streamstech

Salman

23/01/24

NLP

Task Contamination → doesn't work on data that haven't been seen before in LLMs.

Edit Distance: graffe

candidate words: giraffe graf grail

{ Spell-Checking }

Coreference: Harvard President Claudine

→ Harvard University " "

They are coreference because they are directed towards a particular word.

The apple was red

2$ 2$ 4$ 4$

1. The apple was big and red

cost: 4$

2. The fish was blue.

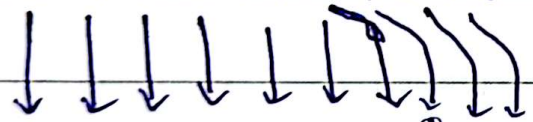
cost: 8$

By calculating the cost we can find which sentence is more similar.

Operation and cost
Insertion → 2$
Deletion → 2$
Substitution →

What is the minimum edit distance to make.

INTENTION → EXECUTION.



* E X E C U T I O N

d s s i s

cost: 8

LEVANSHTEIN.

Insertion — 1$

deletion — 1$

substitution — d + i

= 2$

streamstech

* Edit Distance:

1. Levenshten 2. Hamming 3. Jaro-winkler.

* Token Based:

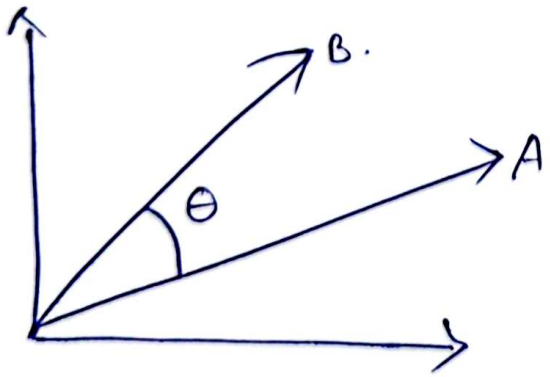
1. Cosine-similarity 2. Jaccard-Index.

3. Tversky-Index.

* Sequence Based:

1. LCS.
2. Ratcliff-Obsershelp Similarity.

Cosine-similarity



The more similar two sentences are, the more it will coincide making θ smaller and smaller.

cosine similarity (doc1, doc2)

$$= \frac{\vec{A} \cdot \vec{B}}{|A| \cdot |B|}$$

Doc 1: The cat in the hat.

Doc 2: A black cat in a black hat.

Unique tokens:

	The	Cat	in	Hat	a	Black
Document 1	2	1	1	1	0	0
Doc-2	0	1	1	1	2	2.

vector $A = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ $B = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \end{bmatrix}$

$$\text{similarity} = \frac{\vec{A} \cdot \vec{B}}{|A|\,|B|} = \frac{(2\times0) + (1\times1) + (1\times1) + (1\times1) + (0\times2) + (0\times2)}{\sqrt{7} \cdot \sqrt{11}}$$

$$= 0.342.$$

Perpendicular vector (no similarity) close to 0

Disadvantage

→ can't detect semantics of sentence.

> Doctor ~~di~~ came after patient died
> Doctor came before patient died
>
> (opposing sentence).

→ No negative values.

→ Sensitive towards length of the vector. More length will have more sparsity.

Sparsity: A matrix/vector having lots of zeros

Jaccard Similarity:

$$JS(doc1, doc2) = \frac{doc1 \cap doc2}{doc1 \cup doc2}$$

→ I wake up early
→ I go to Canada for Phd. } no similarity

Cat, hat

= {'The', 'cat', 'in', 'the', 'hat'} ∩ {'A', 'black', 'cat', 'in', 'a', 'black', 'hat'}.

{'The', 'cat', 'in', 'hat', 'a', 'black'}.

= 3/6 ≒ 50%.

Lematization : Root of word

→ throw threw thrown

streamstech

24 Jan 24

N-gram Language Models

"Please turn in your homework"

Converting the sentence to bi-grams we get:

bi-grams = [Please turn, Turn in, in your, your homework]

tri-grams = [Please turn in, turn in your, in your homework]

Skip bi-grams = [Please in, turn your, in homework]

Language Model: Assigns probablistic values to Language data to do a particular task. Like GPT does to generate/predict the next word.

Markov Model: To predict something you don't have to look to far. So :-

MLE (Maximum Likelihood Estimation)

$$P(W_n \mid W_{n-1}) = \frac{C(W_{n-1} \cdot W_n)}{C(W_{n-1} \cdot W)}$$

We can predict the next word by just looking at the next word.

