Natural Language Processing

* **Probabilistic language models**
  - → conditional probability
  - → The Chain Rule

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

  - → **Markov Assumption**: reduced context window
  - → start symbol <S> and stop symbol </S>
* Unigram, Bigram, Trigram
  - ↳ $P(w_1) \times P(w_2) \times \cdots \times P(w_n)$: $P(w_1) = \text{count}(w_1)/\text{Total words}$
  - Bi: $P(w_2|w_1) \times \cdots \times P(w_n|w_{n-1})$: $P(w_2|w_1) = \text{count}(w_1 w_2)/\text{count}(w_1)$
  - Tri: $P(w_3|w_1w_2) \times \cdots \times P(w_n|w_{n-2}w_{n-1})$: $P(w_3|w_1w_2) = \text{count}(w_1w_2w_3)/\text{count}(w_1w_2)$
* we count stop symbol as a word in our dictionary.

Example

$$P(\text{I am}) = \frac{\text{count}(\text{I am})}{\text{count}(\text{I})} = \frac{2}{3}$$

$$P(\text{am Sam}) = \frac{\text{count}(\text{am Sam})}{\text{count}(\text{am})} = \frac{1}{2}$$

* Issues with probabilistic models:
  - → **zero probability**: unseen data leading to zero overall
  - → **underflow**: small prob leading to near zero prob
* underflow: take logs of prob multiplications (solution)
* zero prob: smoothing is the solution
* **The Shannon Visualization method**
  - → next word prediction method
* The **intuition of smoothing** is to maintain the true distribution by giving weight which is borrowed from seen data to unseen data.
* **Laplace smoothing**: pretend we saw each word one more time than we did.

Example

Train set:
- <s> I ate an apple today </s>
- <s> I at a banana </s>
- <s> strawberry is sweet </s>

test set:
- <s> I ate an orange today </s>

$$\text{Laplace} = \frac{\text{count}(\text{word}) + 1}{N + V}$$

$N = 15$: total words

$V = 11$: unique vocab

so for word "I": $= \frac{\text{count}(I) + 1}{N + V} = \frac{2+1}{15+11}$

Backoff , Interpolation.

* **Linear Interpolation**:

$$P(w_n | w_{n-1} w_{n-2}) = \lambda_1 P(w_n | w_{n-1} w_{n-2}) + \lambda_2 P(w_n | w_{n-1}) + \lambda_3 P(w_n) \quad \therefore \sum \lambda_i = 1$$

* Set lambdas using held-out corpus
  → Training data , Held out data , best data

* unknown word '<unk>'

* Stupid backoff : doesn't make a distribution

<s> I saw van </s>
<s> saw a cat </s>
<s> he ate an orange </s>
} training

<s> I saw an orange </s> } test.

$\lambda_1 = 0.7$
$\lambda_2 = 0.3$

Bigrams with linear interpolation

$P(<s> I) = \lambda_1 P(<s> I) + \lambda_2 P(I)$

$P(I \text{ saw}) = \lambda_1 P(I \text{ saw}) + \lambda_2 P(\text{saw})$

$P(\text{saw an}) = \lambda_1 P(\text{saw an}) + \lambda_2 P(\text{an})$

$P(\text{an orange}) = \lambda_1 P(\text{an orange}) + \lambda_2 (\text{orange})$

↑ bigrams ↑ unigrams

* **Intrinsic , Extrinsic Evaluation**

* **Perplexity**

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \cdots w_n)}} \quad // N$$

$$= 2^{-1/n \sum \lg P(w_i)}$$

* Text preprocessing

* Text classification
→ Aspect based sentiment analysis
* Classification Methods:
→ Hand-coded rules
→ supervised machine learning
* Naive Bayes Classification
→ Negation dealing.
→ Naive Bayes works like unigram, words are independent
→ Bag of words representation ← prior prob of class

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$ where 'd' is the document and 'c' is the class

$$P(Pos|d) = P(d|Pos)\,P(Pos) \,/\, P(d)$$
$$P(Neg|d) = P(d|Neg)\,P(Neg) \,/\, P(d)$$

* P(d) from denominator can be removed in cases where the documents are same.

