Módulo II: Fundamentos de Modelos de Lenguaje

Dr. Gaddiel Desirena López

Procesamiento de Lenguaje Natural (NLP)

I.- Modelo Vectorial y Preprocesamiento de Información

Motivation

- ¿Cómo obtiene un motor de búsqueda como Google documentos relevantes de una consulta determinada?
- ¿Cómo puede una empresa tramitar las reclamaciones hechas por sus usuarios en sus portales web?

Estos problemas lo estudian:

- *Recuperación de información*: ciencia de la búsqueda de información en colecciones de documentos.
- *Text Mining*: extracción automática de conocimientos del texto.

¡Ambos están estrechamente relacionados con la NLP!

Tokens y Tipos

Tokenización: la tarea de dividir una oración o documento en partes llamadas emph tokens.
Se pueden emplear transformaciones adicionales como la eliminación de caracteres especiales (por ejemplo, puntuación), minúsculas, etc.
[Manning et al., 2008].

Example

Input: I like human languages and programming languages.
Tokens: [I] [like] [human] [languages] [and] [programming] [languages]

Tipos

- Un *tipo* es una clase de *token* que contiene una única secuencia de caracteres.
- Se obtienen identificando tokens únicos dentro del documento.

Tipos de la sentencia pasada: [I] [like] [human] [languages] [and] [programming]

El token *languages* se repitió en esta sentencia.

Extracción de Vocabulario

- Un *termino* es un *tipo* normalizado.
- La normalización es el proceso de crear clases de equivalencia de diferentes emph tipos. Esto quedará claro en las siguientes diapositivas.
- El vocabulario V, es el conjunto de términos (tokens únicos normalizados) dentro de una colección de documentos o corpus D.

Eliminación de Stopwords

- Para reducir el tamaño del vocabulario y eliminar los términos que no aportan mucha información, se eliminan los términos que aparecen con alta frecuencia en el corpus.
- Estos términos se llaman *stopwords* e incluyen artículos, pronombres, preposiciones y conjunciones.
  Ejemplo: [a, an, and, any, has, do, don't, did, the, on].

¡La eliminación de stopwords puede resultar inconveniente en muchas tareas de NLP!
Ejemplo: I don't like pizza $=>$ pizza ("I", "don't", y "like" se eliminan)

Stemming

Un proceso de normalización de términos en el que los términos se transforman a su raíz para reducir el tamaño del vocabulario. Se lleva a cabo aplicando reglas de reducción de palabras.
Ejemplo: algoritmo de Porter.

(F)	**Rule**			**Example**		
	SSES	→	SS	caresses	→	caress
	IES	→	I	ponies	→	poni
	SS	→	SS	caress	→	caress
	S	→		cats	→	cat

Ejemplo: $d =$ I like human languages and programming languages $=>$ I like human languag and program languag [1]
El vocabulario del documento d después de eliminar stopwords y aplicar el steamming:

termId	value
t1	human
t2	languag
t3	program

[1] `http://9ol.es/porter_js_demo.html`

Lematización

- Otra estrategia de normalización de términos.
- También transforma las palabras en sus raíces.
- Realiza un análisis morfológico utilizando diccionarios de referencia (tablas de búsqueda) para crear clases de equivalencia entre *tipos*.
- Por ejemplo, para el token *studies*, una regla de stemming devolvería el término emph studi, mientras que a través de la lematización obtendríamos el término *study*[2].

[2] `https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/`

Regla de Zipf [1]

- La ley de Zipf, propuesta por *George Kingsley Zipf* en [Zipf, 1935], es una ley empírica sobre la frecuencia de términos dentro de una colección de documentos (**corpus**).
- Establece que la frecuencia f de un término en un corpus es inversamente proporcional a su clasificación r en una tabla de frecuencias ordenada:

$$f = \frac{cf}{r^{\beta}} \tag{1}$$

- Donde cf es una constante dependiente de la colección y $\beta > 0$ es un factor de desintegración.
- Si $\beta = 1$, entonces f sigue exactamente la ley de Zipf; de lo contrario, sigue una distribución similar a Zipf.
- La ley se relaciona con el principio de esfuerzo mínimo. A menudo usamos algunas palabras para escribir ideas.
- La ley Zipf es un tipo de distribución de la ley de potencia (distribuciones de cola larga)

Regla de Zipf [2]

