

Not their surrounding POS tags

Calculate the probality of w_1 given t_1, w_2 given t_2, w_n given t_n

3rd approximation :-

A tag is dependent only on the prev tag



HIDDEN MARKOV MODEL

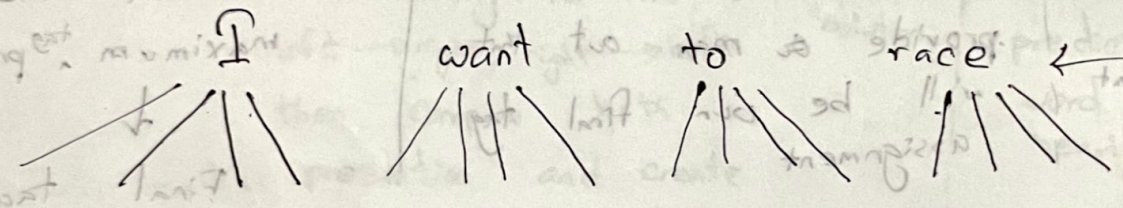
☆ To infer hidden state of things from visible things.

Not observable ⟶ hidden state

observable ⟶ visible state

visible ⟶ words

Not visible ⟶ tags



every single word will have the probability of every single tag

↳ Goal ⟶ Find the globally best possible assignment.

↳ (we won't use local maxima)

$$\hat{t}_1 = \text{AM}(P(t_1^N \mid w_1^N))$$

$$= \text{AM}\left(P(w_1^N \mid t_1^N)\, P(t_1^N)\right)$$

$$\simeq \text{AM} \prod_{i=1}^{N} \left(P(w_i \mid t_i) \prod_{i=2}^{N} P(t_i \mid t_{i-1})\right)$$

The words are independent.

$$\hat{t}_{1:n} = \operatorname{argmax}\left(\underbrace{P(w_1^n \mid t_1^n)}_{\text{Words likelihood P.}} \underbrace{P(t_1 \mid t_{1-1})}_{\text{Tag transition P.}} \right)$$

1st

For all possible w_i's & t_i's we create a table.

$P(w_i \mid t_i)$ → word likelihood table

TRANSMISSION PART

Word

2nd

Trans Create a transition probability table.

INFERENCE PART [VITERBI ALGORITHM]

I want to race

Initialization

$P(t \mid \langle s \rangle) = P(I \mid t)\, P(t \mid \langle s \rangle)$

$= P(I \mid NN)\ P(NN \mid s)$

$P(I \mid VB)\ P(VB \mid s)$

$P(I \mid PPSS)\ P(PPSS \mid s)$

$P(I \mid TO)\ P(TO \mid s)$

we will use argmax in the values that the value which is maximum we will bring the arrow from there

Find the maximum

	I	want	to	race
PPSS	0.37 × 0.067 = 0.025	0		
NN	0	0.00006		
TO	0	0		
VB	0 × 0.019 = 0	0.00093		

prev × trans × words likely

Optimal value for

NN/PPSS

0.025 × 0.0025

$P(I|want) = P \quad AM \quad P(want|t) \quad P(t|i-1)$

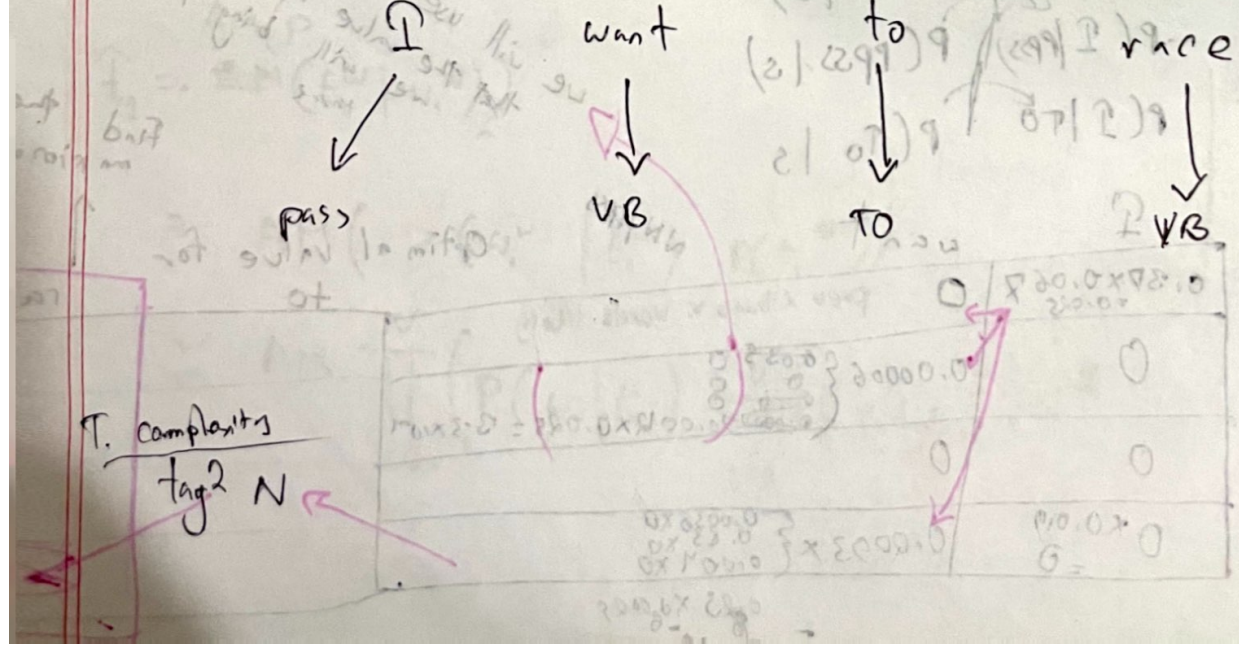
$= P(Want|VB)$ $\quad P(want|TO)$ ~~$P(want$~~

$P(want|PPSS)$ $\quad P(want|NN)$

$P(VB\|VB)$	$P(TO\|VB)$
$P(VB\|TO)$	$P(TO\|TO)$
$P(VB\|NN)$	$P(TO\|NN)$
$P(VB\|PPSS)$	$P(TO\|PPSS)$

Quiz

Speech language processing (Jurafsky). $\longrightarrow$ 8.2, 8.4.



T. complexity

$tag^2 N$

Disadvantage:-

(i) The number gets smaller & smaller as it moves forward.

