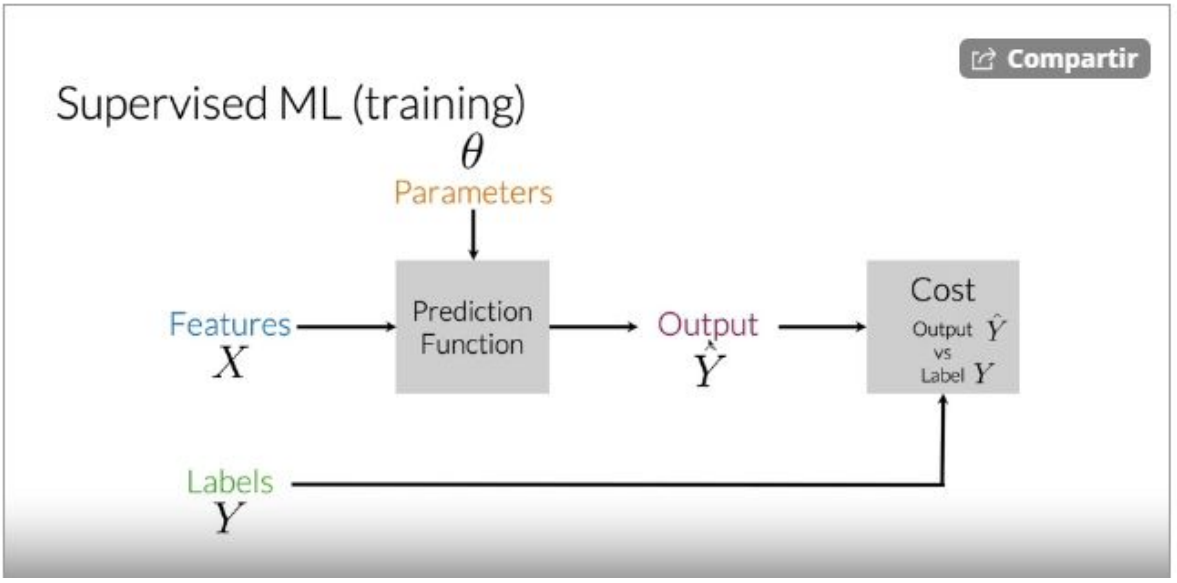
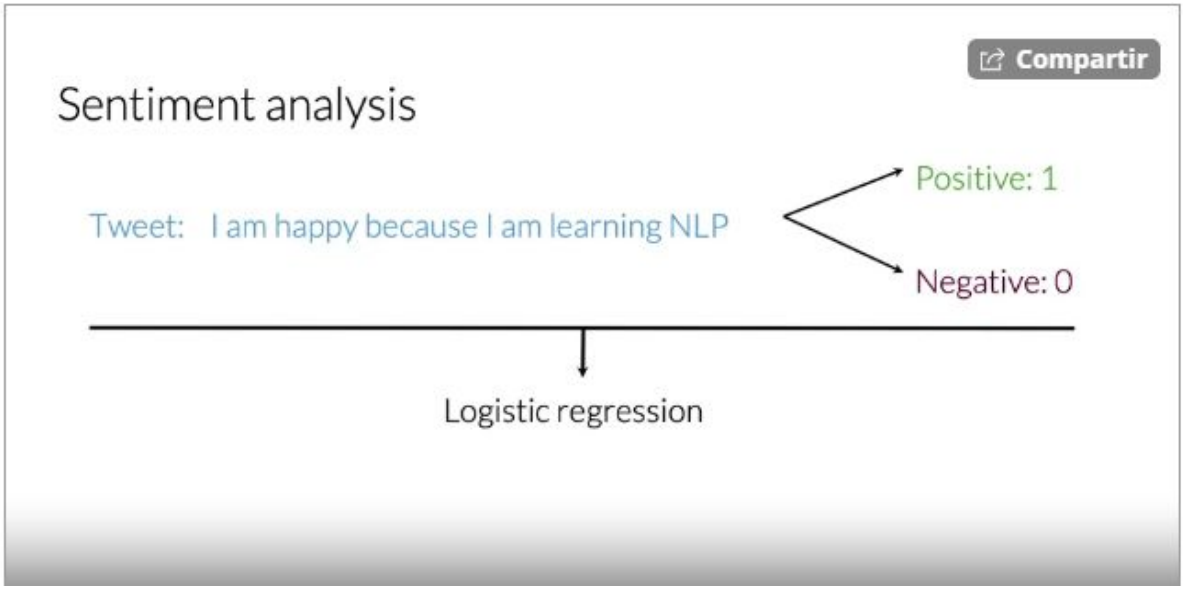
NLP SPECIALIZATION

Week 1: ***Natural Language Processing with Classification and Vector Spaces***

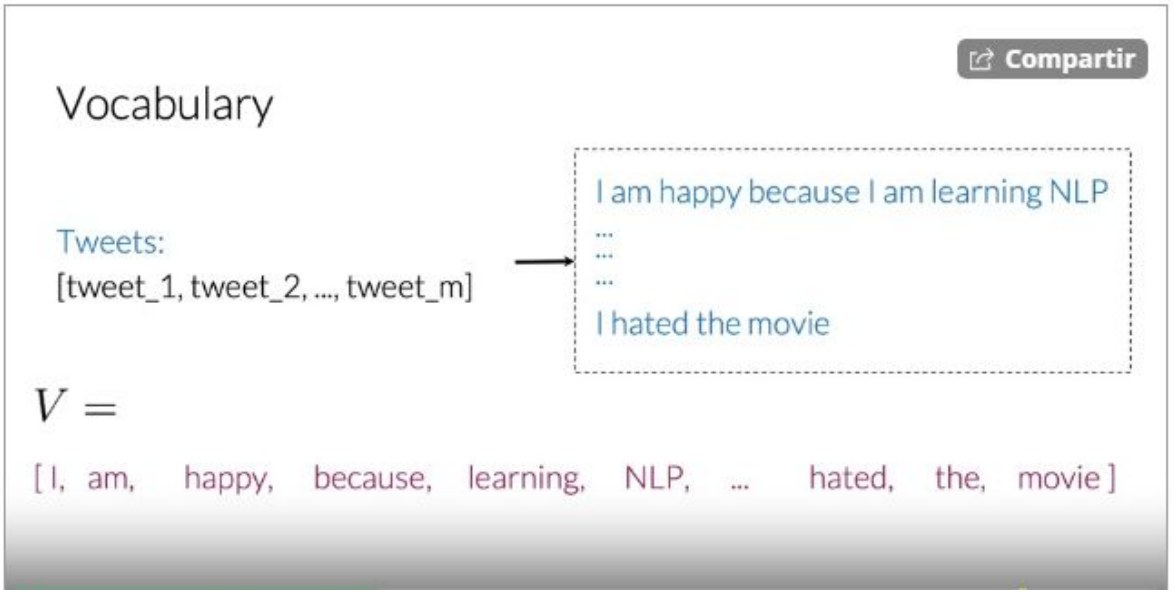
Supervised ML & Sentiment Analysis



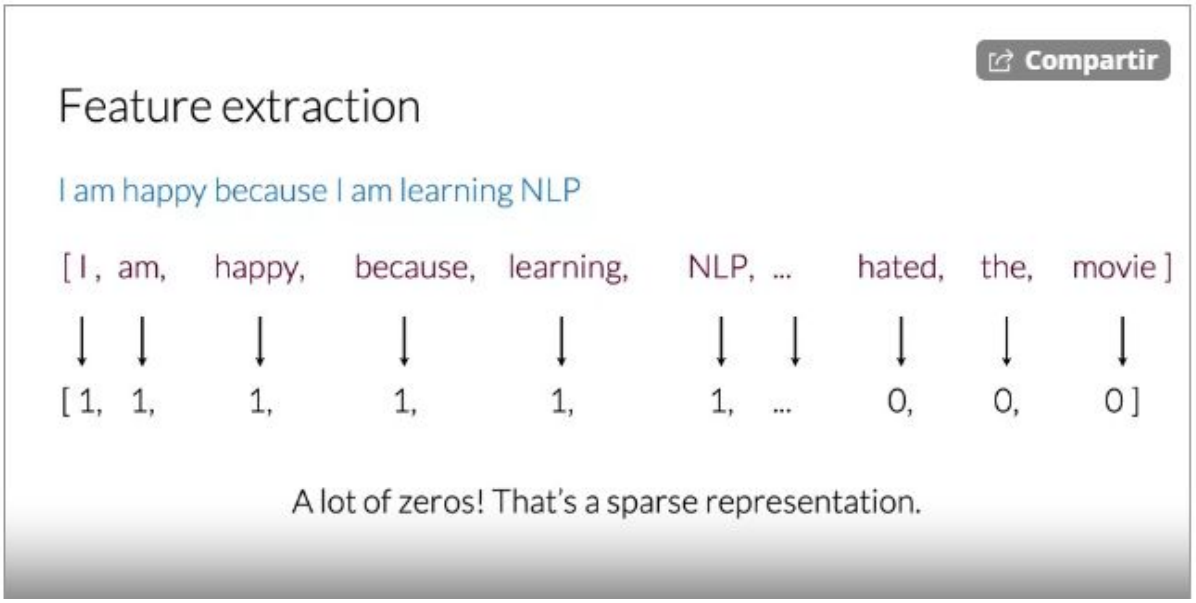


Pasos para análisis con NLP:

1) create a vocabulary

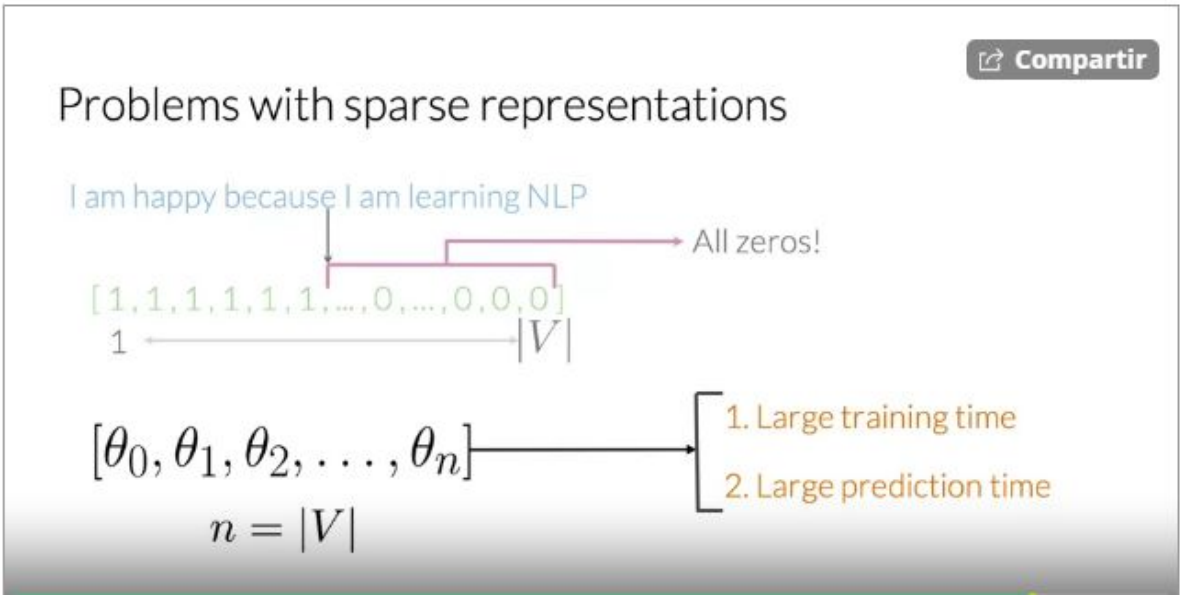


2) Feature extraction: unique words in the vocabulary (conteo de palabras únicas)