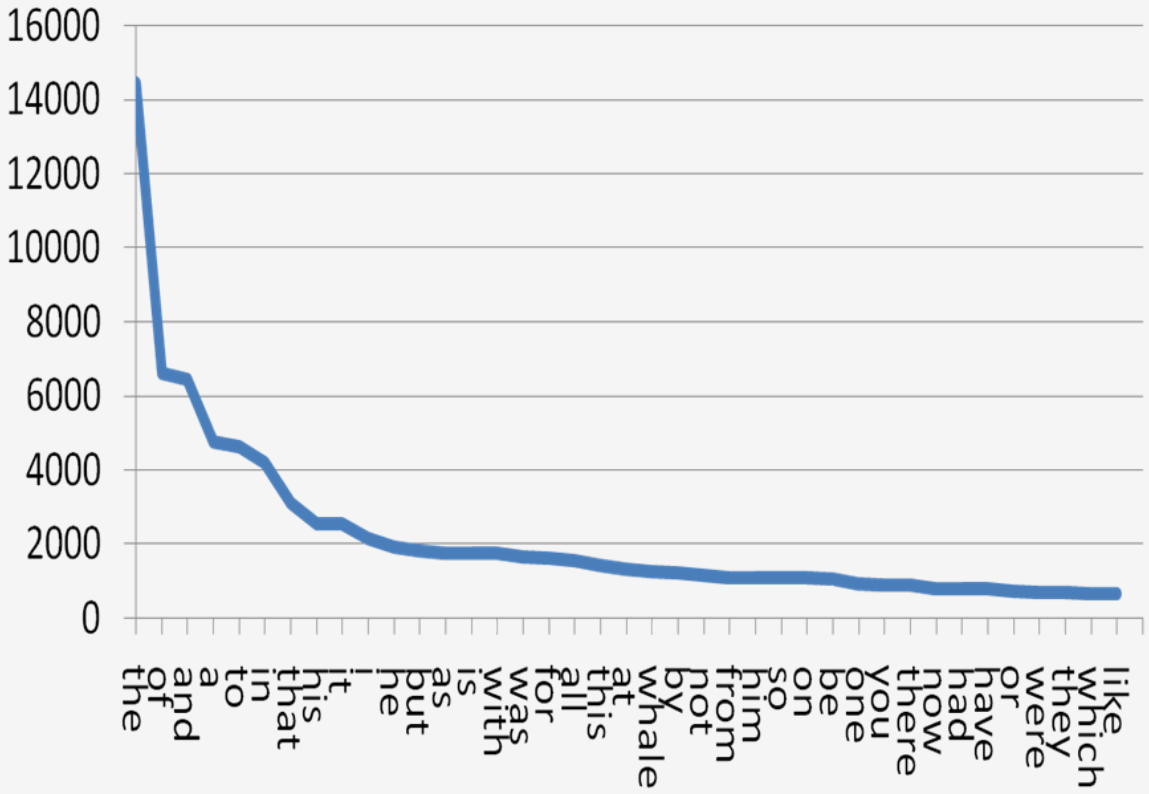


Figura 1: Zipf's law

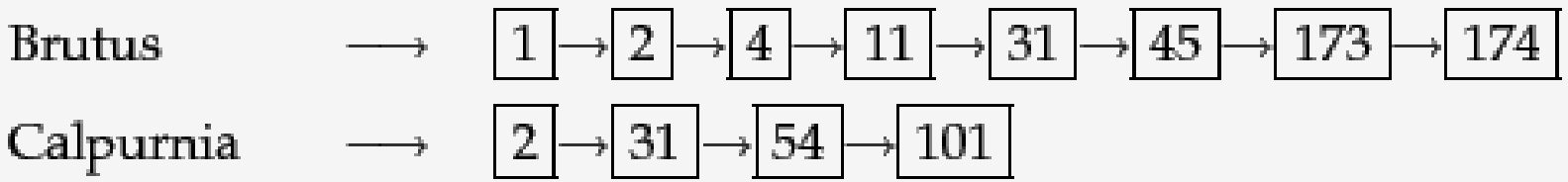
- Si trazamos un gráfico de log - log, obtenemos una línea recta con pendiente $-\beta$.
- Enumerar las palabras más frecuentes de un corpus se puede utilizar para crear una lista de *stopwords*.