(ii) without unknown word zero will not propagate forward.

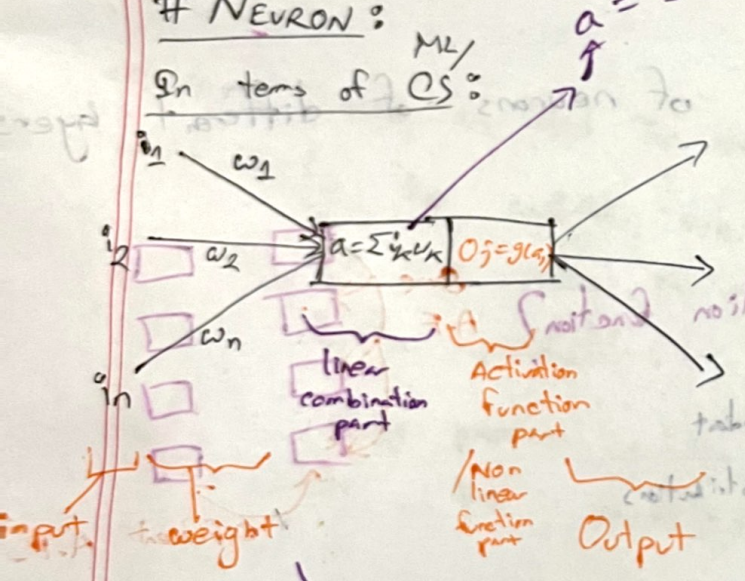
(iii) Hard to add features in the model.

NEURAL NETWORKS:

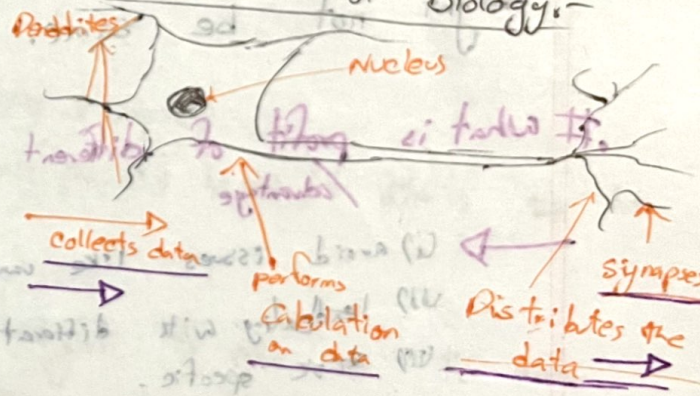
12/12/24

NEURON:

In terms of CS: ML/



In terms of Biology:-



$O_j = g(a)$

$\hookrightarrow O_j = \max(0, a_j)$

TENSOR:

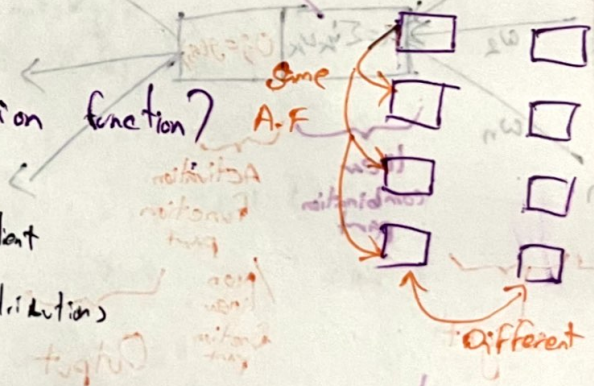
It is an N-dimesional / Multidimensional array of data

The Neural Network :-

Fully connected neural network :- when every single neuron stays connected with every neuron of the next layer.

☆ ~~Every~~ The activation function of every neuron of the same layer will be same.

☆ The activation function of neurons of different layers might not be same.



What is profit / advantage of different activation function?

→ (i) avoid issues like vanishing gradient
(ii) dealing with different data distributions
(iii) task specific.

Why do we need Activation function?

⇒ A.F is a type of gate. It works as a gate in the neural network which decides if the inputs/values should pass forward or not.

?

TYPE of A.F. → Linear A.F.
→ Non Linear A.F.

★ Why do we need non linear A.F.? → enable the model to learn and represent complex relationships in data.

→ Our real world is too complex to model using linear functions. It is often impossible to classify real world data using linear function. [Real world problem often have complex non-linear relationships between features and labels]

→ The combination of two linear function is another linear function. We can't introduce any non linearity in the linear function.

★ Where would we find the weights?

⇒ Initially the weights are randomly assigned

⇒ As the time passes, the weights are updated and it starts to make better predictions

The training process updates the weight to minimize the error between the predicted outputs and the actual targets.

set the ML engineers

HYPERPARAMETERS:-

(i) 1st HP → how many layers would there be in the NN

depends on the number of features ⇒ NI ↑ As it goes forward to next layers neuron the neuron should be decreased

(ii) In every layers how many neurons do we need.

(iii) In every layer which activation function should we use.

(iv) size of the layer

(i) Number of neurons in each lay
(ii) Number " layer
(iii) Activation function.
(iv) The optimizer we need t use.

RELU:

Advantage:-

(i) It perfectly acts as a gate.
(ii) It is ~~really~~ easy to implement

PROBLEM

(i) Badluck (There might be a seenario that every ~~neuro~~ ReLU of the neurons gets off, then the neuron cannot possibly pass the data forward to the next layer neurons. As a result, the network will stop learning.

(ii) Leaving capacity.
→ Non differentiable at $x=0$ / breaking point

why differentiation needed?
→ For optimization.

Solution of Problems:-

(i) Even if all the ~~gates~~ ReLU's gets ~~sat~~ off. Some small amount of value will still pass through

(ii) We use GeLU to make the entire function differentiable by making the elbow like a small curve.

SIGMOID :

→ switches between 0 to 1. why

→ we use this as a output activation function

↓

provides a probabily value by

~~TAN~~ TAN H :-

→ Works very good in the intermediate layer as a activation function.

→ switches between −1 to 1

Q.1

Q.2

Q.3 → Q.5

OPTIMIZER :

It is such type of algorithm whose task is to make the learnable parameters to learn using the optimum weight. It tries to reduce the loss

---

27/11/24

QUES-3

P(water | VB) P(VB | VB) × prev.