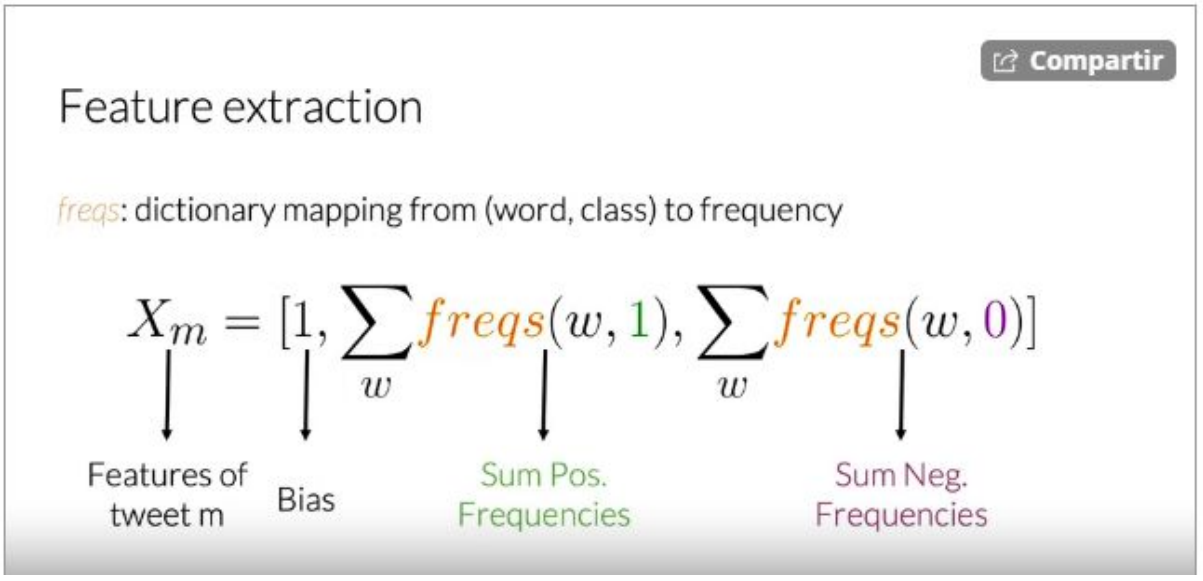


Nota: El problema con este tipo de "features" es que los vectores aparte de ser del tamaño del vocabulario son muy esparsos

Vocabulary & Feature Extraction



y hace que entrenar la logistic regression sea muy lento. Por lo tanto es mejor crear las features de interés en donde el vector ya no sea de la dimensión del vocabulario V, si no por ejemplo de 3 dimensiones
[bias, sum positive words, sum negative words]:



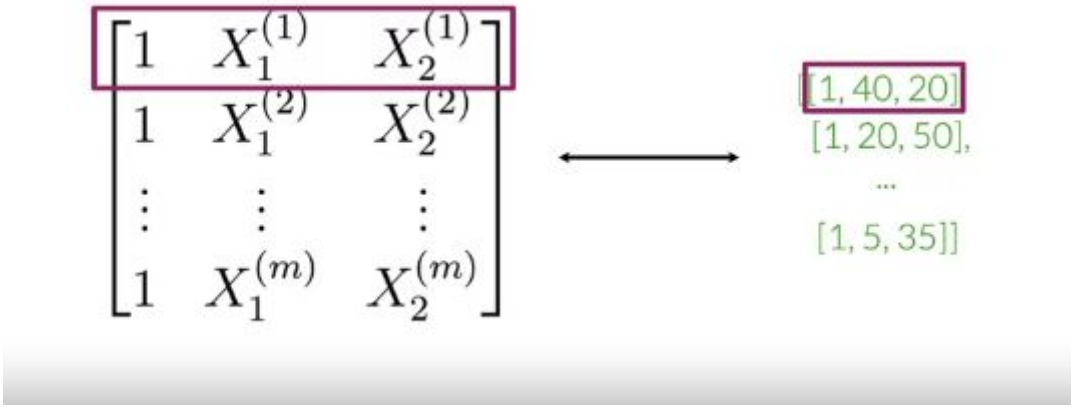
de este modo para cada tweet o texto, se obtiene una *feature* que corresponde a la cantidad de palabras positivas y negativas dentro de este, por lo tanto al tener m tweets se tendrá una matriz con este tipo de *features*

X_1 : [1, 5, 4]
X_2: [1, 3, 7]
.

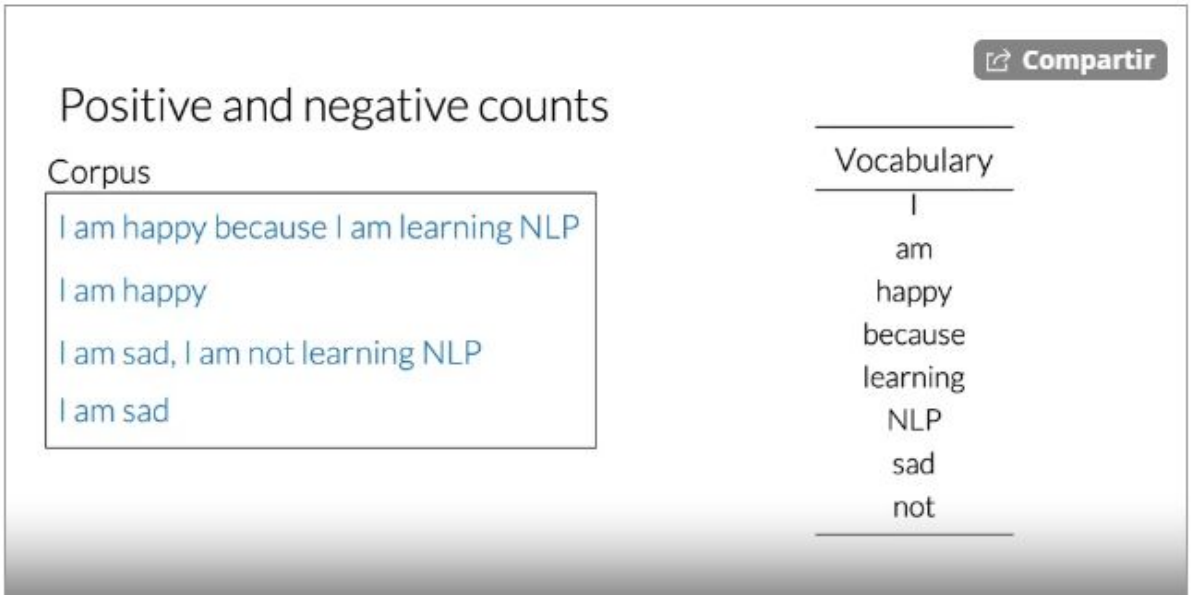
.

.

X_j: [1, n, m]



Conteo de palabras : Positive and Negative counts



En este caso al vocabulario se le agrega una columna de frecuencias de palabras por las categorías “positive” y “negative”; las categorías en las que deseamos que nuestro modelo aprenda a clasificar

Compartir

Positive and negative counts

Positive tweets

I am happy because I am learning NLP

I am happy

Vocabulary	PosFreq (1)
I	3
am	3
happy	2
because	1
learning	1
NLP	1
sad	0
not	0

Compartir

Positive and negative counts

Vocabulary	NegFreq (0)
I	3
am	3
happy	0
because	0
learning	1
NLP	1
sad	2
not	1

Negative tweets

I am sad, I am not learning NLP

I am sad

Word frequency in classes

Vocabulary	PosFreq (1)	NegFreq (0)
I	3	3
am	3	3
happy	2	0
because	1	0
learning	1	1
NLP	1	1
sad	0	1
not	0	1

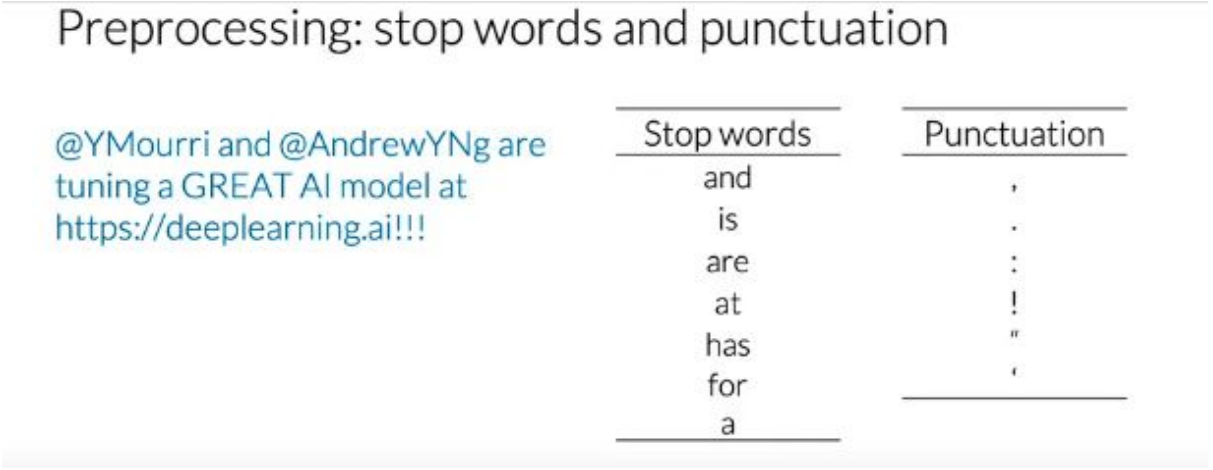
freqs: dictionary mapping from (word, class) to frequency

Nota:esto es lo que por debajo usa la técnica TF-IDF

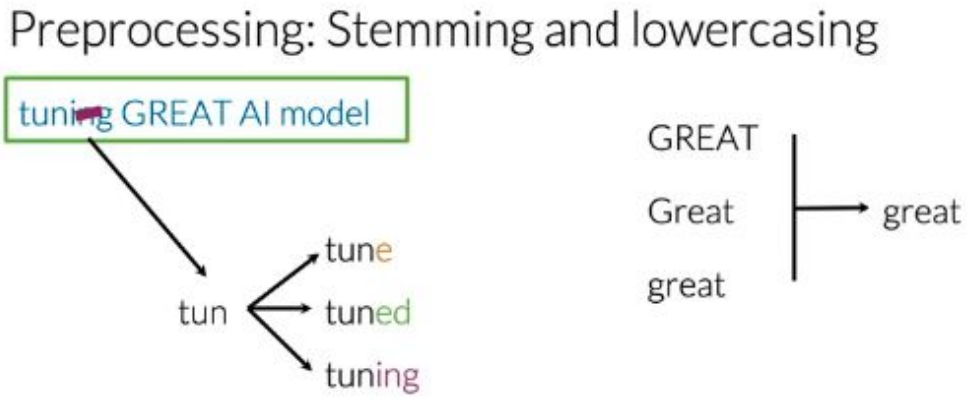
3) Preprocessing:

Tokenizing the string,Lowercasing,Removing stop words and punctuation,Stemming

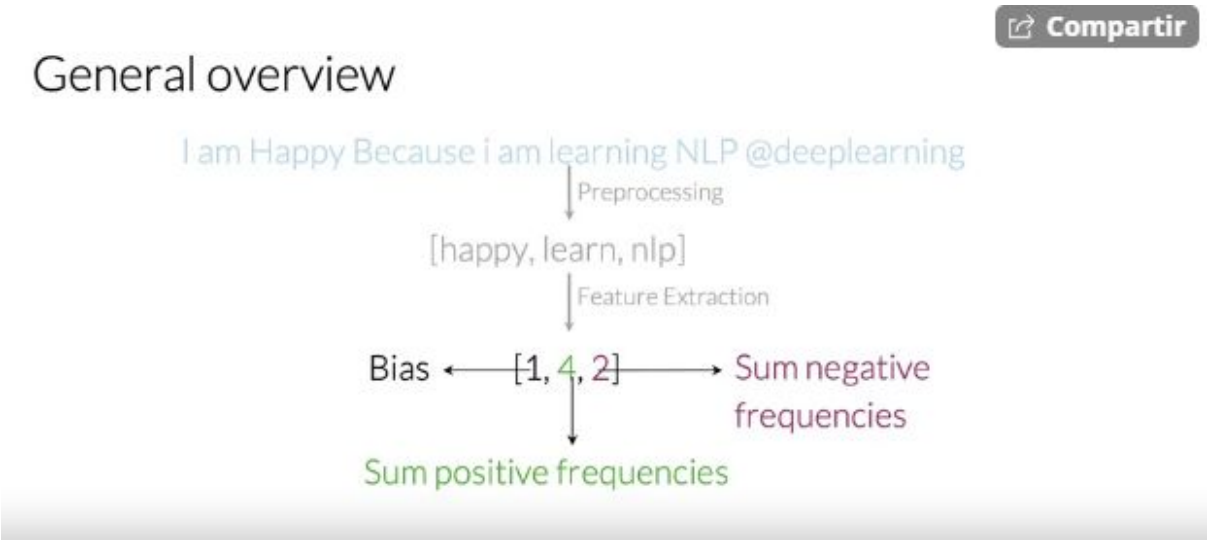
a) cleaning the text of stop words and punctuation, solo quedar con palabras de valor en el contexto

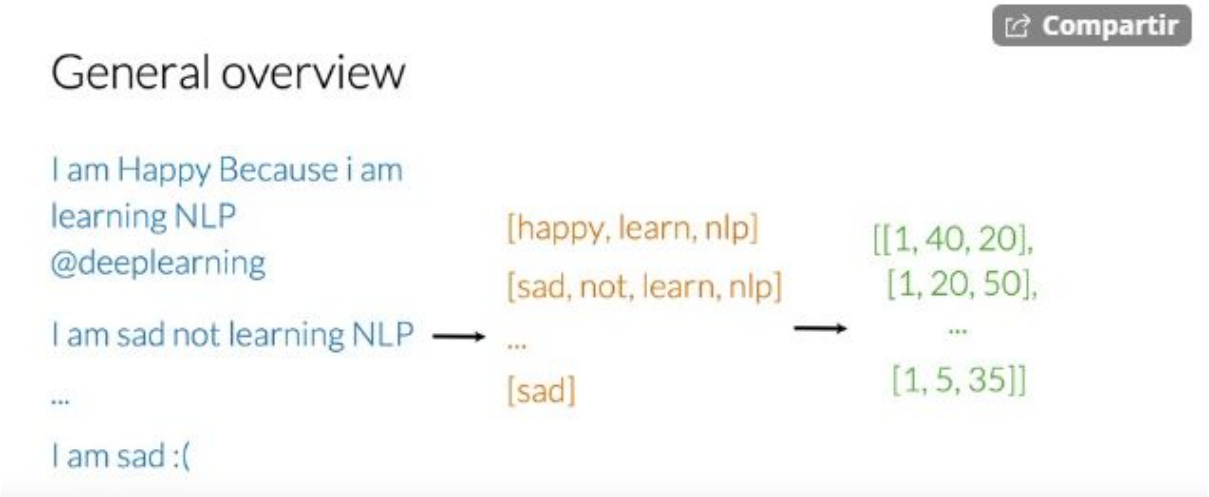


b) stemming and lower casing: lematizar a la raíz de la palabra y volver a minúsculas (HOMOGENEIZAR LA DATA)



Este proceso nos lleva a que la construcción de las features sea más homogénea





Numéricamente esto se hace de la siguientes forma:

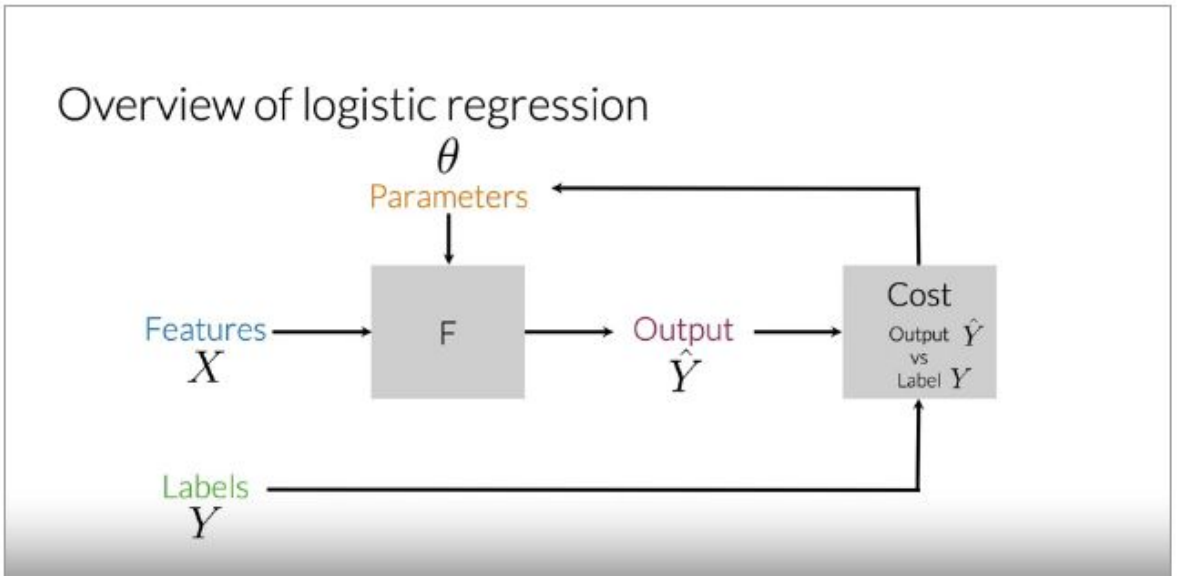
```
freqs = build_freqs(tweets,labels) #Build frequencies dictionary
X = np.zeros((m,3)) #Initialize matrix X
for i in range(m): #For every tweet
        p_tweet = process_tweet(tweets[i]) #Process tweet
    X[i,:] = extract_features(p_tweet,freqs) #Extract Features
```

process_tweet(): Cleans the text, tokenizes it into separate words, removes stopwords, and converts words to stems.
build_freqs(): This counts how often a word in the 'corpus' (the entire set of tweets) was associated with a positive label 1 or a negative label 0. It then builds the freqs dictionary, where each key is a (word,label) tuple, and the value is the count of its frequency within the corpus of tweets.

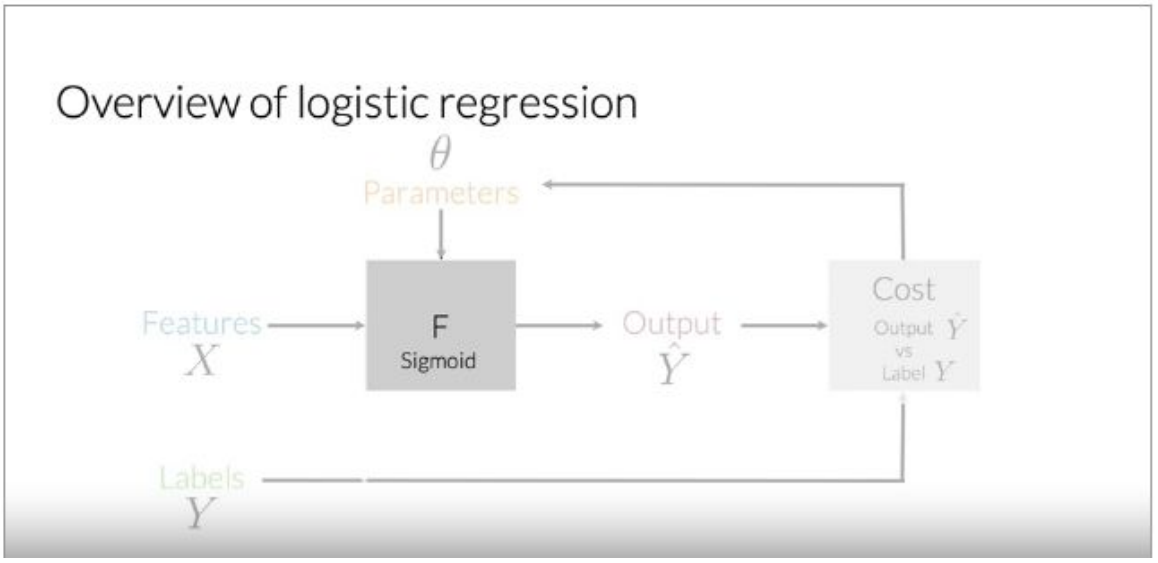
Logistic regression model for tweets classification

La función usada para el modelo de clasificación de tweets corresponde a una sigmoid, la cual toma valores positivos y negativos en el rango (0,1), 0 para negativos y 1 para positivos, con punto de inflexión en 0, que para la función corresponde al threshold 0.5