Publicación de listas e índice invertido

Sea D una colección de documentos y V el vocabulario de todos los términos extraídos de la colección:

- La lista de publicación de un término es la lista de todos los documentos donde el término aparece al menos una vez. Los documentos se identifican por sus identificadores.
- Un índice invertido es una estructura de datos de tipo diccionario que mapea términos $t_i \in V$ en sus listas de publicación correspondientes.

$$< term > \rightarrow < docId >^*$$



Motores de búsqueda web [1]

Un motor de búsqueda es un sistema de recuperación de información diseñado para buscar información en la Web (resolver necesidades de información) [Manning et al., 2008]. Sus componentes básicos son:

- Crawler: un robot que navega por la Web de acuerdo con una estrategia definida. Por lo general, comienza navegando por un conjunto de sitios web semilla y continúa navegando por sus hipervínculos.
- Indexador: encargado de mantener un índice invertido con el contenido de las páginas recorridas por el Crawler.
- Procesador de Query: encargado de procesar las consultas de los usuarios y buscar en el índice los documentos más relevantes para una consulta.
- Funcjóm de ranking: la función utilizada por el procesador de consultas para clasificar los documentos indexados en la colección por relevancia de acuerdo con una consulta.
- Interfaz de Usuario: recibe la consulta como entrada y devuelve los documentos clasificados por relevancia.

Motores de búsqueda web [2]

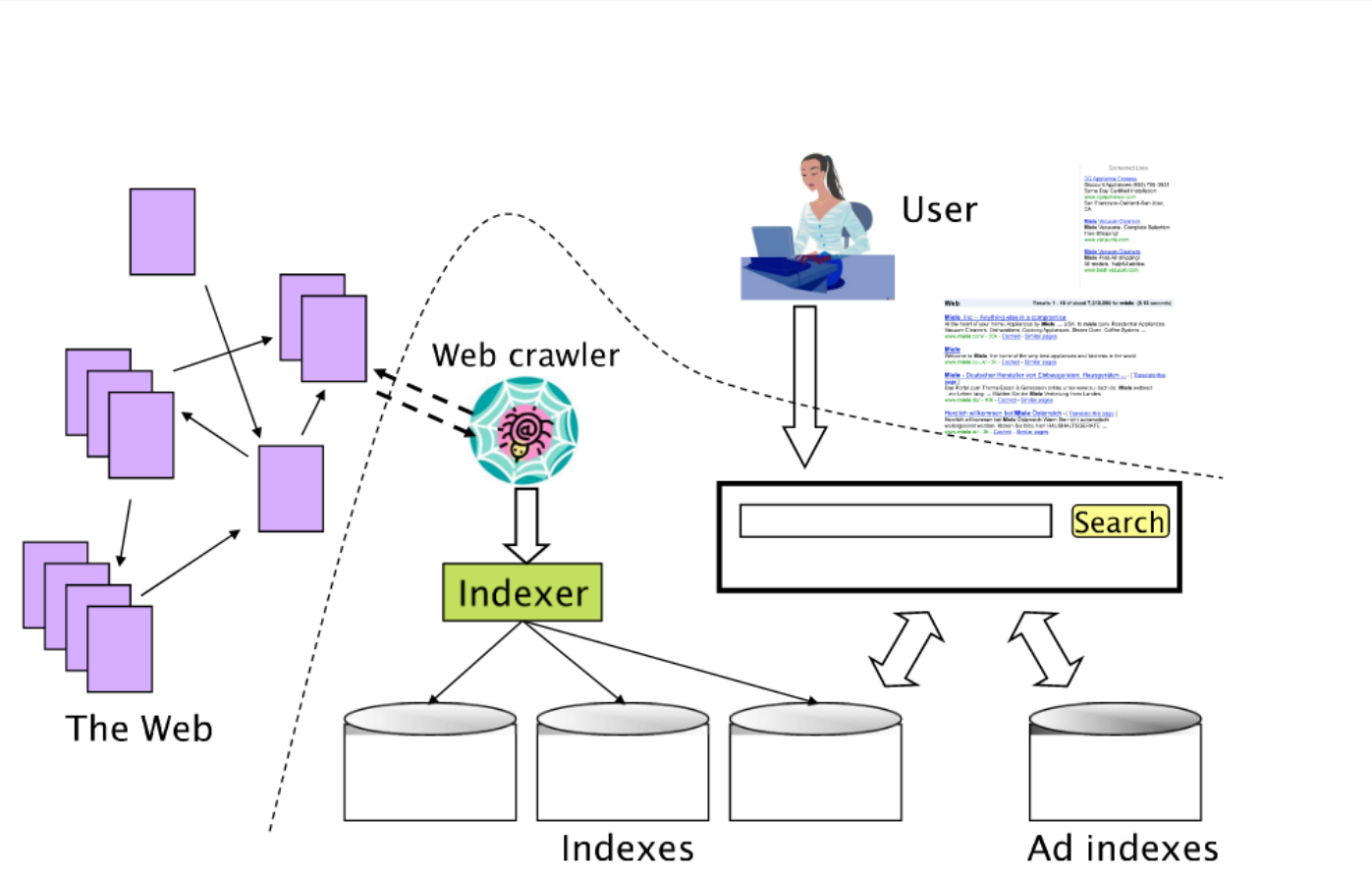


Figura 2: Los diversos componentes de un motor de búsqueda web [Manning et al., 2008].