→ $P(I \mid \langle s \rangle)$ [what is the probability the I is the starting of the string].

$$= \frac{C(\langle s \rangle I)}{C(\langle s \rangle)} = 2/3$$

<s> I am Salma </s>
<s> Salma I am </s>
<s> I do not like eggs </s>

→ $P(am \mid I) = 2/3$

→ $P(\langle /s \rangle \mid Abrar) = \frac{C(Abrar \langle /s \rangle)}{C(Abrar)}$

$= 1/2$ ■■■,

Longer grams contain more content.
Shorter grams often fails to capture the context.

Rule-Based: Hardcoded probability assignment.
Feature-Based - Captures all probability automatically

<s> I want Chinese Food </s>

$P(\langle i \rangle \mid \langle s \rangle) = 0.25$

$P(\langle /s \rangle \mid food) = 0.68$

We are given probabilities of bi-gram data :-

$P(\langle s \rangle \text{ I want Chinese Food } \langle /s \rangle)$

$= P(i \mid \langle s \rangle)\ P(want \mid i)\ P(Chinese \mid want)$

$P(Food \mid chinese)\ P(\langle /s \rangle \mid Food)$

$= 0.25 \times 0.33 \times 0.0065 \times 0.52 \times 0.68$

$= 0.0002$

$\Rightarrow \log(P_1 \times P_2 \times P_3) = \log P_1 + \log P_2 + \log P_3$

$\Rightarrow P_1 + P_2 + P_3 = e\ (value)$

30/01/24

NLP

* Extrinsic Evaluation :

e.g. Prediction models like gmail reply sggstns.

→ How well a model performs on real-world tasks.

* Intrinsic Evaluation:

~~eg~~ Used in a simulated environment which means Test-Based scenario to evaluate, for assessment.

e.g. Understanding Syntatic Structure.

Q: How can we say if a Language Model fits test-da better?

→ The probability of the prior language model is better.

Q: What happens if I ~~feed~~ evaluate my language model with 'Seen' data

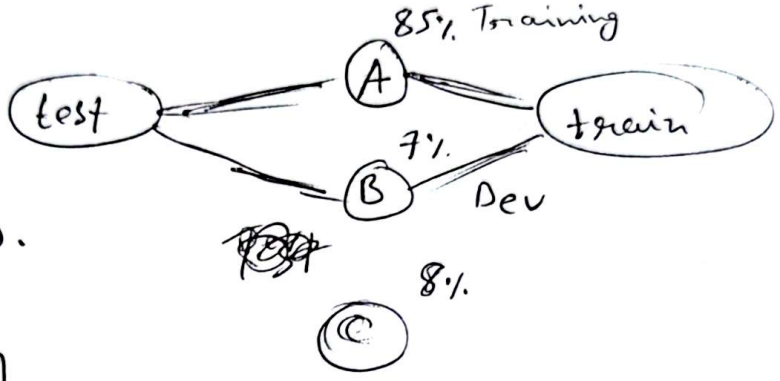
→ Overfitting: Memorizes the data instead of generalizing. Biasness increases. Performs bad in Training set. But performs well in

streamstech

Underfitting: Low Biasness. Performs bad in the Test, Training and validation set.

- Why shouldn't we train with 100% data?

=> Doesn't represent real-world scenario.



Train-test-split

EVALUATION METRIC – PERPLEXITY

→ Used to evaluate how well a model performs generative tasks.

$$\text{Perplexity, } PPL(W) = \sqrt[N]{\frac{1}{P(w_1 w_2 \cdots w_N)}}$$

$$= 2^{-1/N \sum_{i=1}^{N} \log_2 P(x_i)}.$$

"The cat is on the mat" (Test Sample).