Only these two needed

If the situation given is we don't need to use prev. Because prev is included in tags (The words which are previously situated are given

REGULARIZER :- set of strategies to reduce the generalization error

"1 of the hyperparameter"

Engineering Hacks

Mathematical regularization

(i) [DROP OUT]

⇒ Suppose, we have 100 neurons in a single layer. In every epoch we randomly turn off some neurons. in that scenario, so, we

(ii) Early stopping

⇒ when validation reaches a flat surface we stop there.

(stops when performance gets saturated)

ADAM Optimizer (Advantage)

(i) Easy to implement (iii) Needs no manual tuning
(ii) Little memory requirements (iv) suited for large data.

(iv) Data augmentation.

=> Introduce new training data with the more variation.

Bias variancy trade

Bias ↑ variance ↓ (underfitting)
Bias ↓ variance ↑ (overfitting)

Mathematical Regularization :-

→ L1 Regularization :-

weight → modulus
add all weights | $L1 = |w_1| + |w_2| \ldots |w_n|$

(selects the best features

→ L2 Regularization :

[Prioritizes the best features.]

$L2 = w_1^2 + w_2^2 \ldots + w_n^2$

Bias → partiality without any observation

Bias ↑ observation

what less observation Bias is the highes

Variance

To reduce overfit introduce a little bit of bias.

LEARNING RATE :-

(one of the hyperparameter)

It is the lambda value of the neural

→ how quickly a network updates its parameters.

Epochs :-

One complete pass through the entire training dataset during the training p of a model.

BATCH

~~BATCH~~ SIZE :- (mini batch)

It matters in terms of time. | subset of training data.

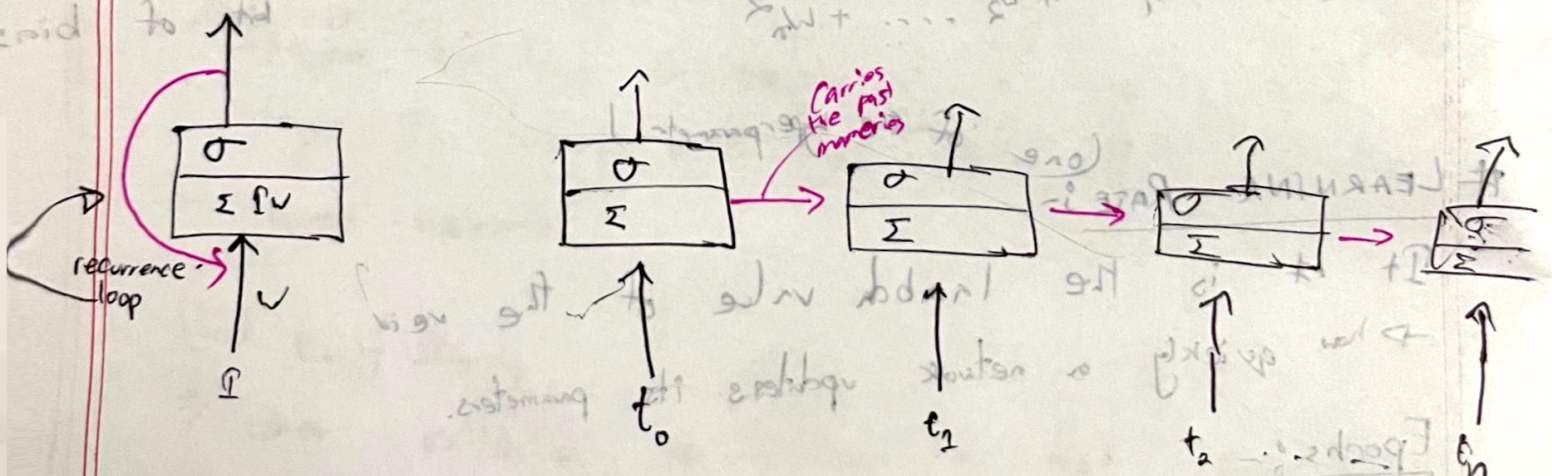
→ By seeing how →

RNN ← sequence learning classification.

18/12/24

we have to handle sequential data.

I AM A STUDENT



Every element ... own neuron.

Encoder architecture

Decoder architecture: One to many → chatgpt

Encoder - decoder architecture: eg. summariser (many to many)

EXAMPLE ~~MATH~~ RNN formul

$$h_t = \tanh(W x_t + U h_{t-1})$$

$$= act(U h_{t-1} \ldots)$$

HYPERPARAMETERS complex task type

(i) ~~what~~ will be the (non)Activation function.

hidden state output.

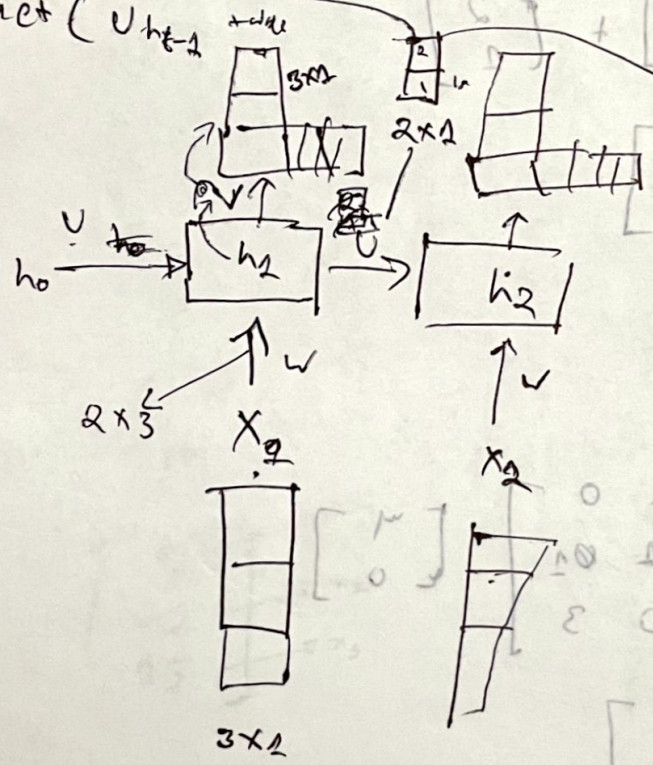
Example :-

tags = 3

ADJ

DET

NN



~~U is~~

U is a square matrix

U → 2×2

Vector ↑

weight matrix

h

hidden layer ~~dimensionality~~ dimensionality

Sigmoid Function = $\frac{1}{1+e^{-x}}$

$$x_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$h_2 = U \times h_{t-1} + W \times x_1$$

$$= \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 2 & 0 & -1 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2 \\ -2 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