El modelo de espacio vectorial

- Para clasificar las consultas o medir la similitud entre dos documentos, necesitamos una métrica de similitud.
- Los documentos se pueden *representar* como vectores de términos, donde cada término es una dimensión vectorial [Salton et al., 1975].
- Los documentos con diferentes palabras y longitudes residirán en el mismo espacio vectorial.
- Estos tipos de representaciones se denominan *Bolsa de palabras*.
- En las representaciones de bolsa de palabras se pierde el orden de las palabras y la estructura lingüística de una oración.
- El valor de cada dimensión es un peso que representa la relevancia del término t_i en el documento d.

$$d_j \rightarrow \overrightarrow{d_j} = (w(t_1, d_j), ..., w(t_{|V|}, d_j)) \tag{2}$$

- ¿Cómo podemos modelar qué tan informativo es un término para un documento?

Término Frecuencia - Frecuencia inversa del documento [1]

- Sea $tf_{i,j}$ la frecuencia del término t_i en el documento d_j.
- Un término que ocurre 10 veces debería proporcionar más información que uno que ocurre una vez.
- ¿Qué pasa cuando tenemos documentos que son mucho más largos que los demás?
- Podemos normalizar por la frecuencia máxima de términos en el documento.

$$ntf_{i,j} = \frac{tf_{i,j}}{\text{máx}_i(tf_{i,j})}$$

- ¿Un término que aparece en muy pocos documentos proporciona más o menos información que uno que aparece varias veces?
- Por ejemplo, el documento *The respected major of Pelotillehue*. El término *Pelotillehue* aparece en menos documentos que el término *major*, por lo que debería ser más descriptivo.

Término Frecuencia - Frecuencia inversa del documento [2]

- Sea N el número de documentos de la colección y n_i el número de documentos que contienen el término t_i, definimos idf_{t_i} de la siguiente manera:

$$idf_{t_i} = log_{10}(\frac{N}{n_i})$$

- Un término que aparece en todos los documentos tendría $idf = 0$ y uno que aparece en $10\,\%$ de los documentos tendría $idf = 1$.
- El modelo de puntuación tf - idf combina las puntuaciones tf e idf, lo que da como resultado los siguientes pesos w para un término en un documento:

$$w(t_i, d_j) = tf_i \times log_{10}(\frac{N}{n_i})$$

- Las consultas del motor de búsqueda también se pueden modelar como vectores. Sin embargo, estas consultas tienen términos de entre 2 y 3 aproximadamente, lo que ocasionan muchos ceros en el vector. Para evitar tener demasiadas dimensiones nulas, los vectores de consulta se pueden suavizar de la siguiente manera:

$$w(t_i, d_j) = (0.5 + 0.5 \times tf_{i,j}) log_{10}(\frac{N}{n_i})$$

Similitud entre vectores

- Representar consultas y documentos como vectores permite calcular su similitud.
- Un enfoque sería utilizar la distancia euclidiana.
- El enfoque común es calcular el coseno del ángulo entre los dos vectores.
- Si ambos documentos son iguales, el ángulo sería 0 y su coseno sería 1. Por otro lado, si son ortogonales, el coseno es 0.
- La similitud del coseno se calcula de la siguiente manera:

$$cos(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1} \cdot \vec{d_2}}{|\vec{d_1}| \times |\vec{d_2}|} = \frac{\sum_{i=1}^{|V|}(w(t_i, d_1) \times w(t_i, d_2))}{\sqrt{\sum_{i=1}^{|V|} w(t_i, d_1)^2} \times \sqrt{\sum_{i=1}^{|V|} w(t_i, d_2)^2}}$$

- Esto se llama erróneamente *distancia del coseno*. En realidad, es una métrica de similitud.
- Observe que la similitud de coseno normaliza los vectores por su norma euclidiana $||\vec{d}||_2$.