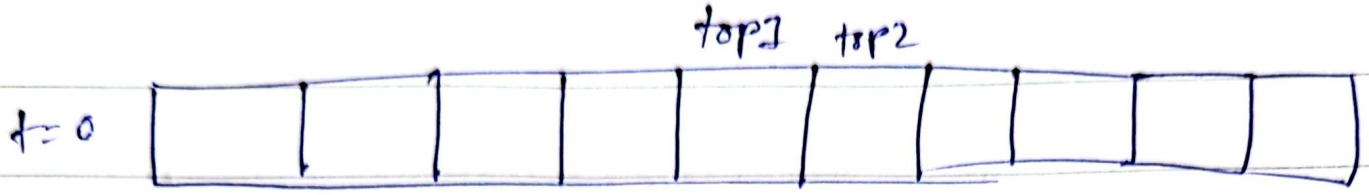
P("The") = 0.1 P("cat") = 0.2 ("is") = 0.15
("on") = 0.1 ("the") = 0.2 ("mat") = 0.25.

$$PPL = 2^{-1/6\left[\log_2(0.1) + \cdots + \log_2(0.25)\right]}$$
$$= 2^{-1/6 \times (-15.02)} = 2^{2.5} \cong 5.66.$$

⇒ There are Six Branches in the training corpus where the above probabilities will found.

The lower the Perplexity, the better the Language model in Predicting.

Algo Lab.



06-02-24

NLP

Disadvantage of a large N-gram Model.

Smoothing / Discounting: Take some from the probabilities of other values and add to counts that are zero.

The simplest Smoothing Method is Laplac method.
We ~~x~~ increase all the probabilities by 1.
So, all 0s become 1.

$$P = \frac{count(w,v)+1}{\text{all count of bigrams}+1}$$

count(i|i) = 5.

P(i|i) = 5/2533

Lap. P(i|i) = 6/2534

abc

I want food lunch.

$P = P(I \mid <s>) \times P(want \mid I) \times P(food \mid want) \times P(lunch \mid food) \times P(</s> \mid lunch)$.

A variant of Laplac smoothing is K-smoothing.

Add-K. # How do you get K? $\frac{count(w,v)+1K}{\text{all counts of bigram} + K}$.

Backoff, Interpolation

If a probability is not given at let say, Bigram we backoff and move to the Unigram.

$P(lunch \mid food) = 0$, ∄ in bigram. So, we backoff to Unigram and take that probability.

Interpolation:

$$P(W_n \mid W_{n-1} W_{n-2}) = \lambda_1 P(W_n) + \lambda_2 (\text{Bigram } P) + \lambda_3 (\text{Trigram } P).$$

$\therefore \sum^{n} = \lambda P.$

Knesser-Ney

I can't see without my reading Francisco.

Pcontinuation $\propto \{v : C(vw) > 0\}$ techniques vs before ws

$$\frac{P_c(\text{classes})}{P_c(\text{Francisco})} = \frac{2}{1}$$

Higher Probability of occuring in different contexts.

Sentiment Classification

* Binary Cls: Email spam or not.
* Multinomial Cls: Discrete Classification.
* Ordinal
* Multiclass C
* MultiLevel Classfy

streamstech

Bag of Words : Doesn't store the order of words.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \frac{P(c) \times P(d|c)}{P(d)}$$

document

$$= \underset{c \in C}{\operatorname{argmax}} \underbrace{P(c)}_{\text{(Prior)}} \times \underbrace{P(d|c)}_{\text{(likelihood)}}$$

If from a document,

Prior: Probability of documents taken from class ($P(c)$

$P(d|c)$ Probability that the document is already in class.

07 Feb 24

NLP

$P(d|C) = P(f_1 \cdot f_2 \cdot f_3 \cdots f_n | c)$

→ all possible combination of features.

This is very expensive = $^{100}C_{100}$.

To simplify it, we take a Naive approach:-

$P(f_1|c) \cdot P(f_2|c) \cdot P(f_3|c) \cdots P(f_n|c)$.

So,

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|C) = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in Position} P(w_i|c)$$

$$= \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in pos} \log(w_i|c)$$

They are called as linear classifiers. aka Generative Classifier.

Spam: 2, Normal: 3, Total: 5. $n_{doc} = 2/5$ $n_{norm} = 3/5$.

Likelihood: $P(w_i|c) = \dfrac{count(w_i, c) + 1}{\{\sum_{(w \in V)} count(w, c) + 1\}}$ → Laplace Smoothing.

vocabulary

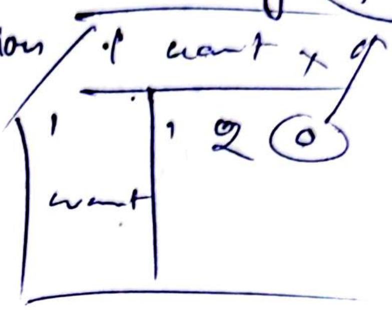
$$= \frac{count(w_i, c) + 1}{\sum_{(w \in V)} count(w, c) + |V|}$$

Ignore the words that aren't in bag of words.