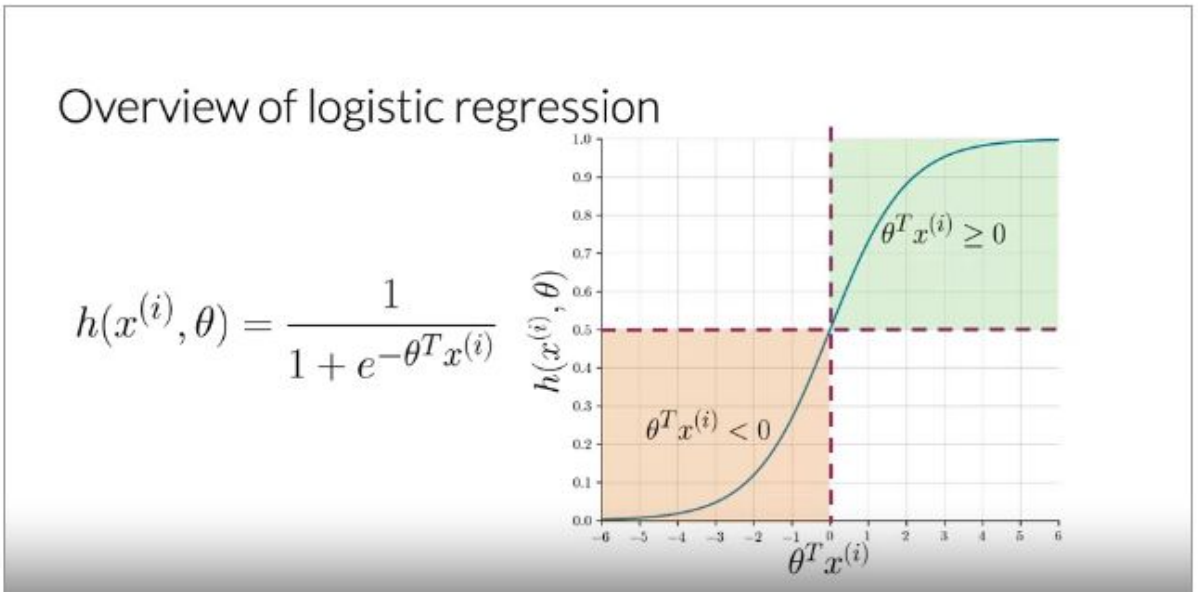
Logistic Regression Overview



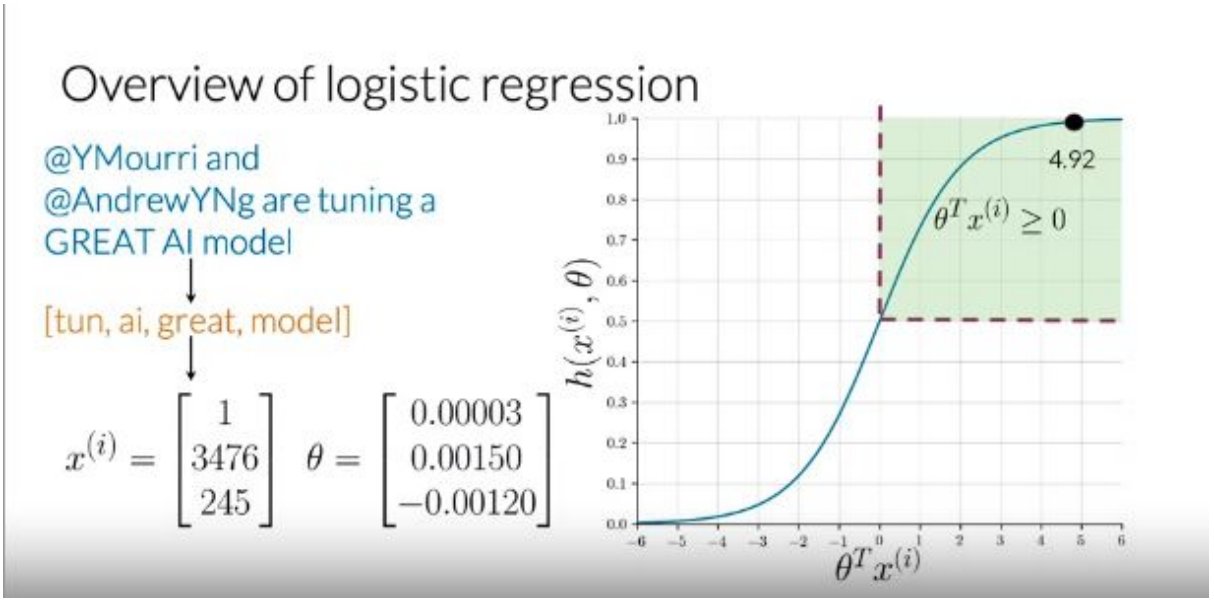
Logistic Regression Overview



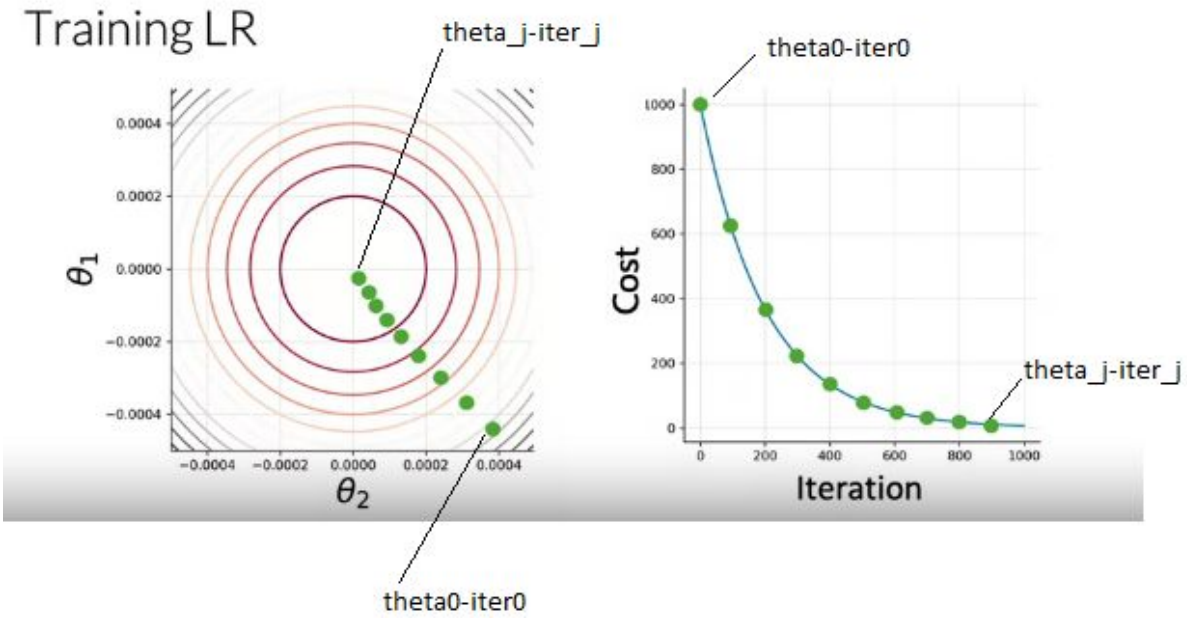
Logistic Regression Overview



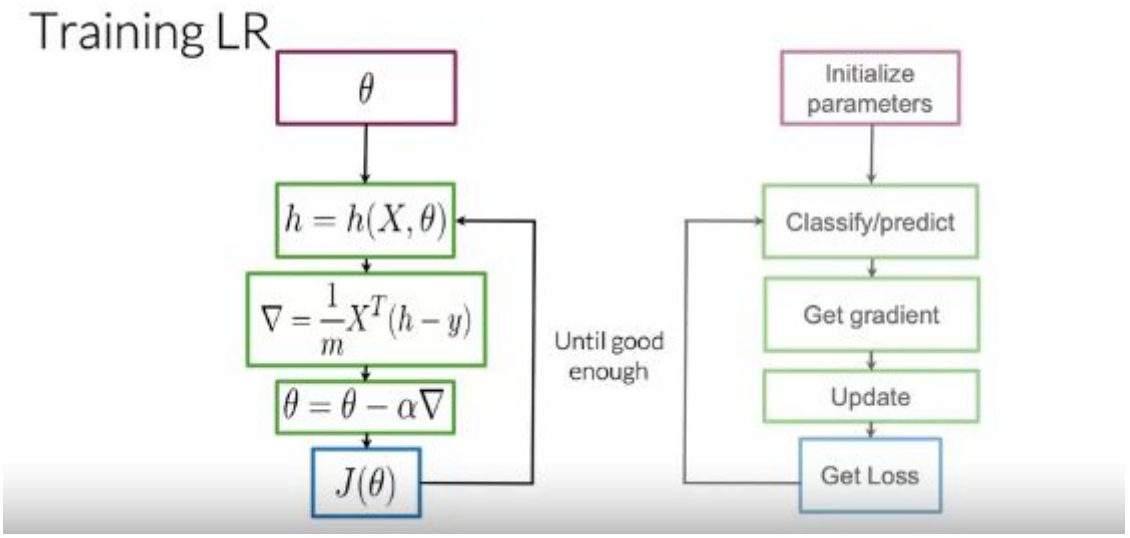
sea el siguiente tweet :



en este caso los valores del parámetro \theta llevan a una predicción en el cuadrante positivo, en donde a medida que se entrena el modelo los valores de \theta se actualizan tal que la función de costo sea mínima



Los pasos a seguir son los siguientes



Después de entrenar el modelo este se debe testear tal que se valide si ha aprendido correctamente a clasificar (se ha generalizado). Esto es, se espera que para valores predichos por debajo del threshold sea una calificación negativa 0 y para valores por encima sea una clasificación positiva 1

Testing logistic regression

- X_{val} Y_{val} θ

$h(X_{val}, \theta)$

$pred = h(X_{val}, \theta) \geq 0.5$

$$\begin{bmatrix} 0.3 \\ 0.8 \\ 0.5 \\ \vdots \\ h_m \end{bmatrix} \geq 0.5 = \begin{bmatrix} 0.3 \geq 0.5 \\ 0.8 \geq 0.5 \\ 0.5 \geq 0.5 \\ \vdots \\ pred_m \geq 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ pred_m \end{bmatrix}$$

Para validar el correcto aprendizaje se usa una métrica particular: el "*accuracy*" la cual mide la diferencia que hay entre el valor predicho y el valor esperado para todos lo *examples* en el test. Esta mide la cantidad de *True Positives & True Negatives (la correcta clasificación)*

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ pred_m \end{bmatrix} == \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ Y_{val_m} \end{bmatrix}$$

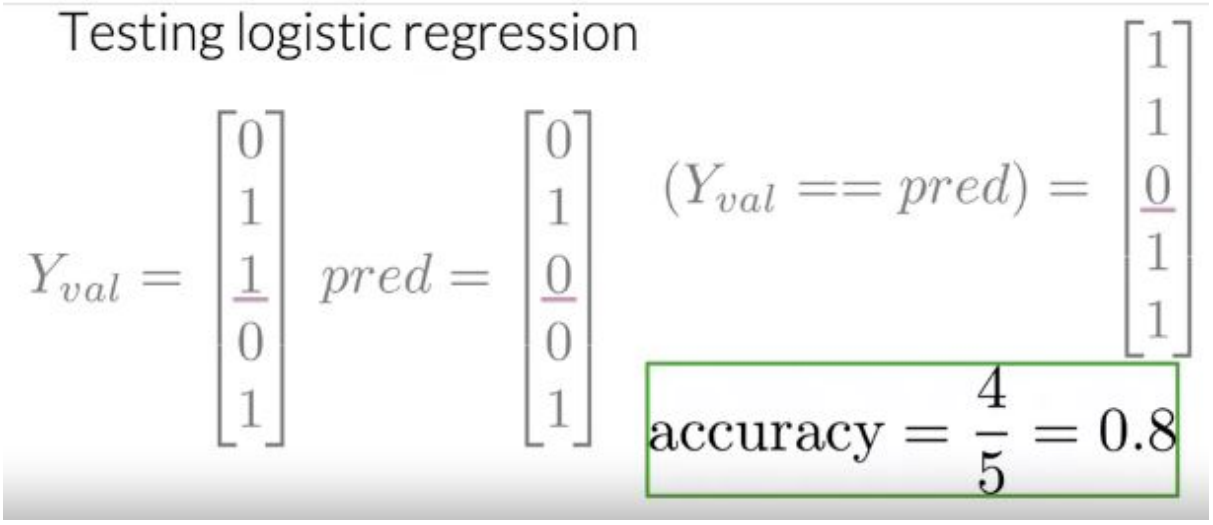
$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ pred_m == Y_{val_m} \end{bmatrix}$$

si la predicción es correcta se obtiene un 1, por el contrario se obtiene 0

$$\sum_{i=1}^{m} \frac{(pred^{(i)} == y_{val}^{(i)})}{m}$$

el resultado total de la comparación se suma y se divide por el total de observaciones

Ejemplo:



corresponde a un accuracy (TP) del 80%

Logistic regression cost function

La forma de la función de costo para la logistic regression es de la siguiente forma

The cost function used for logistic regression is the average of the log loss across all training examples:

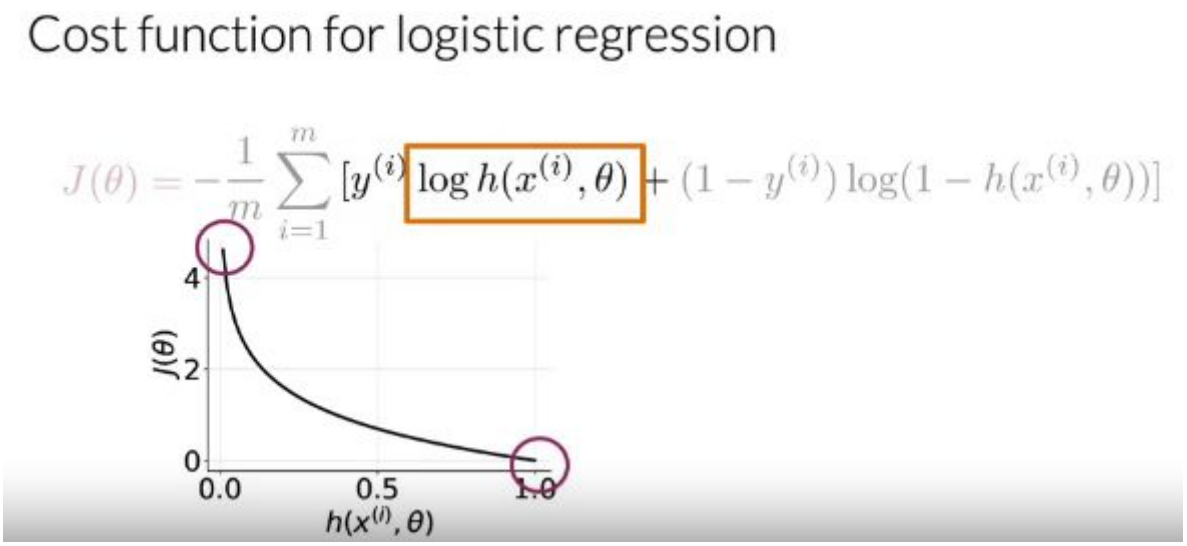
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(h(z(\theta)^{(i)})) + (1 - y^{(i)}) \log(1 - h(z(\theta)^{(i)}))$$

The loss function for a single training example is

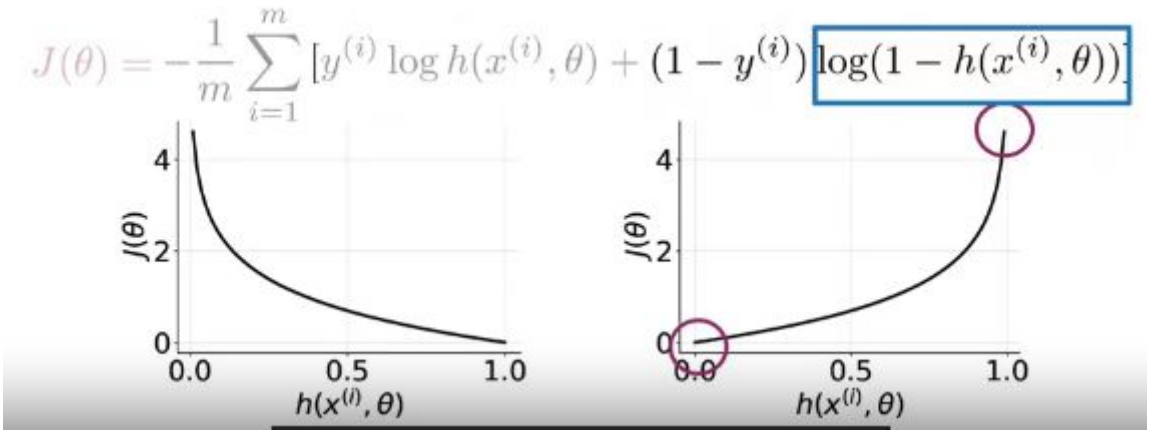
$$Loss = -1 \times \left(y^{(i)} \log(h(z(\theta)^{(i)})) + (1 - y^{(i)}) \log(1 - h(z(\theta)^{(i)}))\right)$$

Donde el objetivo es que la función de costo sea mínima. Analicemos cómo cada término ayuda a lograr dicho objetivo

primero que todo el primer término corresponde al promedio negativo de todas las observaciones lo que nos va ayudar a encontrar la dirección del mínimo global de J



El segundo término da cuenta del comportamiento de J para cuando **y=1**, en este caso a medida que la predicción h(x,\theta) se acerca al valor de y=1 el valor de J se acerca a 0, ya que la predicción concuerda con el valor esperado!. Por el contrario cuando la predicción es cercana a 0, J tiende a \inf, ya que la predicción desacuerda con el valor esperado!



El tercer término da cuenta del comportamiento de J para cuando **y=0**, en este caso a medida que la predicción h(x,\theta) se acerca al valor de y=0 el valor de J se acerca a 0, ya que la predicción concuerda con el valor esperado! Por el contrario cuando la predicción es cercana a 1, J tiende a \inf, ya que la predicción desacuerda con el valor esperado!

Update the parameters

Para poder cumplir el objetivo de que la C*ost function* tienda al mínimo global, es decir que las predicciones son correctas a cada iteración se deben ajustar los valores de los parámetros \theta y para ello se usa la técnica del gradiente descendiente, la cual se obtiene de la derivada respecto a \theta de la Cost function **J:**

Update the weights

To update your weight vector θ, you will apply gradient descent to iteratively improve your model's predictions. The gradient of the cost function J with respect to one of the weights θ_j is:

$$\nabla_{\theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h^{(i)} - y^{(i)})x_j$$

- 'i' is the index across all 'm' training examples.
- 'j' is the index of the weight θ_j, so x_j is the feature associated with weight θ_j
- To update the weight θ_j, we adjust it by subtracting a fraction of the gradient determined by α:

$$\theta_j = \theta_j - \alpha \times \nabla_{\theta_j} J(\theta)$$

- The learning rate α is a value that we choose to control how big a single update will be.

Naive Bayes Model for tweets classification

Este modelo usa como base la probabilidad de que ocurra un evento u otro, por ejemplo sea el caso de tener N=20 tweets, en donde hay una porción de tweets positivos y negativos.

Supongamos que tenemos Npos=13 tweets positivos, por lo tanto la probabilidad de que ocurra el evento de un “tweet” positivo o negativo viene dada como:

Probabilities

Corpus of tweets

A → Positive tweet

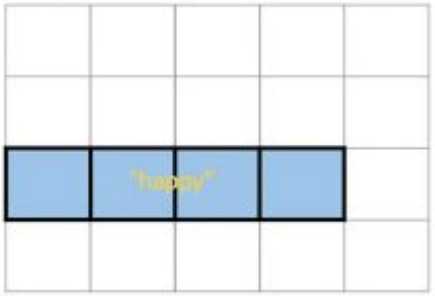
$P(A) = N_{pos} / N = 13 / 20 = 0.65$

P(Negative) = 1 - P(Positive) = 0.35

Sea ahora el caso que dentro de estos tweets (positivos y negativos) se menciona la palabra “happy”, la probabilidad de que ocurra este evento viene dada como:

Probabilities

Tweets containing the word “happy”



B → tweet contains “happy”.

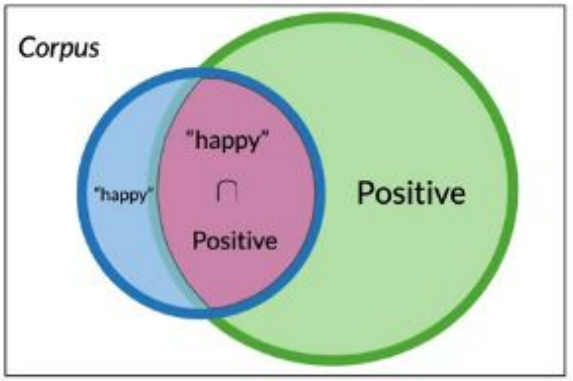
$P(B) = P(happy) = N_{happy} / N$

P(B) = 4 / 20 = 0.2

Ahora encontremos la probabilidad de que la palabra “happy” corresponda unicame/ a los tweets positivos, esto nos lleva a la probabilidad de intersección

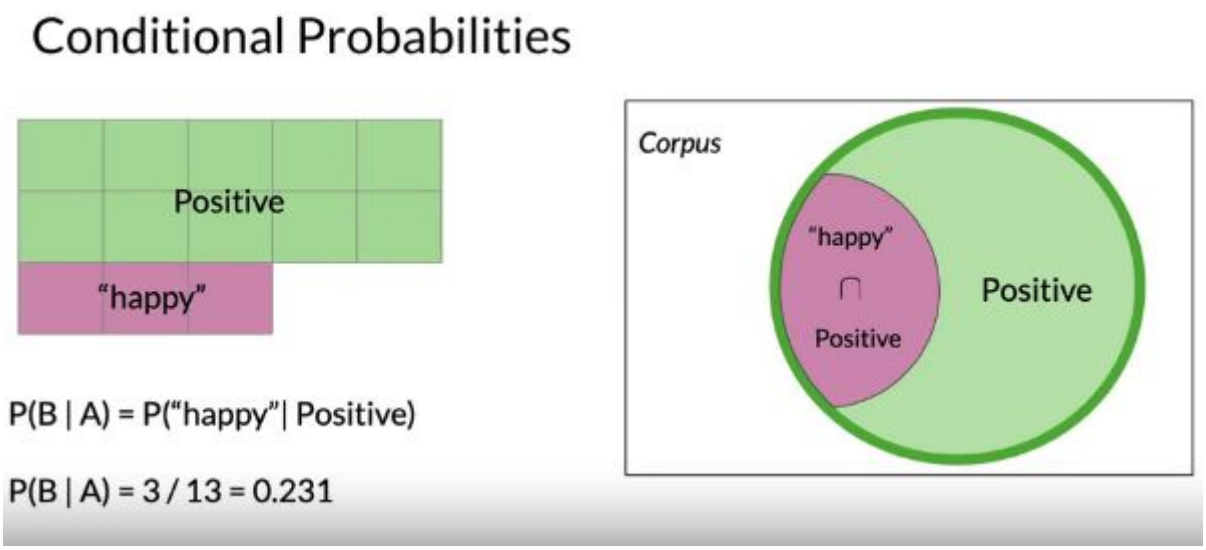
Probability of the intersection

$$P(A \cap B) = P(A, B) = \frac{3}{20}$$

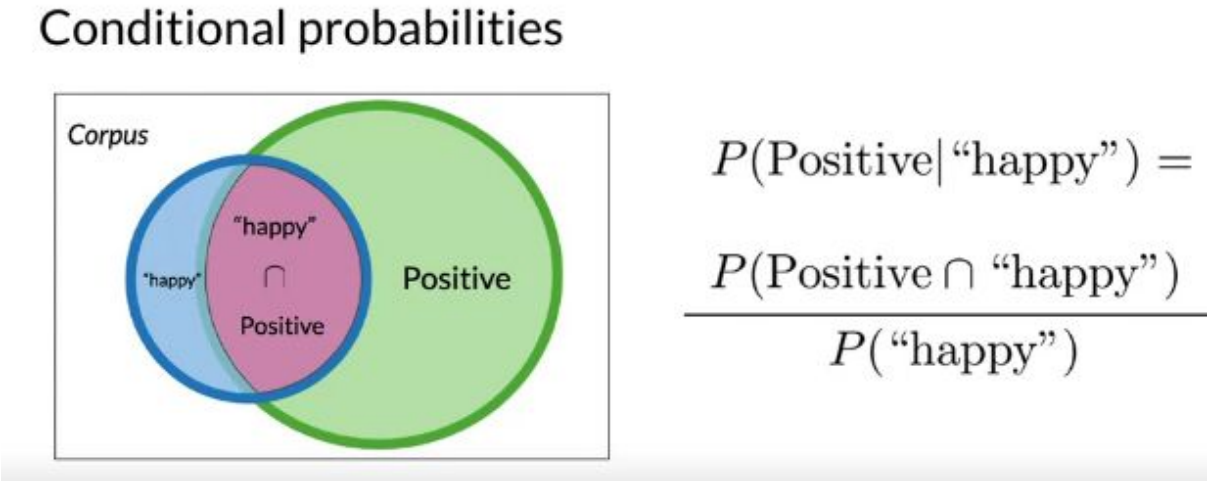


Ahora analicemos la situación de que ocurra la palabra “happy” dentro de los tweets positivos, esta situación nos lleva al caso de la probabilidad condicionada, la cual se denota como:

P(B|A) : la probabilidad de que ocurra el evento B, habiendo ocurrido el evento A
En este caso el evento B: se encuentre la palabra “happy”
A: los tweets positivos



En diagramas de Venn se puede interpretar también la probabilidad condicionada, por ejemplo sea el caso de querer determinar la probabilidad de que un tweet sea positivo dentro de los tweets que contienen la palabra “happy”



$$P(\text{Positive}|\text{“happy”}) = \frac{P(\text{Positive} \cap \text{“happy”})}{P(\text{“happy”})}$$

$$P(\text{“happy”}|\text{Positive}) = \frac{P(\text{“happy”} \cap \text{Positive})}{P(\text{Positive})}$$

A partir de estos análisis se puede obtener el Teorema de Bayes

$$P(X|Y) = P(Y|X) \times \frac{P(X)}{P(Y)}$$

En nuestro caso de análisis tenemos que la probabilidad de que ocurra un tweet positivo dado la palabras “happy”, se obtiene como:

$$P(\text{Positive}|\text{“happy”}) = P(\text{“happy”}|\text{Positive}) \times \frac{P(\text{Positive})}{P(\text{“happy”})}$$

Luego de haber entendido el Teorema de Bayes vamos a construir un modelo de clasificación de tweets en las categorías positivas y negativas haciendo uso de esta metodología, para ello vamos a seguir los pasos 1) a 3) usados para el modelo de regresión logística, esto es:

Naïve Bayes for Sentiment Analysis

Positive tweets

I am happy because I am learning NLP

I am happy, not sad.

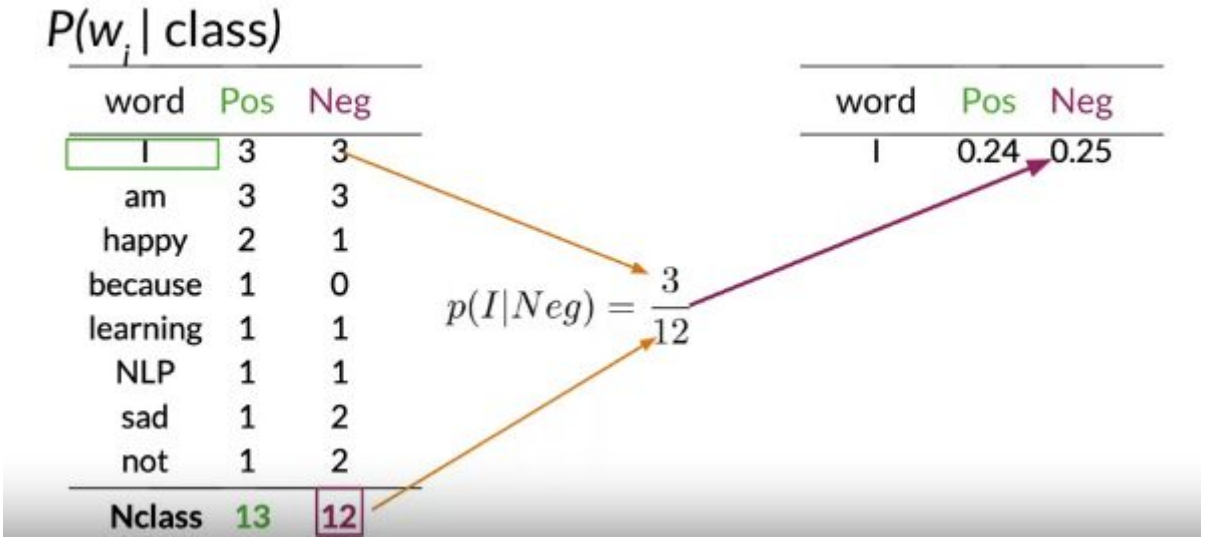
Negative tweets

I am sad, I am not learning NLP

I am sad, not happy

word	Pos	Neg
I	3	3
am	3	3
happy	2	1
because	1	0
learning	1	1
NLP	1	1
sad	1	2
not	1	2

En este caso se construye un nuevo vocabulario a partir de calcular la probabilidad de que ocurra una palabra dada dentro del vocabulario de frecuencias para cada una de las clases, esto es:



En este caso el cálculo de la probabilidad condicionada nos dará información de valor para las palabras dentro de cada categoría, como es el caso de la palabra “happy” en donde su probabilidad es mayor en los tweets positivos que en los negativos

word	Pos	Neg
I	0.24	0.25
am	0.24	0.25
happy	0.15	0.08
because	0.08	0
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.17
not	0.08	0.17

sea el siguiente caso, en donde la probabilidad del tweet se construye como el producto de las probabilidades condicionadas de cada una de las palabras dentro de tweet, sin embargo las que tienen la misma probabilidad para ambas clases se descartan y solo nos quedamos con las que difieren entre sí

Tweet: I am happy today; I am learning.

$$\prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)} = \frac{0.14}{0.10} = 1.4 \quad > 1$$

$$\frac{0.20}{0.20} * \frac{0.20}{0.20} * \frac{0.14}{0.10} * \frac{0.20}{0.20} * \frac{0.20}{0.20} * \frac{0.10}{0.10}$$

word	Pos	Neg
I	0.20	0.20
am	0.20	0.20
happy	0.14	0.10
because	0.10	0.05
learning	0.10	0.10
NLP	0.10	0.10
sad	0.10	0.15
not	0.10	0.15

en este caso la de la palabra “happy” en donde la razón entre ambas probabilidades es 1.4 que es mayor a 1 y por lo tanto su categoría es la de tweets positivos.

Para el caso cuando la probabilidad es 0 vamos a hacer uso de una técnica conocida como

Laplacian Smoothing

La cual es una nueva forma de calcular la probabilidad condicionada de una palabra en una categoría dada tal que este evite probabilidades nulas:

Laplacian smoothing to avoid $P(w_i|class) = 0$

Laplacian Smoothing

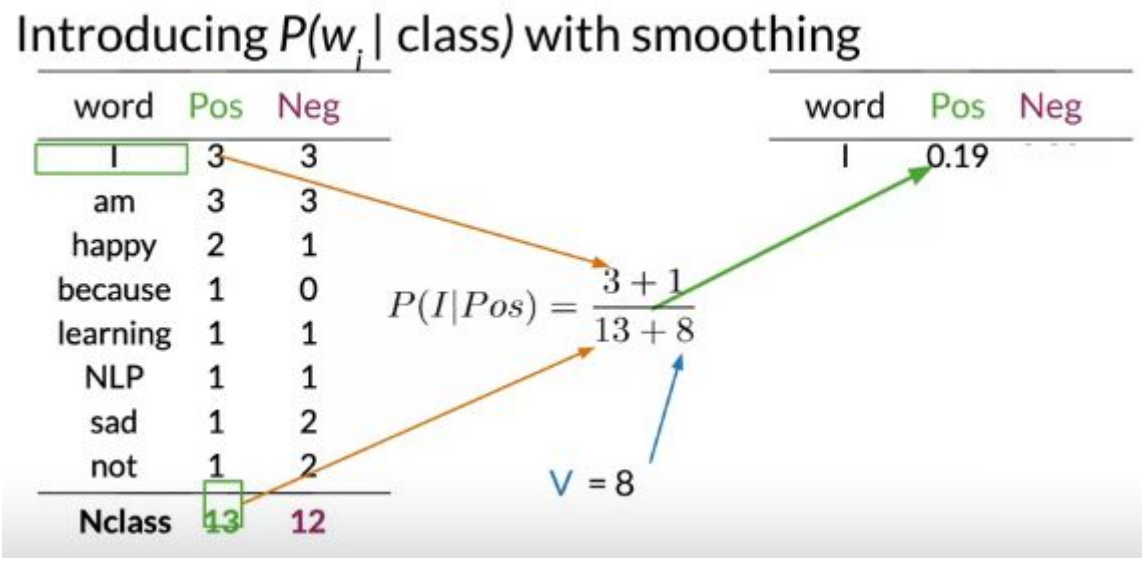
$$P(\mathrm{w_i}|\mathrm{class}) = \frac{\mathrm{freq(w_i, class)}}{\mathrm{N_{class}}} \qquad \mathrm{class} \in \{\mathrm{Positive, Negative}\}$$

$$P(\mathrm{w_i}|\mathrm{class}) = \frac{\mathrm{freq(w_i, class)} + 1}{\mathrm{N_{class}} + \mathrm{V_{class}}}$$

N_{class} = frequency of all words in class

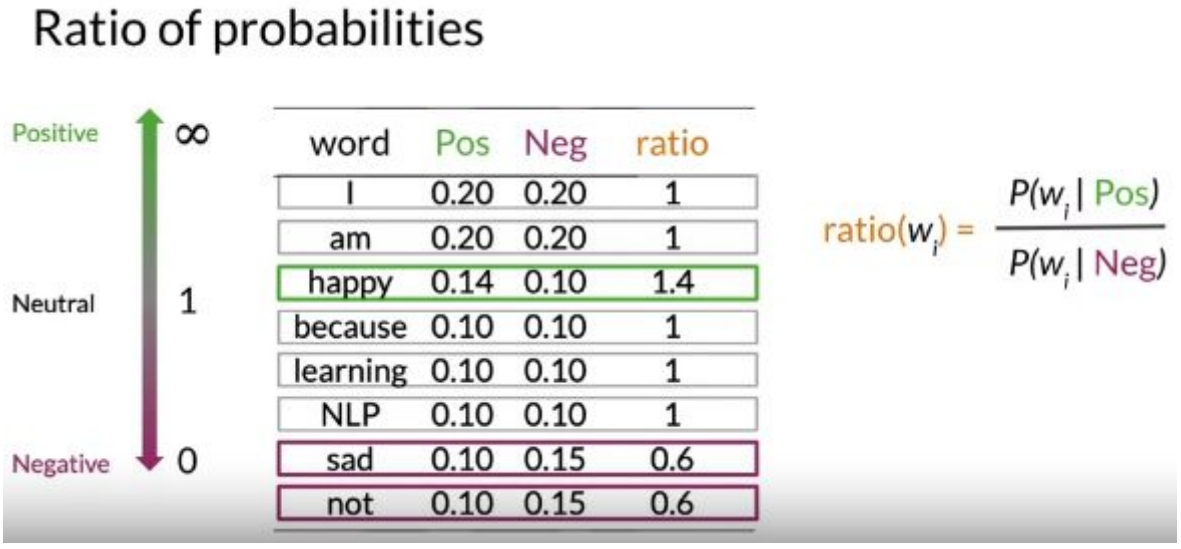
V_{class} = number of unique words in class

por ejemplo:



con esta nueva forma de calcular la probabilidad se cumple que la suma de las probabilidades debe ser = 1