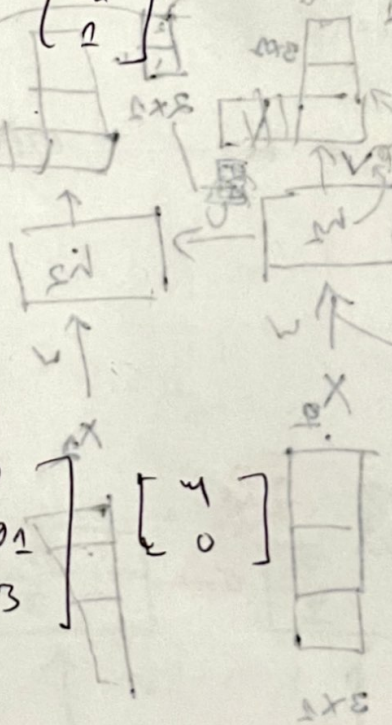
$$= \begin{bmatrix} 4 \\ -1 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$h_1 = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

Now,

$$Vh_1 = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 4 \\ -4 \\ 0 \end{bmatrix}$$



Bi directional RNN Math ?

$$O^{(t)} = \frac{e^{xi}}{\sum e^{xi}} \left[Vh^{(t)} \right]$$

$$h_2 = Ux + Wx_2$$

$$= \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 2 & 0 & -1 \\ 2 & -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 8 \\ 4 \end{bmatrix} + \begin{bmatrix} 2 & -1 \\ \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$h_2 = \begin{bmatrix} 7 \\ 5 \end{bmatrix}$$

$$Vh_2 = \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 7 \\ 5 \end{bmatrix}$$

$$= \begin{matrix} 7 \\ -7+5 \\ 15 \end{matrix}$$

$$Vh_2 = \begin{bmatrix} 7 \\ -2 \\ 15 \end{bmatrix} \begin{matrix} \leftarrow x_1 \\ \leftarrow x_2 \\ \leftarrow x_3 \end{matrix}$$

$$\frac{e^{15}}{e^{7} + e^{-2} + e^{15}} = 0.9996$$

$$\frac{e^{-2}}{e^{7} + e^{-2} + e^{15}} = 4.14 \times 10^{-7}$$

$$\text{Sigmoid} = \frac{e^{7}}{e^{7} + e^{-2} + e^{15}}$$

$$= 3.35 \times 10^{-4}$$

$$O^{(2)} \Rightarrow \begin{bmatrix} \text{ind}=0 & 3.35 \times 10^{-4} \\ \text{ind}=1 & \\ \text{ind}=2 & 0.9996 \\ & 4.14 \times 10^{-7} \end{bmatrix} \xrightarrow{\text{argmax}} \text{class } 1$$

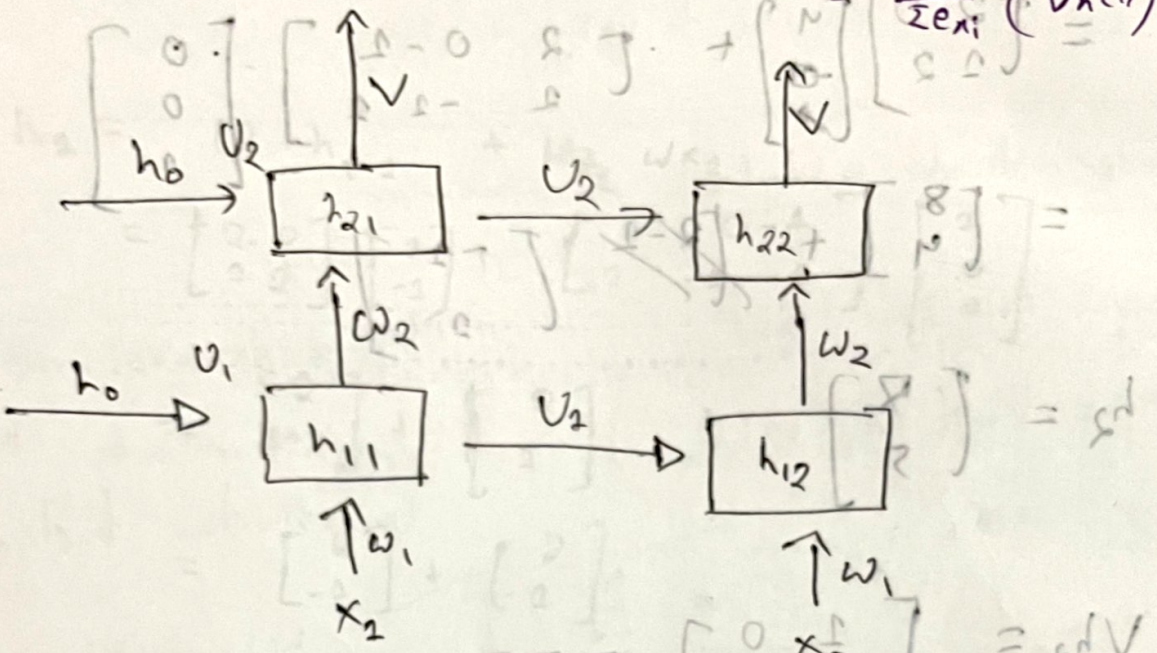
No. of hidden layer → depends on the size of input.

Double layer RNN

$$h^{(t)} = \tanh\left(U h_{t-1} + W X^{(t)}\right)$$

$$O^{(t)} = \text{Softmax}\left(V h^{(t)}\right)$$

$$= \frac{e^{x_i}}{\sum e^{x_i}}\left(V h^{(t)}\right)$$



Challenges:-

(i) ✗ It is unidirectional → solve → make it bi directional

(ii)

(iii) Problem of vanishing gradient → Solve → LSTM

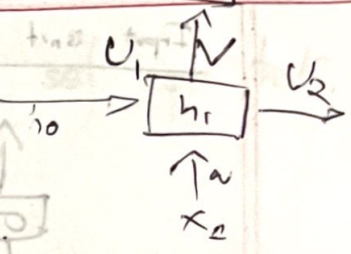
Drawbacks:-

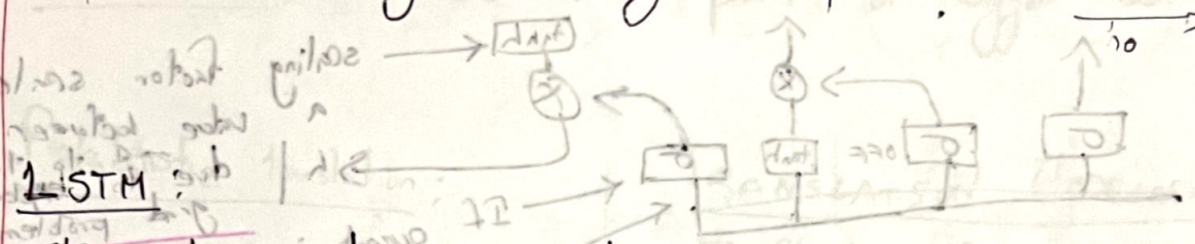
→ Can only see the past, not the future.