When to do Laplace:

Ignore word that never occured in training (no. of occurrence of i)

Corpus. $S =$ Win ~~a~~ free vacation



$P(spam) = 3/6$ $|V| =$

$P(ham) = 3/6$

$P(win \mid spam) = \frac{1+1}{10+16}$ $P(win \mid ham) = \frac{0+1}{8+16}$

$P(free \mid spam) = \frac{2+1}{10+16}$ $P(free \mid ham) = \frac{0+1}{8+16}$

$P(vacation \mid spam) = \frac{0+1}{10+16}$ $P(vaction \mid ham) = \frac{1+1}{8+16}$

$$P(c) \cdot \prod P(w_i \mid c)$$

$$P(Spam) \times P(S \mid Spam) = \frac{3}{6} \times \frac{2 \times 3 \times 1}{26^3} \cong 0.000171$$

$$P(Ham) \times P(S \mid Ham) = 3/6 \times \frac{1 \times 1 \times 2}{24^3} \approx 0.000072.$$

I really like this book

I don't like this book.

Bag of words <u>cannot</u> handle negation expression

Modification:

Prepending: Add NOT before each token until punctuation when encountered ~~on~~ a negative word (don't, isn't).

I don't NOT_like NOT_this NOT_book

LIWC – Linguistic Inquiry

MPQA

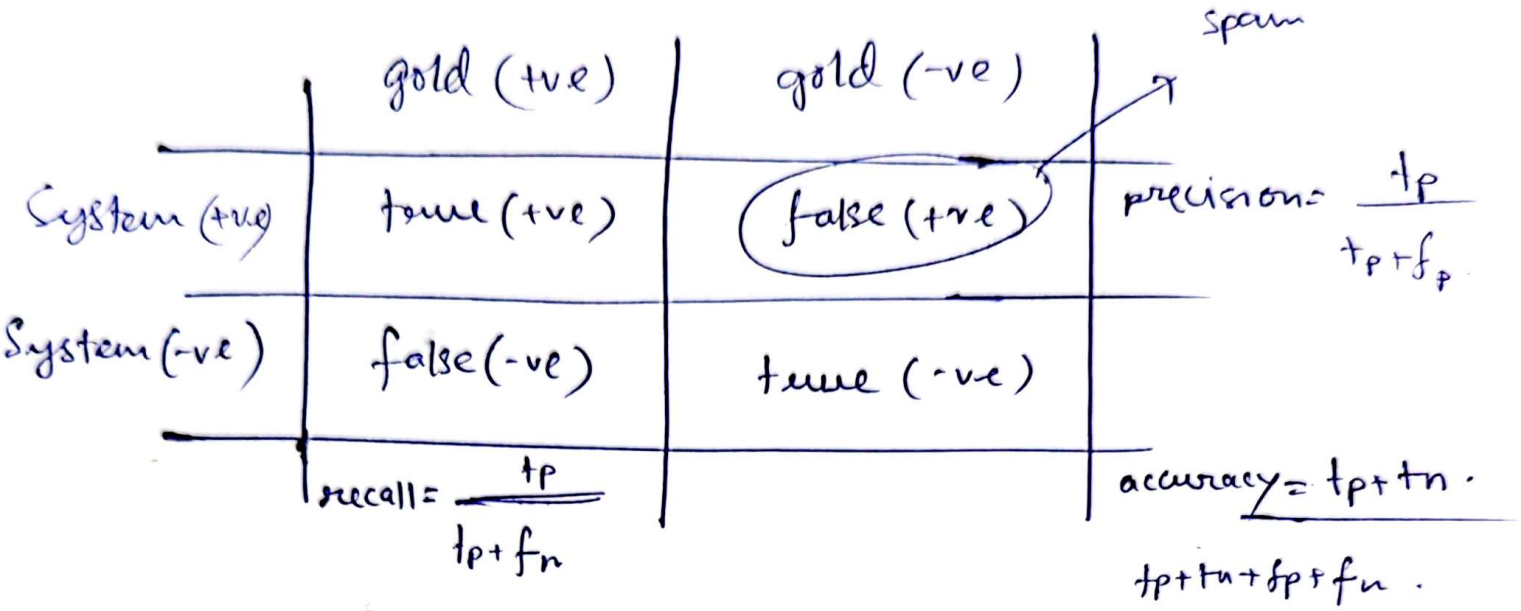
+ve : admirable, beautiful (0.9)
-ve : Catastrophe, envious (0.4)

Evaluation Metric

	Spam	Ham
Sends Ham to Spam 1% time but 99% time it classifies Spam.	99%	1%

So, Accuracy ≈ 99%

But this Accuracy although number-wise good, it's unuseful and not beneficial.