Con esta nueva forma de calcular la probabilidades podemos calcular la razón entre las probabilidades positivas y negativas para obtener una escala que nos indique el sentimiento (positivo, negativo o neutro) dentro del texto



Log likelihood

Para desarrollar herramientas que clasifiquen de manera correcta un tweet como positivo o negativo vamos a hacer uso del logaritmo de la probabilidad, esto es, calcular el logaritmo de la razón entre ambas probabilidades

- Word sentiment

$$ratio(w) = \frac{P(w|pos)}{P(w|neg)}$$

$$\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$$

log likelihood se puede hacer uno por uno y añadirlo a nuestros datos como un nuevo término, en este caso denominado \lambda y tener el resultado final como la suma de cada uno

Calculating Lambda

tweet: I am happy because I am learning.

$$\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$$

$$\lambda(\text{am}) = log\frac{0.04}{0.04} = log(1) = 0$$

word	Pos	Neg	λ
I	0.05	0.05	0
am	0.04	0.04	0
happy	0.09	0.01	
because	0.01	0.01	
learning	0.03	0.01	
NLP	0.02	0.02	
sad	0.01	0.09	
not	0.02	0.03	

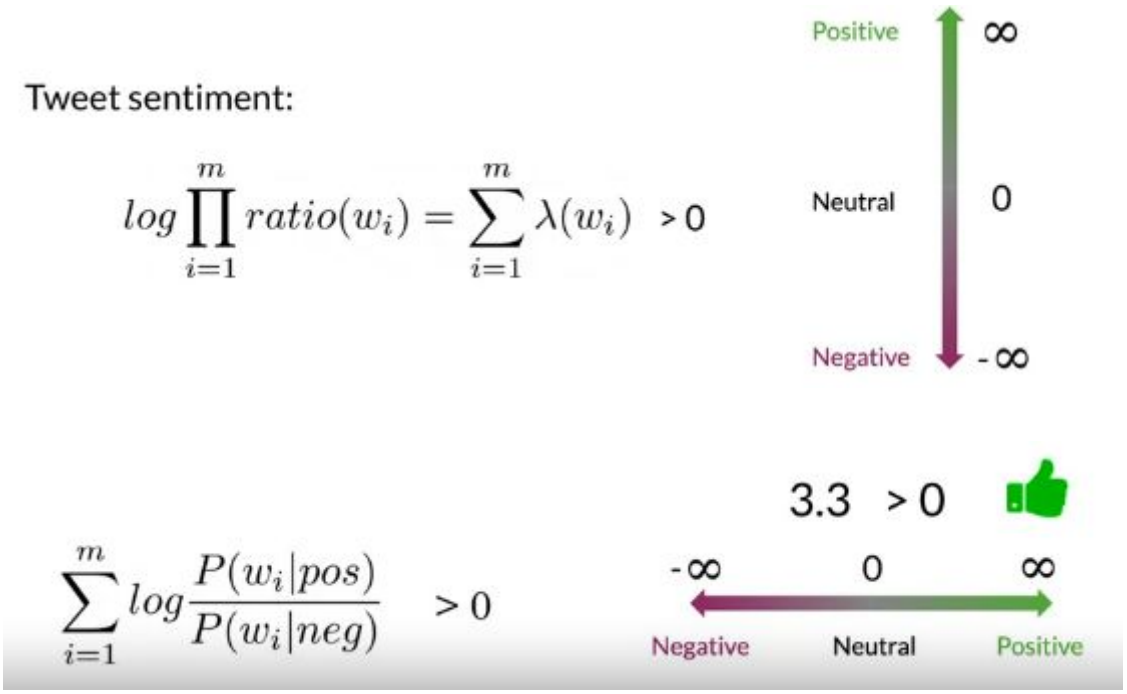
doc: I am happy because I am learning.

$$\sum_{i=1}^{m} log\frac{P(w_i|pos)}{P(w_i|neg)} = \sum_{i=1}^{m} \lambda(w_i)$$

log likelihood = 0 + 0 + 2.2 + 0 + 0 + 0 + 1.1 = **3.3**

word	Pos	Neg	λ
I	0.05	0.05	0
am	0.04	0.04	0
happy	0.09	0.01	2.2
because	0.01	0.01	0
learning	0.03	0.01	1.1
NLP	0.02	0.02	0
sad	0.01	0.09	-2.2
not	0.02	0.03	-0.4

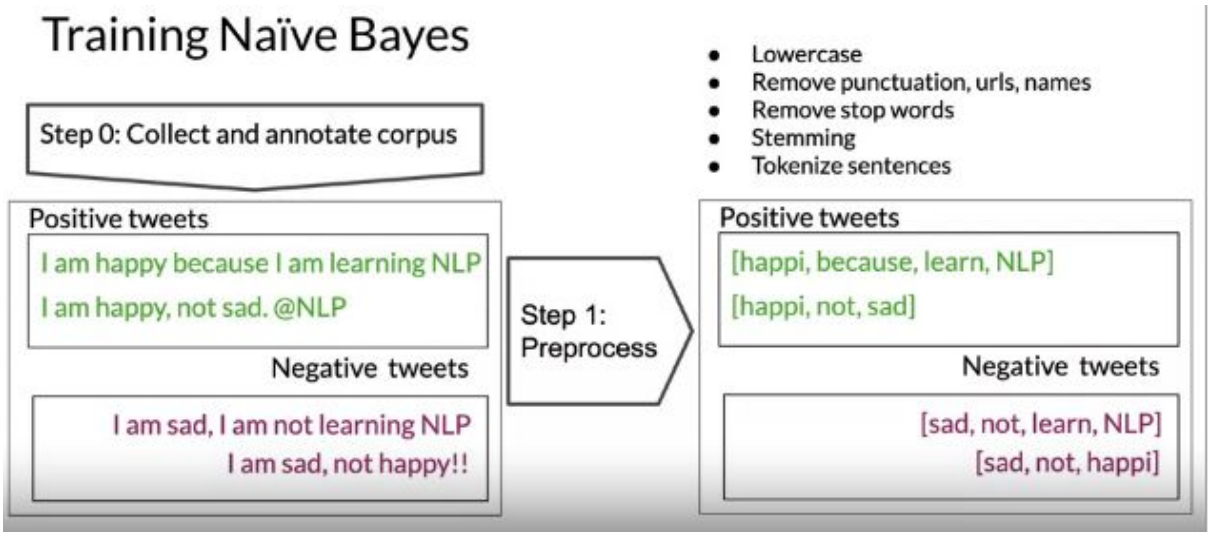
De este modo el sentimiento del tweet debe ser mayor a 0 (del caso neutro) para clasificarse como positivo

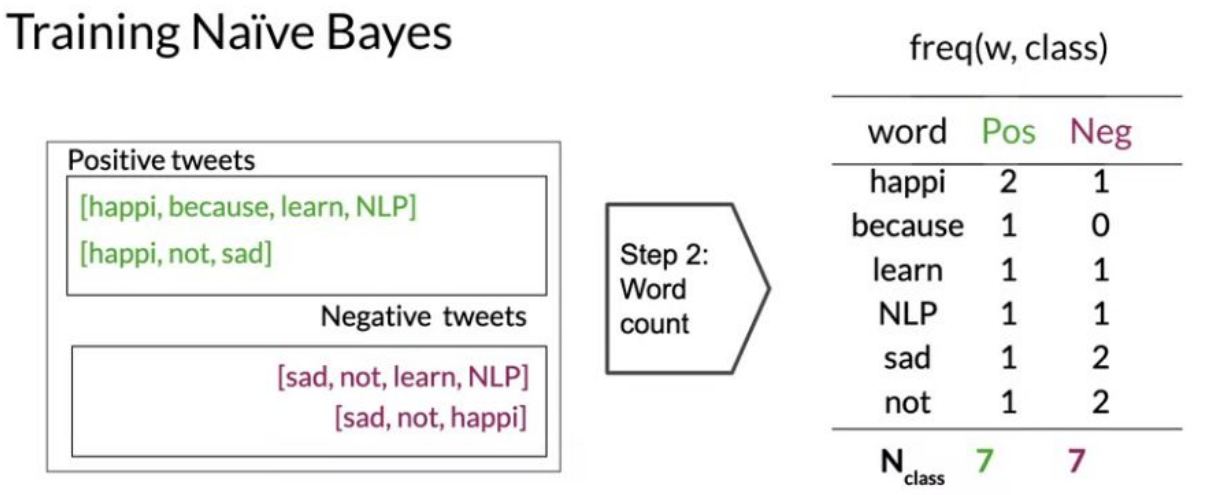


Training Naive Bayes Model

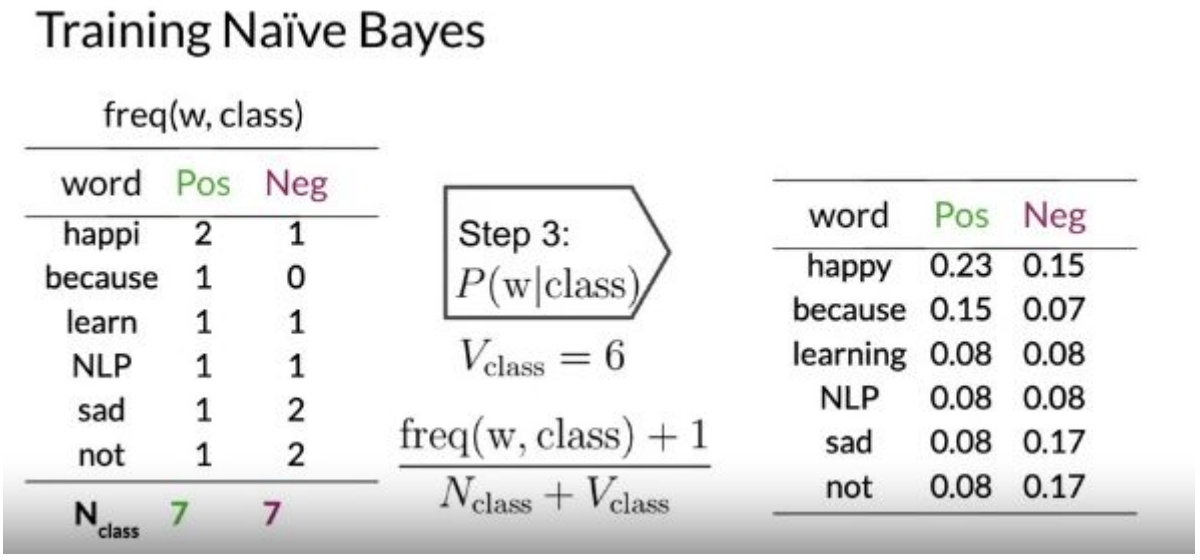
Para entrenar el modelo de clasificación de tweets en las categorías positivos o negativos vamos a seguir lo siguientes pasos (los primeros 2 pasos son iguales a los ejecutados con el modelo de regresión logística):

0. Get or annotate a dataset with positive and negative tweets
1. Preprocess the tweets: process_tweet(tweet) → [w_1, w_2, w_3, ...]
2. Compute freq(w, class)
3. Get P(w | pos), P(w | neg)
4. Get λ(w)
5. Compute logprior = log(P(pos) / P(neg))