→ Must forget the same amount of history at each time ste

23/012/24

How to identify some thing as important?





LSTM:

- New three gates are introduced
- Element wise operations

→ The shape has to be exactly the same to perform those operations

⇒ If the information is important enough we will keep it in the persistent memory.

Forget gate:

If any info is important, the forget gate empties / free up some memory. To clear up the memory.

If it is very important info. if needed it will fully empty up the memory.

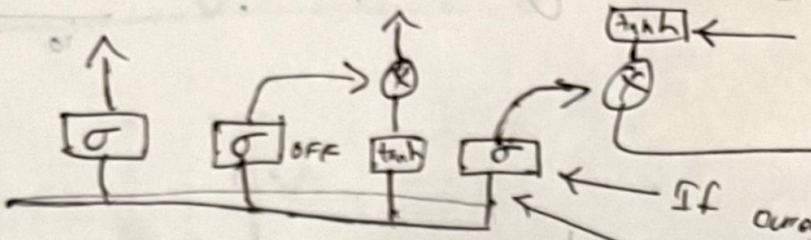
Mathematical functio... that works lik... a gate.

Input gate:-

- how much we will open or close the gate
- " " " " pass the data.

Forget gate ≠ input gate are reciprocal.

Input isn't important



scaling factor scales a value between -1 to $+1$ due to exploding gradient problem

If current input "yes" fully open gate
"no" "close"

Output gate

GRU & LSTM

(i) There are no two channels
(ii) There is a scaling factor

Redundancy of LSTM

→ forget g. & Inp g.

Gated RNN problem can't solve

(i) Back propagation through time
(ii) RNN's are not parallelizable.

30/12/24

TRANSLATION :- Is a part of bigger set of task.

Probabilistic translation :-

Language Modelling :

IBM MODEL 1

Solution :-

(i) Auto regression

→ Why do we not use Dictionary set.

TRANSLATION MODELING :-

checks if correctly identifying word level translation

★ →

Disadvantage :-

(i) factorial complexity

(ii) extremely inefficient process

(iii) Ambiguity words handling.

IBM MODEL 1

Expectation maximization algol algorithm. (EM)

(i) Because a specific words meaning may change

Example

(i) b c [b c → x y]

translate (ii)

(iii) x y [b → y]

Step-1

$P_1(x \mid b) = \frac{1}{2}$

$P_2(y \mid b) = \frac{1}{2}$

$P_3(x \mid c) = \frac{1}{2}$

$P\begin{pmatrix} b & c \\ | & | \\ x & y \end{pmatrix} = \frac{1}{2}$

$P($ b c / y $)$

a b c

x y z

→ 6 alignments per permutation.

step-2

Normalizing

$P\left(\begin{matrix} b \\ | \\ x \end{matrix} \quad \begin{matrix} c \\ | \\ x \end{matrix}\right) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4} = \frac{1}{4} \times 2 = \frac{1}{2}$

$P\left(\begin{matrix} b \\ | \\ x \end{matrix} \quad \begin{matrix} s \\ \times \\ y \end{matrix}\right) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4} = \frac{1}{4} \times 2 = \frac{1}{2}$

$P\left(\begin{matrix} b \\ | \\ y \end{matrix}\right) = \frac{1}{2} \times 2 = 1$

Step-3

$P(x|b) = \frac{1}{2}$ → Normalizing → $\frac{1}{1}$

$P(y|b) = \frac{1}{2} + 1 = \frac{3}{2}$ ← probability exceeds → $\frac{3}{4}$

$P(x|a) = \frac{1}{2}$

$P(y|a) = \frac{1}{2}$

Iteration 2

step-1

$P\left(\begin{matrix} b \\ | \\ x \end{matrix} \quad \begin{matrix} c \\ | \\ x \end{matrix}\right) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$ → divide by 2 → $\frac{1}{8}$

$P\left(\begin{matrix} b \\ \times \\ y \end{matrix} \quad c\right) = \frac{3}{4} \times \frac{1}{2} = \frac{3}{8}$

$P\left(\begin{matrix} b \\ | \\ y \end{matrix}\right) = \frac{3}{4} \times \frac{1}{3} = \frac{1}{4}$

Step-2

$P(x|b) = \frac{2}{7}$ ($P(x|e) = \frac{3}{4}$

$P(y|b) = \frac{2}{8}$ $P(y|e) = \frac{2}{4}$

Challenge! -

(i) If translates to multiple words, so it won't be able to translate. Can't translate more than equal words leads

(ii)

(iii)

→ ⇒ will work in low resource language.

Solution to work better :-

→ Religious books (no dictionary present for low resource language)

CCM models ke iden is Auto regression

Probabilistic translation model → Neural machine translation

LSTM

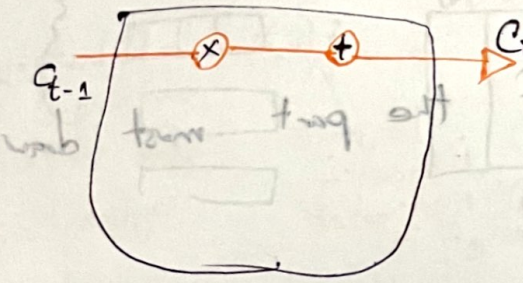
A technique to know which we should remember and which we should forget

[LONG SHORT TERM MEMORY NETWORKS]

→ Designed to avoid Long term dependency problem.