Cosine Similarity

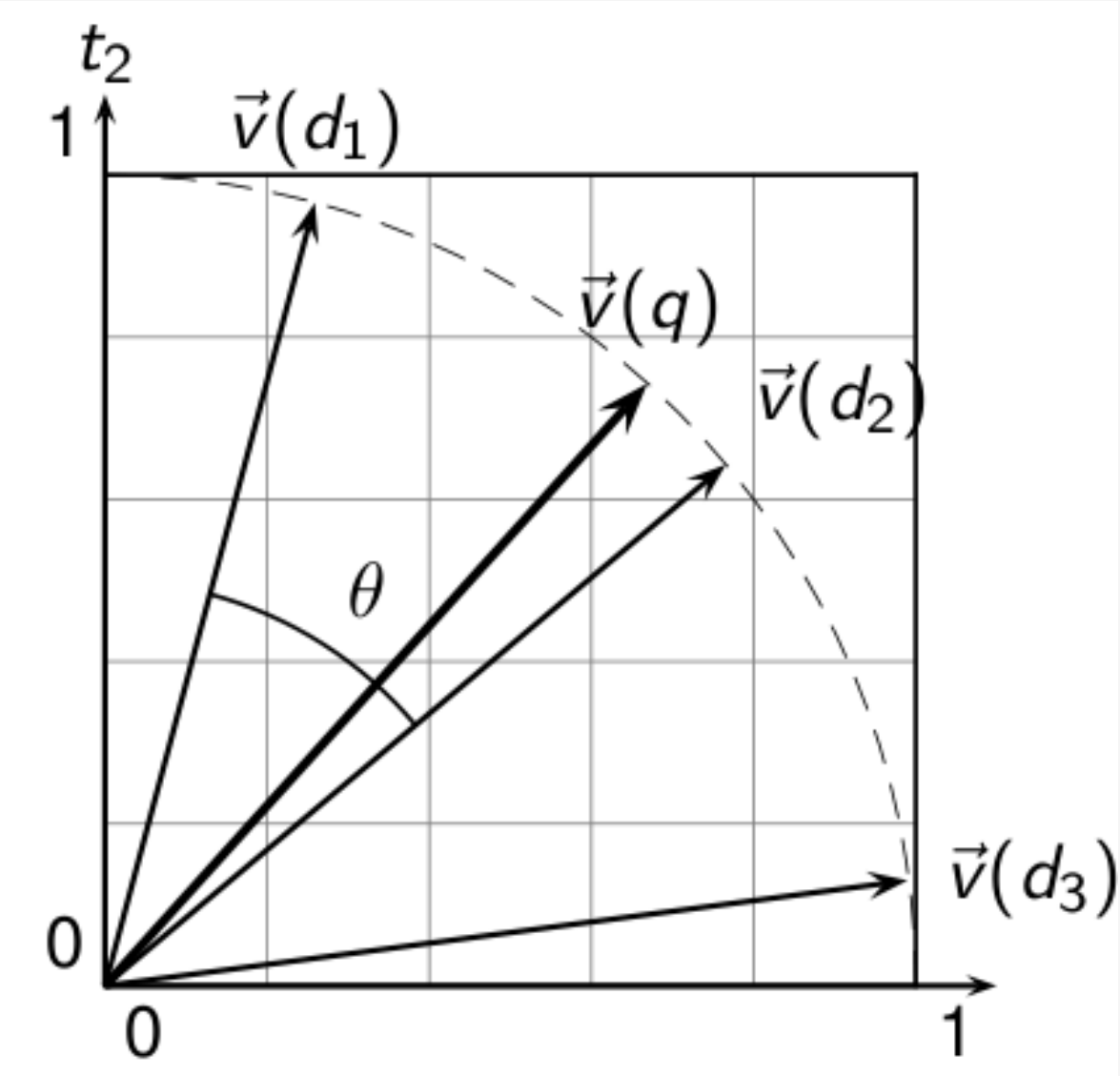


Figura 3: Similaridad del Coseno.

Agrupación de documentos [1]

- ¿Cómo podemos agrupar documentos que son similares entre sí?
- Cada grupo de documentos se denomina *clúster*.
- En la agrupación tratamos de identificar grupos de documentos en los que se maximiza la similitud entre documentos en el mismo grupo y se minimiza la similitud de documentos en diferentes grupos.

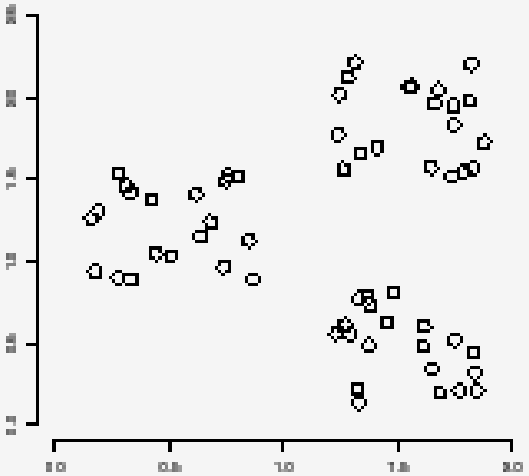


Figura 4: Conjunto de documentos donde se pueden identificar claramente los clusters.