* multinomial naive bayes independence assumptions

$$P(x_1, x_2, \cdots, x_n \mid C)$$
$$= P(x_1|C) \cdot P(x_2|C) \cdots P(x_n|C)$$

maximum likelihood estimate = $P(class) = \frac{document(C=c_i)}{N_{doc}}$

$$p(word|class) = \frac{\text{Count of word in class}}{\text{count of all words in class}}$$

$P(c) = \frac{3}{4}$ =>

$P(chinese|c) = \frac{5}{8}$ = $P(chinese|j) = \frac{1}{3}$

$P(Beijing|c) = \frac{1}{8}$

$p(Macao|c) = \frac{1}{8}$ $P(Shanghai) = \frac{1}{8}$

without smoothing

$$P(d|c) = \frac{3}{4}\left(\frac{5}{8} \times \frac{5}{8} \times \frac{5}{8} \times \frac{0}{8} \times \frac{0}{8}\right) = 0$$

$$P(d|j) = \frac{1}{4}\left(\frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3}\right) = 0.001$$

with smoothing (laplace).

$$P(d|c) = \frac{3}{4}\left(\left(\frac{5+1}{8+6}\right)^3 \times \frac{0+1}{8+6} \times \frac{0+1}{8+6}\right) = 0.000301$$

$$P(d|j) = \frac{1}{4}\left(\left(\frac{1+1}{3+6}\right)^3 \times \left(\frac{1+1}{3+6}\right) \times \left(\frac{1+1}{3+6}\right)\right) = 0.000148$$

* Binarized Naive Bayes / Boolean
  → prioritizes the presence or absence of words more than the count of the words.
  → the data is changed such that there are no duplicates in each document.

Text Classification Evaluation

* Accuracy = $\frac{TP + TN}{TP + TN + FP + FN}$

precision = $\frac{TP}{TP + FP}$ — out of all that the model predicted true, how many were actually true?

recall = $\frac{TP}{TP + FN}$ — out of all that were actually true how many did the model predict correctly?

ground truth / predicted

	gold pos	gold neg
system pos	true pos	false pos
system neg	false neg	true neg.

Confusion matrix

$$A = \frac{25+20}{25+10+45+20}$$

Example

Accuracy = 0.45

C1:

TP	FP
25	45
10 (FN)	20 (TN)

$$Prec = \frac{25}{25+45} = 0.357$$

$$Rec = \frac{25}{25+10} = 0.714$$

C2:

10	30
25	35

$$P = \frac{10}{10+30} = 0.25$$

$$R = \frac{10}{10+25} = 0.2857$$

$$Accuracy = \frac{10+35}{10+30+25+35} = 0.45$$

* **F-score:** $\frac{2PR}{P+R}$ ∴ P = precision, R = recall

↳ harmonic mean of precision and recall

Q: Why do we use harmonic mean instead of arithematic mean?

A: In harmonic mean, if one value is extremely small, it has a huge impact on the result which helps in identifying a heavily biased model.

Macro and Micro Average

Example:

UK: $P = \frac{95}{95+10} \Rightarrow \frac{95}{105}$

$R = \frac{95}{95+1+13+1} = \frac{95}{110}$

for a 6×6 confusion matrix.

Macroaveraging:

Compute the precision and recall of all classes and then take the average, all classes don't have same weightage because the number or size of class will affect the weightage of class in total averaging.

Example

	# of obj	Precision
C1:	1000	0.1
C2:	30	0.8
C3:	500	0.01
C4:	100	0.04

MacroAvg: $P = \frac{0.1 + 0.8 + 0.01 + 0.04}{4}$

$P = 0.2375.$

* in micro, a pooled confusion matrix is formed

pooled: (↙ for example on slides.)

268	99	
99	635	

$= \frac{268}{268+99} = 0.73$ ↖ micro averaging.

* micro-averaging assigns equal weights to each "object" not each class.

K-Fold validation

* divide the validation dataset in 'k' and introduced each in every iteration for validation.

* vectors of words, dimensions are vocabulary

* Embeddings.
  - → Term-document matrix (word x doc)
  - → Term-context matrix (word x word)

	apple	orange	to	and	of	cat	dog	~~and~~ the
the	0	2	2	3	2	1	2	0
dog	0	0	0	2	2	1	0	2

window = 2.

* euclidean, cosine.