100

Spam (-ve) → 99

~~Ham~~ (+ve) → 1

acc = 99%

re = 0

false (+ve) = 0 [no spam calssified as Hami].

true Positive = 0 [Ham not correctly identified].

$$\text{f-measure} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$ → (precision, recall)

$(\beta > 1) \rightarrow R$

$(\beta < 1) \rightarrow P$.

NLP

Feb 13

Confusion Matrix

U = urgent
S = spam
n = non-spam.

	U	S	n
U	89	10	1
S	5	60	50
n	3	30	207

$P_U = \frac{89}{8+10+1}$, $R_U = \frac{8}{8+5+3}$

But how do we measure

* macro-averaging
* micro-averaging

Class 1 : urgent

8	11
8	340

P_{r_1}

Class 2

60	55
40	212

P_{r_2}

Class 3.

200	33
51	83

P_{r_3}.

~~anything~~ Every cell (row & column) wise are added other than 1st cell because they are false negatives.

$$P_{r_1} = \frac{8}{8+11}$$

$$\text{macro-avg precision} = \frac{P_{r_1} + P_{r_2} + P_{r_3}}{3}$$

$$= 0.60.$$

$$\text{micro avg precision} = \frac{268}{268+99} = 0.73$$

268	99
99	635

micro

~~Report~~ ,
when all classes are equally important, use macro.

When we want to report a particular class, use micro.

Given a class is dog, what are the features it has:

$P(c|d) = \hat{C} = P(C) \cdot \underbrace{P(d|c)}_{\text{likelihood}}$ [General Classifier,

Logistic Regression Classifier:

Discriminates between two class. Dogs wearing belt bu no cats wearing belt. So, puts a lot of weight on belt and discriminates. Finds the Discriminative Featu

Logistic Regr. Functions:

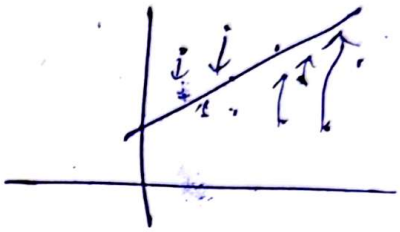
i) Feature Representation
ii) Classification Function
iii) Objective function.
iv) Optimization function.

streamstech

Linear Regression

L_2 (loss function)

Mean Square Error = $\frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$



Mean Absolute Error = $\frac{\sum |y_i - y_i^p|}{n}$ → L_1 loss

Hill Climbing Method (Local search). $y = mx + b$

for random m & b to minimize MSE.

$(0,1) \xrightarrow{m,b} (1,2)$ [Less MSE? go forward].

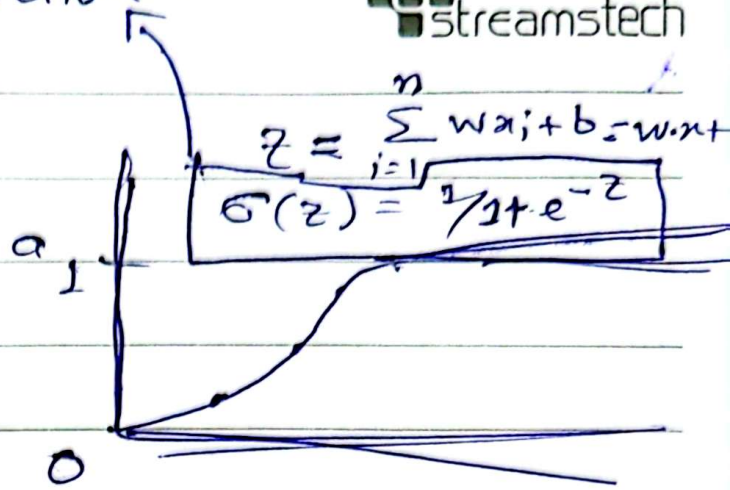
Which Loss Function should we use?

Use equal weights to all sample → L_2

Ignore Outliers (Sakib) → L_1.