Agrupación de documentos [2]

- La agrupación de documentos permite identificar temas en un corpus y reducir el espacio de búsqueda en un motor de búsqueda, es decir, el índice invertido se organiza según los grupos.
- K-means es un algoritmo de agrupamiento simple que recibe el número de grupos k como parámetro.
- El algoritmo se basa en la idea de *centroide*, que es el vector promedio de documentos que pertenecen al mismo grupo.
- Sea S un conjunto de vectores de 2 dimensiones: $\{3, 6\}, \{1, 2\}, \{5, 1\}$, el centroide de S es $\{(3 + 1 + 5)/3, (6 + 2 + 1)/3\} = \{3, 3\}$.

K-Means

1. Comenzamos con k centroides aleatorios.
2. Calculamos la similitud entre cada documento y cada centroide.
3. Asignamos cada documento a su centroide más cercano formando un grupo.
4. Los centroides se recalculan de acuerdo con los documentos que se les asignan.
5. Este proceso se repite hasta la convergencia.

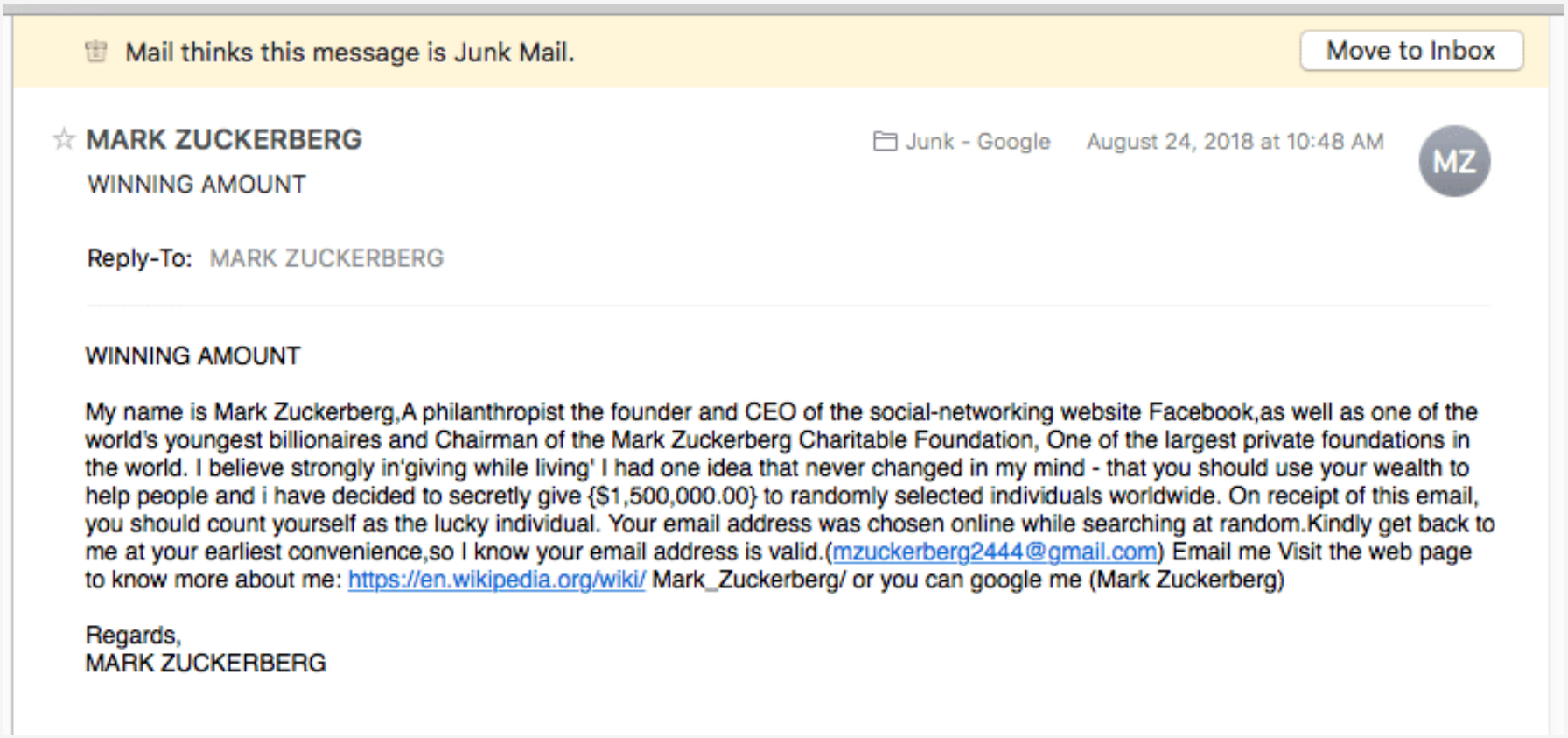
K-means

K-MEANS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)

1 $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
2 **for** $k \leftarrow 1$ **to** K
3 **do** $\vec{\mu}_k \leftarrow \vec{s}_k$
4 **while** stopping criterion has not been met
5 **do for** $k \leftarrow 1$ **to** K
6 **do** $\omega_k \leftarrow \{\}$
7 **for** $n \leftarrow 1$ **to** N
8 **do** $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
9 $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ *(reassignment of vectors)*
10 **for** $k \leftarrow 1$ **to** K
11 **do** $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ *(recomputation of centroids)*
12 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

Figura 5: Algoritmo de K-means

Este correo es SPAM?