* **Euclidean Distance**

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

* **Cosine Similarity**

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$

$\cos(apr, digital) = 0+0+0 / \quad = 0$

$\cos(apr, info) = 1+0+0 / \sqrt{1^2}\sqrt{1^2+2^2} = 0.33$

$\cos(dig, info) = 0+6+2 / \sqrt{1^2+2^2}\sqrt{1^2+6^2+1^2} = 0.58$

* Raw frequency is a bad representation as it does not take into consideration the difference in the length of the documents.
  - so, we **normalize** our vectors
  - → length normalization → divide word count by doc length
  - → $\log_{10}$ normalization → $1 + \log_{10}$ (word count)

* **IDF** : inverse document frequency

IDF = total number of documents / (number of document frequency (w.r.t word))

→ log transformed $\log_{10}\left(\frac{N}{df_w}\right)$

TF - IDF = TF × IDF

Example:

	good	cat	food
d_1	60	0	100
d_2	100	50	0
d_3	0	20	50
d_4	0	1	1

doc-frequency of words	
good	100,000
cat	100
food	500

~~TF IDF = (60) $\log_{10}$ 1 × $\log$ 4~~

$TFIDF_{d_1} = 1 + \log_{10}(60) \times \left(1 + \log_{10}\frac{10^6}{100,000}\right)$

$= 4.556 + 0 + \left[1 + \log_{10}(100) \times \frac{1 + \log_{10}\frac{10^6}{10}}{}\right]$

$= 4.556 + 13 = 18.556$

* IF-IDF vector with context window.

Nural Language Model

→ Sampling
  → top K sampling.

* window based neural language model

Cross entropy:

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

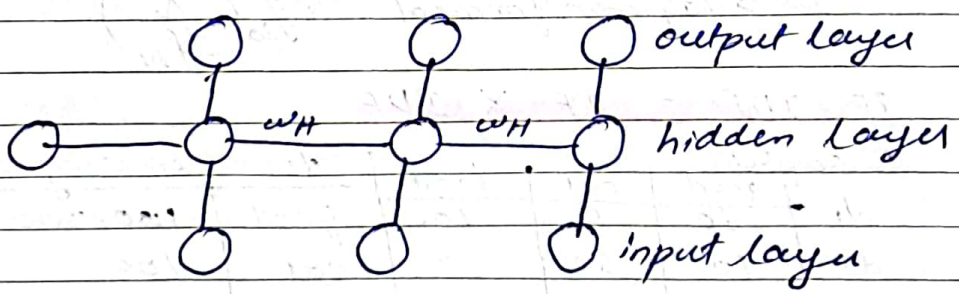
draw backs of neural language model:

→ slow training on more parameters.

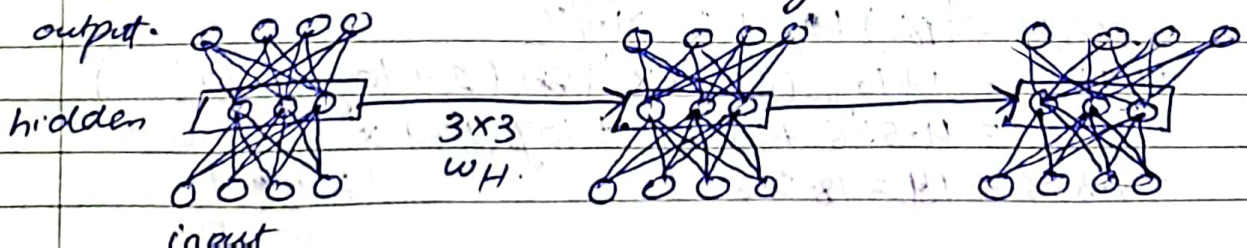
~~→ no ability to~~

→ no symmetry in how the inputs are processed i.e the nearby neighbours & sequence of words

Recurrent Neural Network



* In RNNs, the hidden layers are connected



$$y_3 = w_y h_3$$

$$h_3 = w_x (x_3) + w_h (h_2)$$
$$h_2 = w_x (x_2) + w_h (h_1)$$
$$h_1 = w_x (x_1) + w_h (h_0)$$

→ this shows how the first word x_1 would contribute in the last output y_3.