Characteristics :- Remembering information for long period of time

LSTM STRUCTURE :-



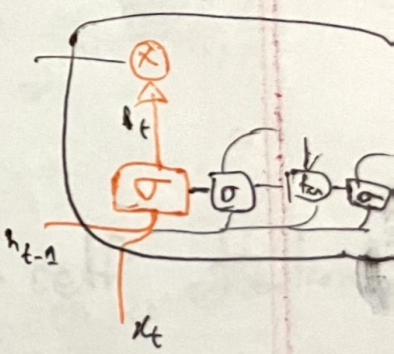
C_t => cell state → kind of conveyor belt.

have the ability to remove or add info to the cell state.

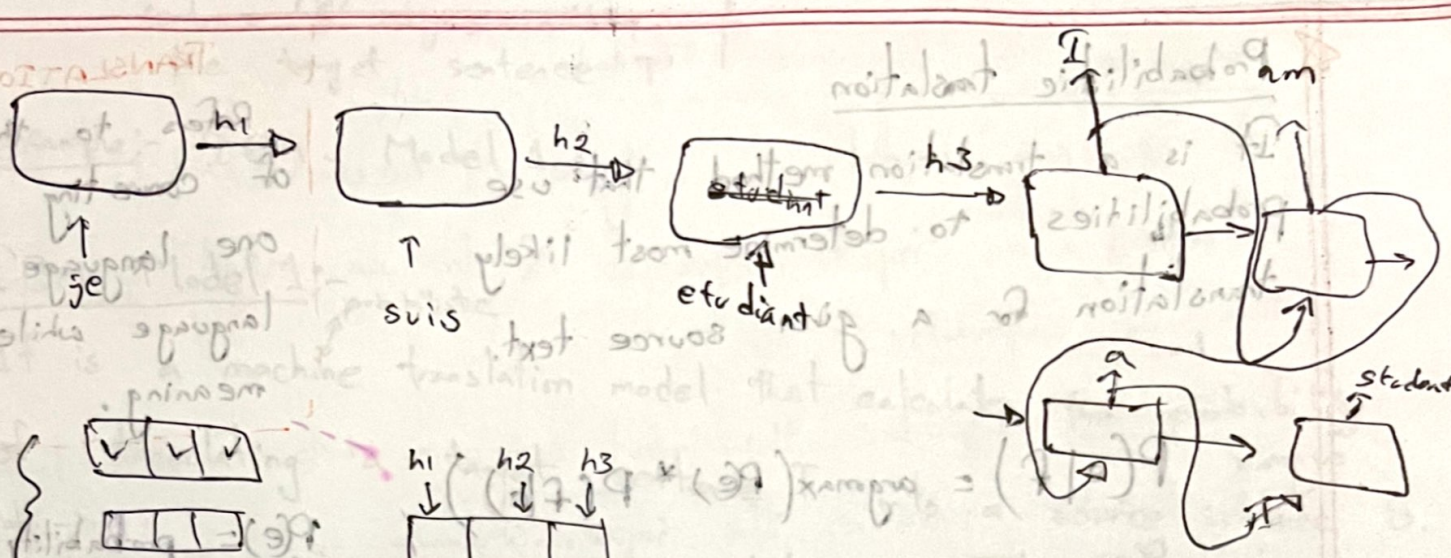
Forget gate layer [sigmoid layer] :-

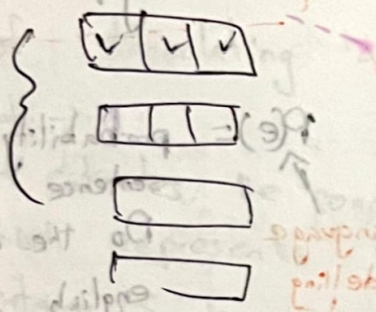
→ what info we are going to throw away from the cell state.

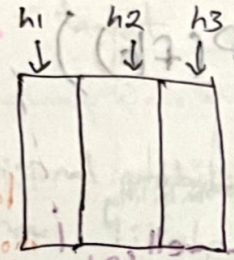
→ output a number between 0 & 1

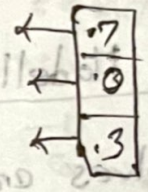


06/01/24









Attention weight matrix

TRANSFORMER:-

① will look at all possible combinations. ⟶ self attention.

TRANSLATION

~~380~~ 350 ~~395~~ → 750

Probabilistic translation

It is a translation method that use probabilities to determine most likely translation for a given source text.

$$\text{argmax}\ P(e|f) = \text{argmax}(P(e) * P(f|e))$$

P(e) = language model

P(f|e) = Translation model probability

measures how well the source sentence 'f' aligns with the target sentence 'e'

Language Modelling:

It involves creating models that assign probabilities to sequence of words in a language.

→ A language model estimates P(e).

It is the probability that a sentence e is an english sentence.

TRANSLATION MODELLING:

It is a probabilistic framework that defines the relationship between a source language S and a target language T.

→ It estimates the conditional probability P(S|T) or P(T|S), representing how well a given sentence target sentence corresponds to a source sentence.

TRANSLATION:

Refers to the process of converting text from one language to another language while preserving meaning.

P(e) = probability of english sentence

Do the words follow english order?

language modelling estimates

In short:-

LM estimates probability of a sequence of words.

technique's are :-

(i) n-gram LM

(ii) RNN's

a sentence is a sequence of words.

$P(T|S) = P(S|T)\ P(T)$

T = target sentence
S = source "

P(T) → language model, measures

P(S|T) → measures how well the source sentence 'S' aligns with the target sentence 'T'

example:- IBM - Model 1

IBM Model 1:-

It is a probabilistic machine translation model that calculates the probability of translating a target sentence T into a source sentence S.

Goals →

(i) It estimates the conditional probability P(S|T), which represents how likely a source sentence S is as a translation of a target sentence T.

[we can use this P for
→ word to word translation
→ Alignment of words between sentences

→ consider all possible word alignments

→ combine the word translation probabilities

(ii) Align each source word s_i to a target word t_j

$$P(f|e) = \sum_{a} \prod_{(e_i, f_j \in a)} P(f_j|e_i)$$

$P(f_j|e_i)$ = for each e_i english word the probability of it being translated as french word f_j

Attention allowed decoder all hidden states of encoder → benefit! → info loss → addressed by attention

Sequence to Sequence Model

A seq2seq model is a type of neural architecture designed to map an input sequence X to an output sequence Y.

two main components :-

Encoder :- → Encodes the input sequence into a fixed-length representation

→ Encoder processes the input sequence word by word and encodes it into a fixed dimensional context vector.

⇒ A context vector that summarizes the entire input sequence.

→ At the end of the input sentence, the encoder produces a context vector that captures the sentence's meaning.