Mail thinks this message is Junk Mail. Move to Inbox

☆ **MARK ZUCKERBERG** Junk - Google August 24, 2018 at 10:48 AM MZ

WINNING AMOUNT

Reply-To: MARK ZUCKERBERG

WINNING AMOUNT

My name is Mark Zuckerberg,A philanthropist the founder and CEO of the social-networking website Facebook,as well as one of the world's youngest billionaires and Chairman of the Mark Zuckerberg Charitable Foundation, One of the largest private foundations in the world. I believe strongly in'giving while living' I had one idea that never changed in my mind - that you should use your wealth to help people and i have decided to secretly give {$1,500,000.00} to randomly selected individuals worldwide. On receipt of this email, you should count yourself as the lucky individual. Your email address was chosen online while searching at random.Kindly get back to me at your earliest convenience,so I know your email address is valid.(mzuckerberg2444@gmail.com) Email me Visit the web page to know more about me: https://en.wikipedia.org/wiki/ Mark_Zuckerberg/ or you can google me (Mark Zuckerberg)

Regards,
MARK ZUCKERBERG

Positivo o Negativo? (review de una Pelicula)

- Increiblememente decepcionante
- Lleno de carácteres estrafalarios y sátira ricamente aplicada, y algunos giros de la trama geniales.
- esta es la mejor comedia loca de todos los tiempos.
- Fue patético. La peor parte fueron las escenas de boxeo.

Cuál es el área de investigación de un artículo específico?

- Química
- Biotecnología
- Matemáticas aplicadas
- Física
- Ingeniería

Clasificación de Texto

- Asignar categorías, tópicos, o géneros
- Detección de SPAM
- Identificación de autores
- Identificación de Edad/Sexo
- Identificación del Lenguaje
- Análisis de Sentimientos

Clasificación de Texto: Definición

- Entrada:
  - sea un documento denotado como d
  - Un conjunto fijo de clases $C = \{c_1, c_2, \ldots, c_j\}$
- Salida: Una clase de documento predecido $c \in C$

Métodos de Clasificación de Texto

Reglas de codificación a mano:

- Reglas basadas en combinaciones de palabras o características: spam -> algún email en la lista negra ó ("dólares"Y "han sido seleccionado")
- La precisión puede ser alta (si las reglas son definidas cuidadosamente por algún experto)
- Crear y Mantener estas reglas pueder ser un trabajo muy costoso

Métodos de Clasificación de Texto

Algoritmos Supervizados de Aprendizaje Máquina:

- Entrada:
  - sea un documento denotado como d
  - Un conjunto fijo de clases $C = \{c_1, c_2, \ldots, c_j\}$
  - Un conjunto de entrenamiento de m documentos correctamente etiquetados manualmente $(d_1, c_1), \ldots, (d_m, c_m)$
- Salida: Un clasificador entrenado $\gamma : d \rightarrow c$

Métodos de Clasificación de Texto

Algunos tipos de clasificadores:

- **Logistic Regression**
- **Naïve Bayes**
- Support-Vector machines
- k-Nearest Neighbors

Preprocesamiento con Python

Vayamos a Python!:

- Tokenization
- Stemming
- Lemmatization.
- StopWords.

II.- Naive Bayes para Clasificación de Documentos

Clasificación de Documentos: Naïve Bayes

Intuición:

- Es un método simĺe (Naïve) basado en las reglas de Bayes.
- Se basa en una representación muy simple de un documento: Bag of Words.