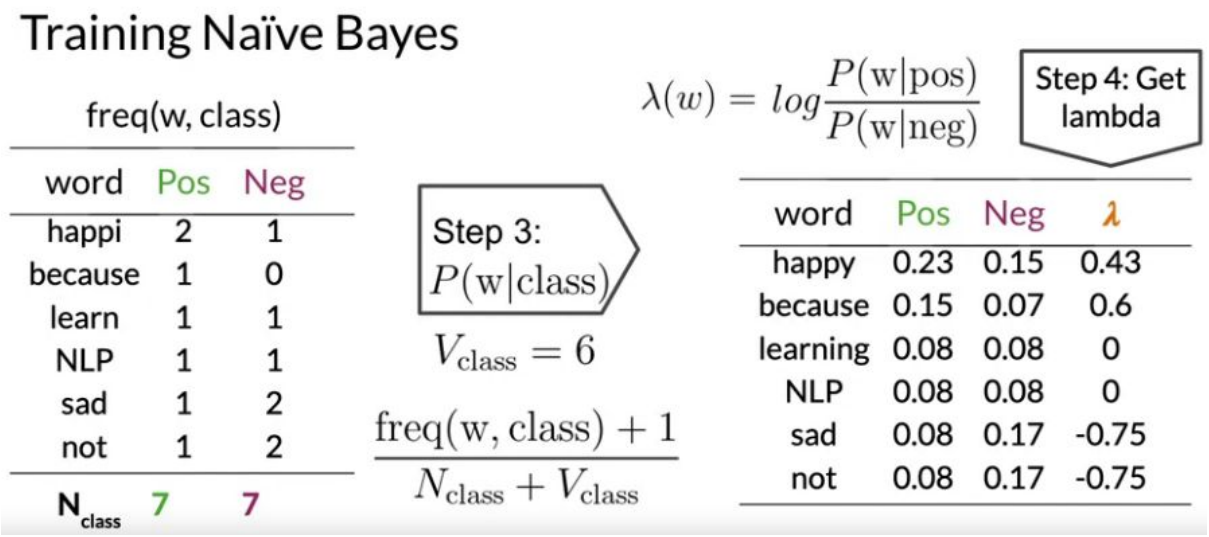




El tercer paso ya marca una gran diferencia, pues en este punto se calcula la probabilidad condicionada para cada palabra dentro de cada clase:



en el cuarto paso se calcula el valor de \lambda (el logaritmo de la razón entre las probabilidades por clase de cada palabra)



para el último paso se va construir un SCORE como la suma de las siguientes probabilidades, la log likelihood que ya la tenemos al calcular el valor de \lambda, solo necesitamos el valor de log(prior), ambas nos permitirán predecir el sentimiento de un nuevo tweet

$$log\frac{P(pos)}{P(neg)} + \sum_{i=1}^{n} log\frac{P(w_i|pos)}{P(w_i|neg)}$$

log prior + log likelihood

Esta fórmula surge del teorem a de Bayes, veamos.
Cuál es la probabilidad de que un tweet sea positivo o negativo dado un conjunto de tweets? esta pregunta se responde de la siguiente forma:

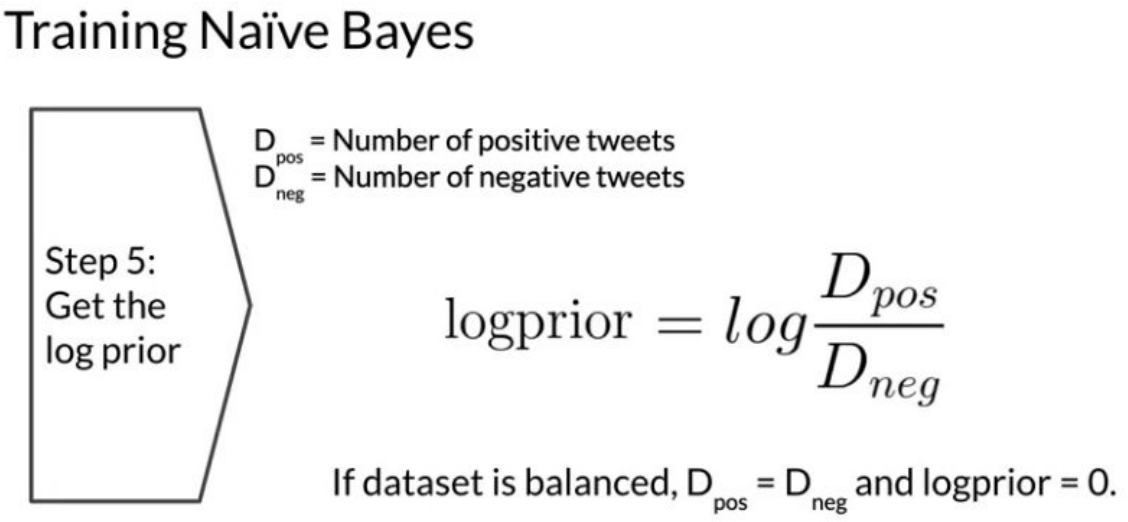
$$P(pos|tweet) \approx P(pos)P(tweet|pos)$$
$$P(neg|tweet) \approx P(neg)P(tweet|neg)$$

$$\frac{P(pos|tweet)}{P(neg|tweet)} = \frac{P(pos)}{P(neg)} \prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)}$$

Ration between probabilities

donde al calcular el algoritmo de un producto, podemos usar el algoritmo de la suma y así tenemos la expresión a calcular!

La log(prior) se calcula cómo:

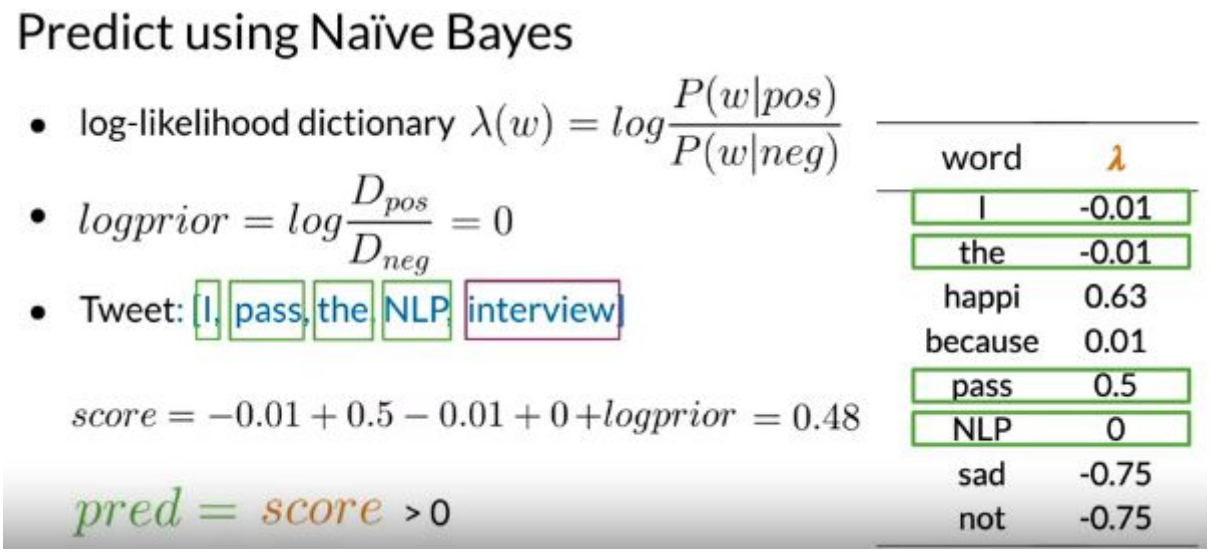


Testing the Naive Bayes Model

Con el scorre construido por medio de la log(prior) y log likelihood se obtiene el sentimiento del tweet por ejemplo, sea el caso de un nuevo tweet:

tweet*: I passed the NLP interview*

El sentimiento de este tweet se calcula con las probabilidades obtenidas del vocabulario construido, una palabra que no se encuentre dentro de este se omitirá, sea el caso de la palabra "interview"



para este caso el tweet se califica como positivo

Cuando vamos a hacer esto para un conjunto de m tweets tenemos lo siguiente:

Testing Naïve Bayes

- $X_{val}\ Y_{val}\ \lambda\ logprior$

$$score = predict(X_{val}, \lambda, logprior)$$

$$pred = score > 0$$

$$\begin{bmatrix} 0.5 \\ -1 \\ 1.3 \\ \vdots \\ score_m \end{bmatrix} > 0 = \begin{bmatrix} 0.5 > 0 \\ -1 > 0 \\ 1.3 > 0 \\ \vdots \\ score_m > 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ pred_m \end{bmatrix}$$

donde el valor predicho es un 1 o un 0, para el caso de un sentimiento positivo o negativo, respectivamente, luego hacemos una validación por medio de la métrica usada para la regresión logística, esto es, el *accuracy*, en donde se compara uno a uno la cantidad de aciertos

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ pred_m \end{bmatrix} == \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ Y_{val_m} \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ pred_m == Y_{val_m} \end{bmatrix}$$

si la predicción es correcta se obtiene un 1, por el contrario se obtiene 0

$$\sum_{i=1}^{m} \frac{(pred^{(i)} == y_{val}^{(i)})}{m}$$

el resultado total de la comparación se suma y se divide por el total de observaciones

Para testar este modelo seguimos entonces los siguientes pasos:

- X_{val} Y_{val} ⟶ Performance on unseen data
- Predict using λ and $logprior$ for each new tweet
- Accuracy ⟶ $\frac{1}{m}\sum_{i=1}^{m}(pred_i == Y_{val_i})$

Case of uses of Naive Bayes Model

Naïve Bayes Applications

- Sentiment analysis
- Author identification
- Information retrieval
- Word disambiguation
- Simple, fast and robust!

Applications of Naïve Bayes

Author identification:

$$\frac{P(\text{book})}{P(\text{book})}$$

Spam filtering:

$$\frac{P(\text{spam}|\text{email})}{P(\text{nonspam}|\text{email})}$$

Information retrieval:

$$P(\text{document}_k|\text{query}) \propto \prod_{i=0}^{|query|} P(\text{query}_i|\text{document}_k)$$

Retrieve document if $P(\text{document}_k|\text{query}) > \text{threshold}$

Word disambiguation:

$$\frac{P(\text{river}|\text{text})}{P(\text{money}|\text{text})}$$

Bank:

Naive Bayes Model Assumptions

Esta técnica supone de entrada que cada palabra dentro de un tweet es independiente a otras palabras en otros tweets, sea por ejemplo el siguiente escenario:

"It's always cold and snowy in ___."

spring?? summer? fall?? winter??

en este caso esta técnica considera a todas las palabras (spring, summer, fall, winter) igual/ probables, ya que son independientes entre sí, sin embargo por contexto se tiene que la palabra correcta es "winter"

Por otro lado este modelo sufre si la data no está balanceada, como es el caso real!

- Relative frequencies in corpus

por lo tanto:

Summary

- Independence: Not true in NLP
- Relative frequency of classes affect the model

Error analysis

Revisar los resultados después del preprocesamiento es importante ya que podríamos estar fallando al cambiar el sentido del tweet, por ejemplo:

1) quitar los caracteres especiales

Processing as a Source of Errors: Punctuation

Tweet: My beloved grandmother ✗

processed_tweet: [belov, grandmoth]

Inicial/ se espera una calificación negativa, pero al quitar el “ :) “ cambiamos todo el sentido

2) el orden de las palabras y eliminación de stopwords

Processing as a Source of Errors: Word Order

Tweet: I am happy because I did not go.

Tweet: I am not happy because I did go.

3) considerar el sarcasmo y la ironía … cómo???

Adversarial attacks

Sarcasm, Irony and Euphemisms

Tweet: This is a ridiculously powerful movie. The plot was gripping and I cried right through until the ending!

processed_tweet: [ridicul, power, movi, plot, grip, cry, end]

si no se considera podríamos calificar como negativo un tweet positivo!