Encoder is mainly a RNN, LSTM, GRU

Decoder :- Decodes the encoded representation into the output sequence

→ The decoder generates the output sequence one word at a time using the summarized context vector

Decoder is another RNN, LSTM, GRU.

PROBLEM :-

The encoder compresses the entire input sequence into fixed length context vector. For long input sequences, this ~~single may to fe~~ can lead to information loss / may fail to fully capture all the important info.

Solve :- AI Attention mechanism.

Review.

HMM

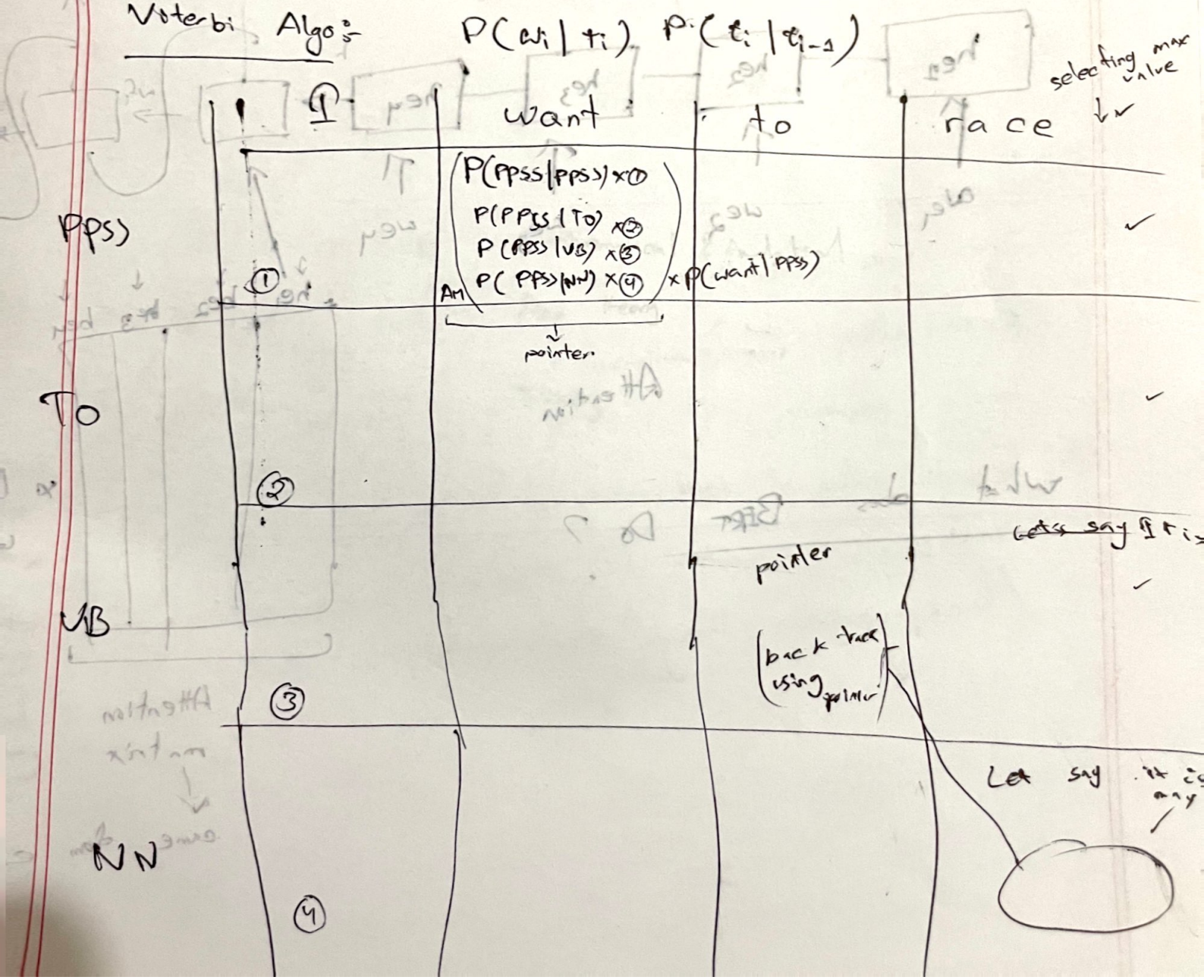
$t_i = \text{Arg max}\left(P(t_i^n \mid w_1^n)\right)$ → why viterbi → $T(n)$ → huge → huge