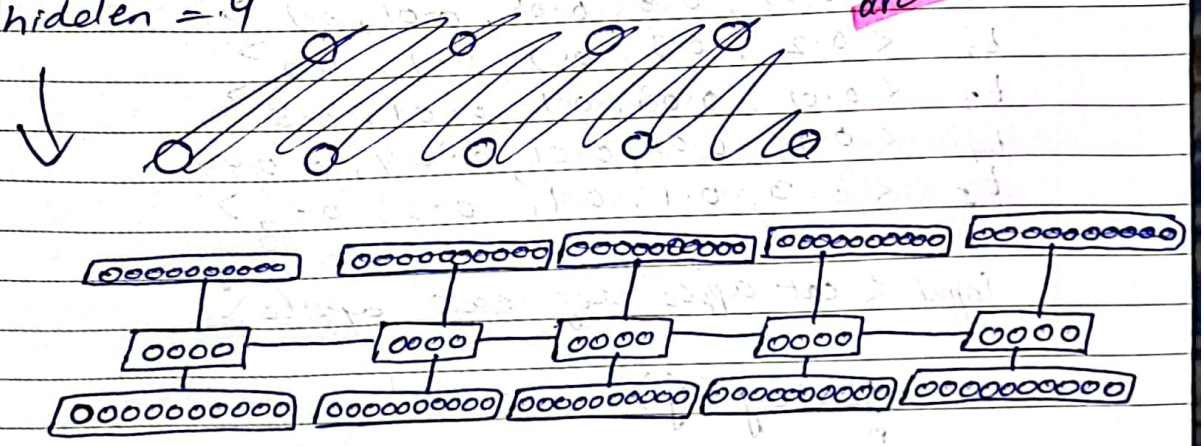
→ vanishing gradient problem

vocab = 10

input word sequence = 5

hidden = 4

draw the RNN architecture.



advantages over neural network:

→ gives more weight to closeby content words and lesser to far away neighbours

→ number of input words doesn't affect the number of weights thus doesn't slow down training

disadvantages:

→ vanishing gradient problem

cross-entropy

$$= \underbrace{y \log \hat{y}}_{y=1} + \underbrace{(1-y) \log (1-\hat{y})}_{y=0}$$

* cross-entropy measures how similar or dissimilar the ground truth & the predicted distributions.

<dog, cat, lion, apple, eat>.

t_1 <0.04, 0.8, 0.05, 0.01, 0.1>
t_2 <0.2, 0.3, 0.2, 0.1, 0.2>
t_3 <0.01, 0.08, 0.1, 0.01, 0.8>
t_4 <0.9, 0.01, 0.01, 0.04, 0.04>
t_5 <0.3, 0.1, 0.1, 0.2, 0.3>

Input < cat apple dog eat apple>

$$-\frac{1}{n} \sum \log (\hat{y})$$

$$-\frac{1}{5}\left[\log(0.8) + \log(0.1) + \log(0.01) + \log(0.04) + \log(0.2)\right]$$

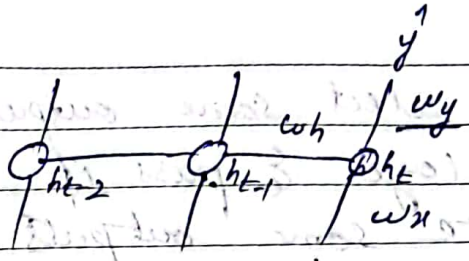
gradient descent:

Chain rule: $\frac{d(J)}{d(u)} = \frac{d(J)}{d(v)} \times \frac{d(v)}{d(u)}$

$J(w) = -\log \hat{y}$

$\frac{\partial J(w)}{\partial (w_2}$ *

$$\frac{d(J)}{d(x_1)} = \frac{d(J)}{d(u)} \times \frac{d(u)}{d(v)} \times \frac{d(v)}{d(x_1)}$$



$$\hat{y}_t = \text{softmax}(z_t)$$

$$z_t = w_y h_t$$

$$h_t = w_x \cdot x + w_h \cdot h_{t-1}$$

$$\frac{\partial L_T}{\partial w_x} = \frac{\partial L_T}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial z_T} \cdot \frac{\partial z_T}{\partial h_t} \cdot \frac{\partial h_t}{\partial w_x}$$

but h_{t-1} also contributes to the

* vanishing gradient problem
  in long input sequences the gradient becomes so small that it doesn't contribute significant change in the far away weights.

* LSTM.
  has gates.

* the forget gates solves the vanishing gradient problem as the gradient doesn't deminish on all nodes, only on contibuting nodes.

Machine Translation.

* parallel corupus

* Neural Machine Translation (NMT)

encoder, decoder.
↳ the error & loss & weight updates only apply to the decoder part.

Beam search

Beam Search: select some outputs of a decoder layer & pass it forward to generate some outputs sequences then compute the sentence probability of these generated sequences, select the best.

Brute Force: generate all possible sentences & compute their sentence probabilities choose the best.