SIGMOID

Sigmoid Function.



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

Do not rise above 'a'.

Decide if the point is closer to 1 or 0 (Binary Classification).

Now let's change the terminology :—

w = weight, b = bias

$$\therefore y = \frac{w}{1+e^{(-wx+b)}}$$

Calculation :

$x = [3, 2, 1, 3, 0, 4.19]$.

$w = [2.5, -5, -1.2, 0.5, 2, 0.7]$ $[b = 0.1]$.

$$P(+|doc) = \sigma(w \cdot x + b)$$

$$= \sigma(0.833) = 1/1+e^{-0.833} \simeq 0.7$$

$$P(-|doc) = 1 - \sigma(z) = 1 - 0.7 = 0.3$$

streamstech

NLP

15/02/24

Previously we saw a doc with only one 6 features and drew a regression line, But when the features are very large in number we create a matrix instead of manually calculating the probabilities.

$z = w \cdot x + b$

$$\underset{[m \times 1]}{\hat{y}} = \underset{[m \times f]}{[X]} \; \underset{[f \times 1]}{[w]} + \underset{[m \times 1]}{[b]}$$

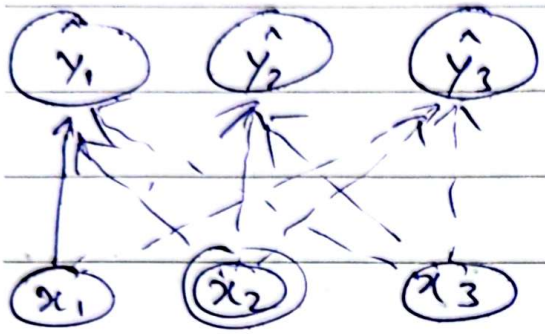
$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_f^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_f^{(2)} \\ \vdots & & & \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_f^{(m)} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_f \end{bmatrix} + \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(m)} \end{bmatrix}$$

m = number of document
f = features.

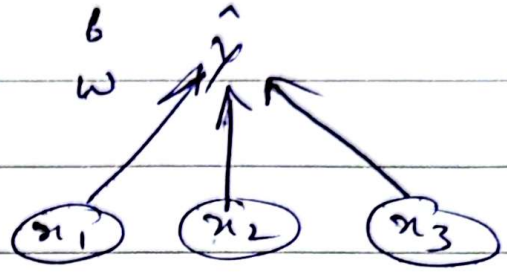
* Sigmoid Function *
(Binary Classifier).

- Classifies only one class.
- ~~Sockets~~. Softmax

For multiple classes,



For single class



(Activation Functions),

Sigmoid : $z = w \cdot x + b$, $z = \frac{1}{1+e^{-z}}$

Softmax : $\hat{y}_1 = \frac{e^{z_1}}{\sum_{i=1}^{n} e^{z}}$, $z = [0.7, 0.15, 0.02 \cdots]$

$L(\hat{y}, y)$ = How much $\hat{y}$ differs from the true y

Bernoulli Distro (when Probability is either 0 and 1).

$P(y|x) = \hat{y}^{y}(1-\hat{y})^{1-y}$ [Gold Level] = 0.

$$\log p(y|x) = y \log \hat{y} + (1-y) \cdot \log(1-\hat{y}).$$

$$L_{CE}(\hat{y}, y) = -[y \log \hat{y} + (1-y) \log(1-\hat{y})].$$

(cross Entropy)

$$= -[y \log \sigma(wx+b) + (1-y) \log(1-\sigma(wx+b))]$$

when we can differentiate it, we can optimize it.

-infinity to 0 is the range. A good classifier will have value close to 0. A bad classifier will go towards -infinity.

Gradient Descent: Slope of a straight line or curve.