$P(w_i^n \mid t_i^n)\, P(t_i^n)$

$P(w_i \mid t_i)$ ↓ bails down to $P(t_i \mid t_{i-1})$

viterbi → we can forward to dynamic programming

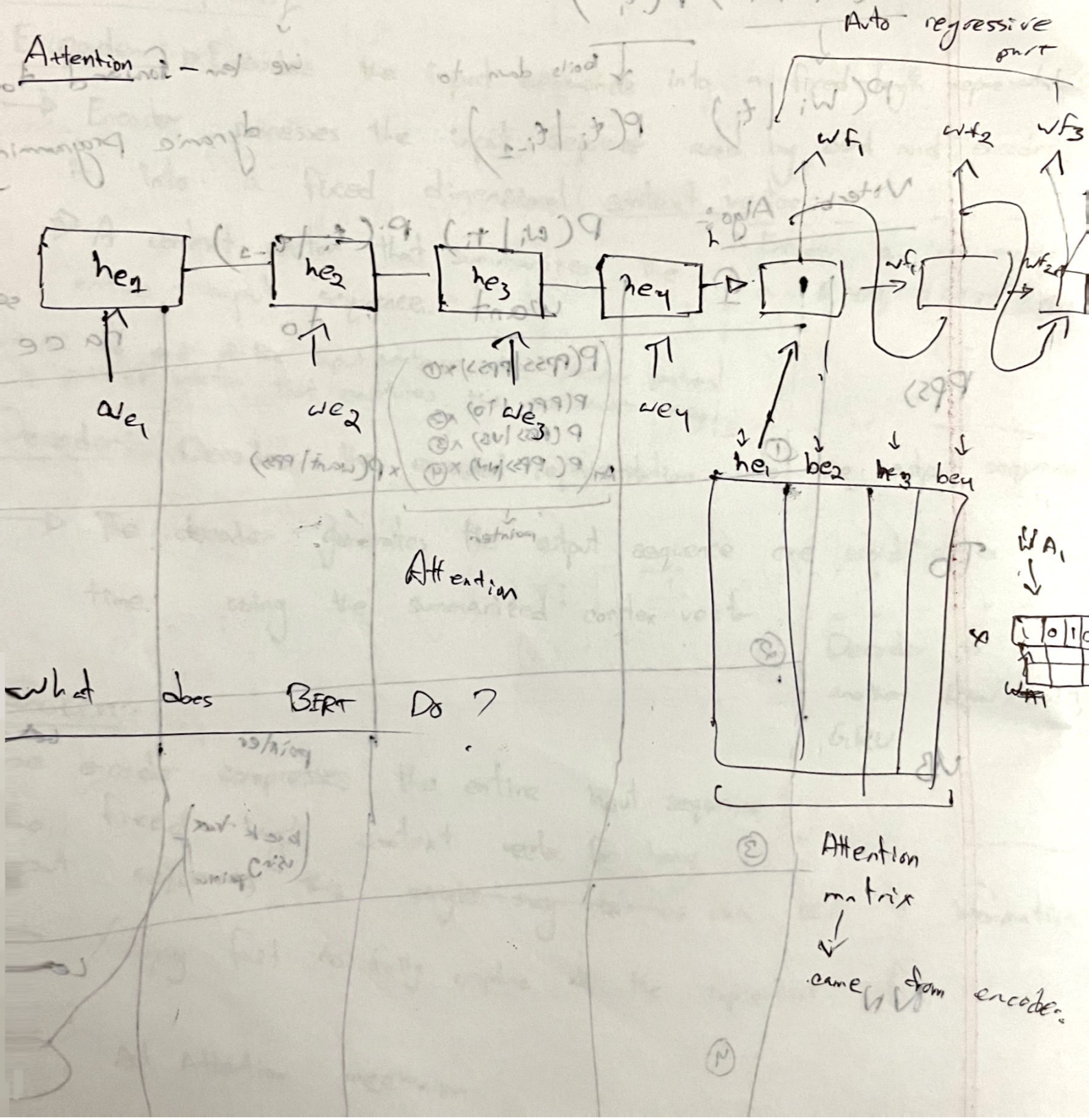
Viterbi Algo:- $P(w_i \mid t_i)$ $P(t_i \mid t_{i-1})$



Difference between HMM & MEMM.

LSTM

Attention :-



What does BERT Do ?

Seq 2 Seq using Attention:- Instead of passing one single context vector.

→ Attention allows the decoder to access all hidden states of the encoder, rather than relying on single fixed length context vector.

→ At each decoding step, the model computes a weighted of the encoder's hidden states, and focuses on most relevent part of

All the hidden states is passed in a attention matrix to the decoder.

The information bottleneck for long sentences solved

PROBLEMS:-

(i) RNN works sequentially, the sequential dependency prevents parallelization. Results slower training

(ii) RNN suffers from vanishing gradient issues when processing long sequences

(iii) Sequential nature of RNN makes it difficult to leverage modern GPU.

(iv) RNN struggle to retain info from very distant word at in long sequence / Backpropagation through time is still an issue.

Solve:-

(i) Transformers use self attention to process all input words in parallel.

(ii) Using attention mechanism and feed forward layers, transformes avoid recurrence eliminate vanishing gradient problem.

(iii) By processing all words simultaneously, transformers can take full advantage of GPU.

(iv) self attention calculates relationship between all words in a sentence, regardless of distance.

Why need GPU:-

(i) It is very good at matrix multiplication.

(ii) More computing unit → better thread parallelization.

(iii) Matrix multiplication requires more computational operations, GPU's provide more memory for computation operations

(iv) have faster access to RAM

TRANSFORMERS

Transformers removes the reliance or Recurrent structures and uses only attention mechanism to process sequences. Introduced self attention as its core operation.

→ Transformers have → n → no. of encoders
→ " " decoders
} big size cuz

→ Calculates the effects of one word on another word (multiple times)

Characteristics:-

(i) Size

(ii) Self attention → calculate the effect of one word on all of its surrounding word, every word

(iii) ~~Rotional~~ Positional encoding → doesn't work sequentially → so how to capture sequence?
↳ provides the position of the word
↳ using this position we find the connection/relation between every words.

(iv) Byte pair encoding → to limit the ~~vocabulal~~ vocabulary size
↳ transformer use byte pair encoding. → we use this instead of word tokenization
↳ reduces the need for large vocabulary
↳ handles rare and unseen words effectively

(v) Multi headed attention. → each ~~hea~~ attention head learns different relationships, ~~partic~~ ~~rel~~ [Calculates attention from different angles

* self attention allows each word to understand its relationship to every other word.

TRAINING A TRANSFORMER:-

- Not enough hardware
- Not enough time / Long training times
- Large computational requirements

Training two parts

- Pretraining: those who have enough resources, and time train a transformer to learn the language representation
- fine tuning: Try to train the pretrained transformer about different tasks.

BERT:-

It is a pre trained transformer based, developed by Google. Designed for various NLP tasks.

Characteristics:-

(i) Bidirectional context understanding: BERT processes text in both direction simultaneously.

(ii) Pretraining & Fine tuning:- BERT is pretrained. can be fine tuned on specific NLP tasks

1. SEMI-SUPERVISED :- training a language.

→ Training on large amounts of text.

→ trained on a certain task enables it to grasp patterns in language.

2. Supervised :-

→ Training on a specific task with a labeled dataset

→ Pretrained on 2 unsupervised task

(i) Masked LM → Replaces a token with a [MASK]

is a task where portion of the input tokens is randomly masked

80% of the time, with a random token

10% of the time and does not replace

of the time.

(ii) Next sentence prediction → learns to understand relations between pair of sentences.

LSTM

Introduces a memory cell and other gates to selectively retain, forget or update information.

Addresses :-

1. Vanishing gradient problem.
2. RNNs struggle to retain info from ~~long~~ earlier time steps of long sequences. [Difficulty in capturing Long Term Dep
3. Lack of effective memory mechanism.

PROBLEMS :-

1. Computationally expensive
2. Processes sequentially ; can't ~~by~~ be parallelized,
3. Vanishing gradient problem.

GRU

Simplified version of LSTM. merges forget g. & input g. into a single gate. faster to train.

Addresses - [LSTM have multiple gate]

(i) Simplifies gates by using only update & reset gate.

(ii) ~~Has~~ much better at using long distance info

(iii) faster training.

PROBLEM

(i) Can't solve backpropagation through time

(ii) Not parallelizable.