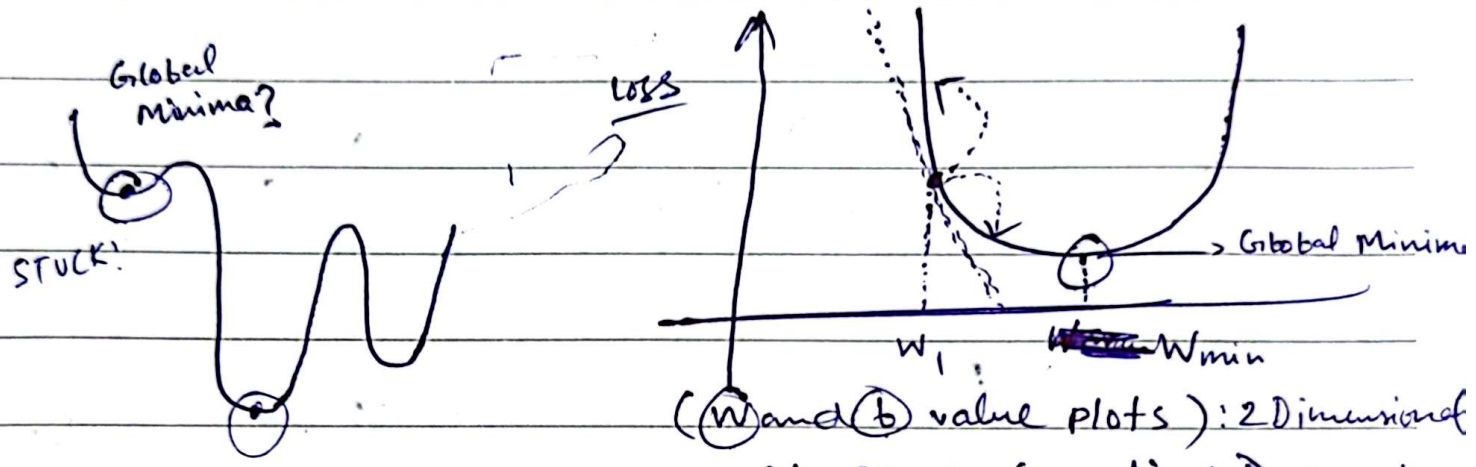
Loss Function: Curve -> when value is 0.

Loss Function,

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{CE}\left(f(x^{(i)}; \theta), y^{(i)}\right)$$

$(\hat{w}, \hat{b})$ ↙ (w, b)

To minimize go downward



Slope(-ve): Gradient Descent

$$w^{t+1} = w^{t} - \eta \frac{d}{dw}\left(L(f(x, w), y)\right)$$

(eta)

eta = learning rate

means how much downward you'll go to reach global minima.

$x = [x_1, x_2]$.

$w_1 = w_2 = b = 0$,

$\eta = 0.1$.

x_1 = (+ve) lexicons = 3.

x_2 = (-ve) lexicons = 2.

$$\theta^{t+1} = \theta^t - \eta \nabla L\left(f(x_i, \theta), y^i\right) \quad \Big| \quad \sigma(0) = \frac{1}{1+e^{-0}}$$

$$\nabla_{w,b} L = \begin{bmatrix} \frac{d}{dw_1} L_{CE}(\hat{y}, y) \\ \frac{d}{dw_2} L_{CE}(\hat{y}, y) \\ dw_b \, L_{CE}(\hat{y}, y) \end{bmatrix} = \begin{bmatrix} (\sigma(w \cdot x + b) - y) x_1 \\ \sigma(w \cdot x + b) - y) x_2 \\ \sigma(w \cdot x + b) - y) y \end{bmatrix}$$

$$= \begin{bmatrix} -0.5 x_1 \\ -0.5 x_2 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

$$L_{CE} = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})].$$

$$\frac{d}{dw_j} L_{CE} = (\hat{y} - y) x_j \quad \Big| \quad \sigma(w \cdot x + b)$$

$$\therefore \Theta_{t+1} = \begin{bmatrix} w_1 = 0 \\ w_2 = 0 \\ b = 0 \end{bmatrix} - 0.1 \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0.15 \\ 0.1 \\ 0.05 \end{bmatrix}$$

Stochastic Gradient Descent.

20/02/24 - NLP.

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(f(x^{(i)}; \theta), y^{(i)}) - R(\theta).$$

L2 Norm:

L_1 & L_2 Norm. → to solve Overfitting

L_2 Norm: Uses Euclidian distance.

for L_2, $R(\theta) = \alpha \sum_{j=1}^{n} \theta_j^2$

L_1 Norm: $R(\theta) = \alpha \sum_{i=1}^{n} |\theta_i|$

- Makes the bigger weights 0
- Makes a Sparse Vector (Breaks larger weights).

Embedding → Express Words as vectors in vector space based on their meaning.

- Previously we used Frequencies to classify features
- Representation Learning → Time Consuming / Bad.

Solution:

Vector Semantics: Projecting words into a multidimensional space based on their meaning.

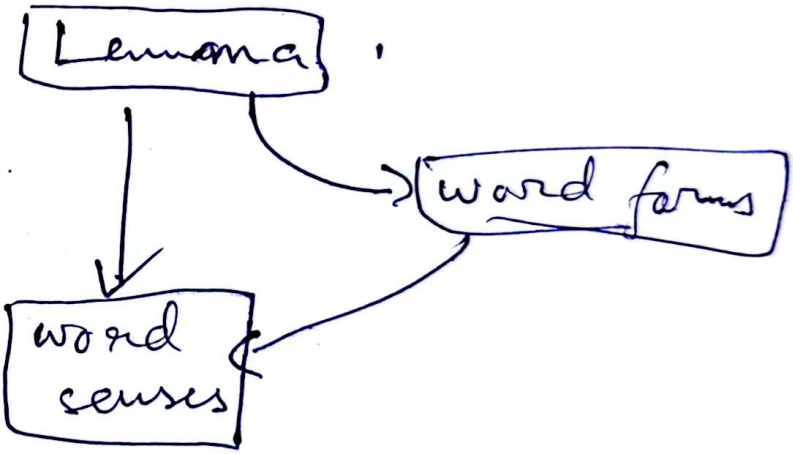
Mouse — Rodent
|
Pointer Device

Mouse → Mice

↓

root word is known as

Lemma.



Synonyms:

When two Lemmas have the same meaning.

Principle of Contrast: Similar words are not always substitutable.

Related Words;

LDA

Latent Dirichlet Allocation.

In a certain interval, which words occur more frequently? When we have a document and don't know the topic, we can use LDA.

* Connotations: i) Valence ii) Arousal iii) Dominance

Excited (—, —, —)

Projecting a word in multi-dimensional space.

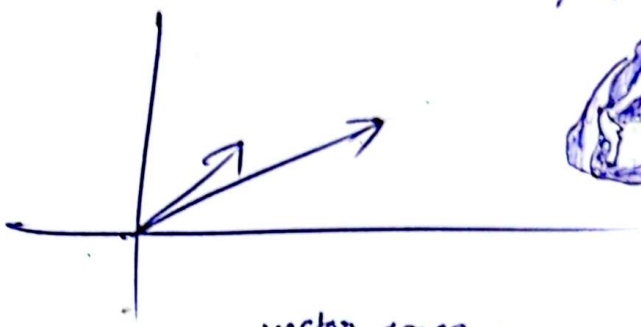
streamstech

22 Feb 24

NLP

- Similar words contain similar documents.
- Similar documents contain similar words.

Cons:
Depending too much on frequency
STILL.



vector space.

$idf = N/df$ $\quad$ $tf\text{-}idf = tf * idf$

$= \log_{10}(N/df)$

If a term appears in all of the documents then
$idf = 0$

Positive
Pointwise Mutual Information

streamstech

NLP

27 Feb.

Embedding: Dense vectors (not too sparse - datas having value zero).

TF-IDF and PPMI are too much frequency dependant which result in a Sparse vector Output.

We use 'word2vec' model to actually learn the weights and represent the data as vectors.

Focus Word ; Content word ;

CBOW

-> We know, the ~~focus wo~~ Context words but we want to predict a focus word.

Skip - Ngram:

-> We know the focus words and want to predict Content words

-> Chances of error / fault is more

~~Purpose~~

→ Skip N-gram can be used to classify the rare words more.

Negative Sampling

To find Content word we take tuples of word.

[Pad]	Data	is	processed	into	information.	[Pad]
	c	f	c			
		c	f	c		

→ The centre word is the focus word.

→ Content Window: 1 in here, before: 1 word, after: 1 word.

Here every word appears as a content word and focus word at least once.

Negative Sampling: Sol^{n} to disaster caused by softmax Function.

$$\log p(c/w;\theta) = \frac{\exp^{V_c} \cdot V_w}{\sum_{c \in C'} \exp^{V_{c'}} \cdot V_w}$$

Scale Unigram with $P = \frac{c^{\alpha}}{\sum c^{\alpha}}$ $\alpha = 0.75$.

So, frequent words are scaled down $(3)^{0.75}$.

non-frequent words are scaled up $(0.001)^{3/4} > 0.001$.

~~What~~ If the words do occur in the context window, then is a 1 or 0.

Inherent Variablity:

→ Create Multiple embeddings

→ Bootstrap & Average to find value.