Representación de texto mediante Bag of Words (BoW)

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Regla de Bayes aplicado a Documentos y Clases

Para un documento d y una clase c:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

La clase a la que el documento d es más probable a pertenecer:

$$\begin{aligned} C_{MAP} &= \operatorname*{argmax}_{c \in C} P(c|d) \\ &= \operatorname*{argmax}_{c \in C} \frac{P(d|c)P(c)}{P(d)} \\ &= \operatorname*{argmax}_{c \in C} P(d|c)P(c) \end{aligned}$$

- MAP es el máximo posteriori (la clase más cercana)
- Regla de Bayes
- Quitamos el denominador

Regla de Bayes aplicado a Documentos y Clases

$$
\begin{aligned}
C_{MAP} &= \operatorname*{argmax}_{c \in C} P(d|c)P(c) \\
&= \operatorname*{argmax}_{c \in C} P(x_1, x_2, \ldots, x_n|c)P(c)
\end{aligned}
$$

- Un documento d puede ser representado por sus características: $d = \{x_1, \ldots, x_n\}$
- Tenemos $O(|X|^n \cdot |C|)$ parametros
- Podemos estimarlo si tenemos un gran número de ejemplos de entrenamiento disponibles.

Naïve Bayes Multinomial: suposición de independencia

- **Suposición en la bolsa de palabras (BoW):** Suponemos que las posiciones de las palabras no importan.
- **Suposición de independencia condicional:** Asumimos que las probabilidades de las características ($P(x_i|c_j)$) son independientes dados una clase c:

$$P(x_1, x_2, \ldots, x_n|c) \quad = \quad P(x_1|c)(x_2|c)(x_3|c) \cdot \ldots P(x_n|c)\cdot$$

Naïve Bayes Multinomial: suposición de independencia

- **Suposición en la bolsa de palabras (BoW):** Suponemos que las posiciones de las palabras no importan.
- **Suposición de independencia condicional:** Asumimos que las probabilidades de las características ($P(x_i|c_j)$) son independientes dados una clase c:

$$P(x_1, x_2, \ldots, x_n|c) \quad = \quad P(x_1|c)(x_2|c)(x_3|c) \cdot \ldots P(x_n|c)$$

$$\begin{aligned} C_{MAP} &= \operatorname*{argmax}_{c \in C} P(x_1, x_2, \ldots, x_n|c)P(c) \\ C_{MAP} &= \operatorname*{argmax}_{c \in C} P(x_1|c)(x_2|c)(x_3|c) \cdot \ldots P(x_n|c)P(c) \\ C_{NB} &= \operatorname*{argmax}_{c \in C} P(c) \prod_{x \in X} P(x|c) \end{aligned}$$

Aplicanco Naïve Bayes Multinomial a la clasificación de documentos

- **todas las posiciones en el documento de test $\rightarrow$ posiciones**

$$C_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in \textit{posiciones}} P(x_i|c_j)$$

Proceso de Aprendizaje en un modelo de Naïve Bayes Multinomial

- **Primer intento**: Estimar el máximo likelihood usando el cálculo de las frecuencias en los datos:

$$\hat{P}(c_j) = \frac{doccount(C{=}c_j)}{N_{doc}}$$

$$\hat{P}(w_j|c_j) = \frac{count(w_i,c_j)}{\sum_{w \in V} count(w,c_j)}$$

$\hat{P}(w_j|c_j) = \frac{count(w_i,c_j)}{\sum_{w \in V} count(w,c_j)}$ es la fracción de veces que la palabra w_i aparece a lo largo de todas las palabras en el documento con la clase c_j.

Proceso de Aprendizaje en un modelo de Naïve Bayes Multinomial

- **Estimación de Parámetros**: Crear un mega-documento para la clase j concatenando todos los documentos en esta clase:
  - Usar la frecuencia de w en el mega-documento.

Problema con el Máximo Likelihood

- Qué pasa si no tenemos en nuestros datos de entrenamiento una palabra en específico (ejemplo: **"fantastic"**) y queremos clasificar esta palabra en una clase llamada **positive**?

$$\begin{aligned}\hat{P}(\text{"}fantastic\text{"}|positive) &= \frac{count(\text{"}fantastic\text{"},positive)}{\sum_{w \in V} count(w,positive)} \\ &= 0\end{aligned}$$

- Algún $\hat{P}(w_j|c_j)$ con probabilidad cero no se puede condicionar, sin importar las probabilidades de las otras evidencias.

$$C_{MAP} = \underset{c \in C}{\text{argmax}}\, P(c) \prod_i P(x_i|c)$$

Técnica de Laplace-Smoothing para Naïve Bayes

- El suavizado de Laplace es una técnica de suavizado que resuelve el problema de probabilidad cero en Naïve Bayes.

$$\hat{P}(w_j|c_j) = \frac{count(w_i,c_j)+1}{\sum_{w \in V}(count(w,c_j)+1)}$$

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

Modelo de Naïve Bayes Multinomial: Entrenamiento

- De un corpus de entrenamiento, extraer el *Vocabulario*.
- Calcular los términos $P(c_j)$.
  - Para cada $c_j \in C$ hacer:

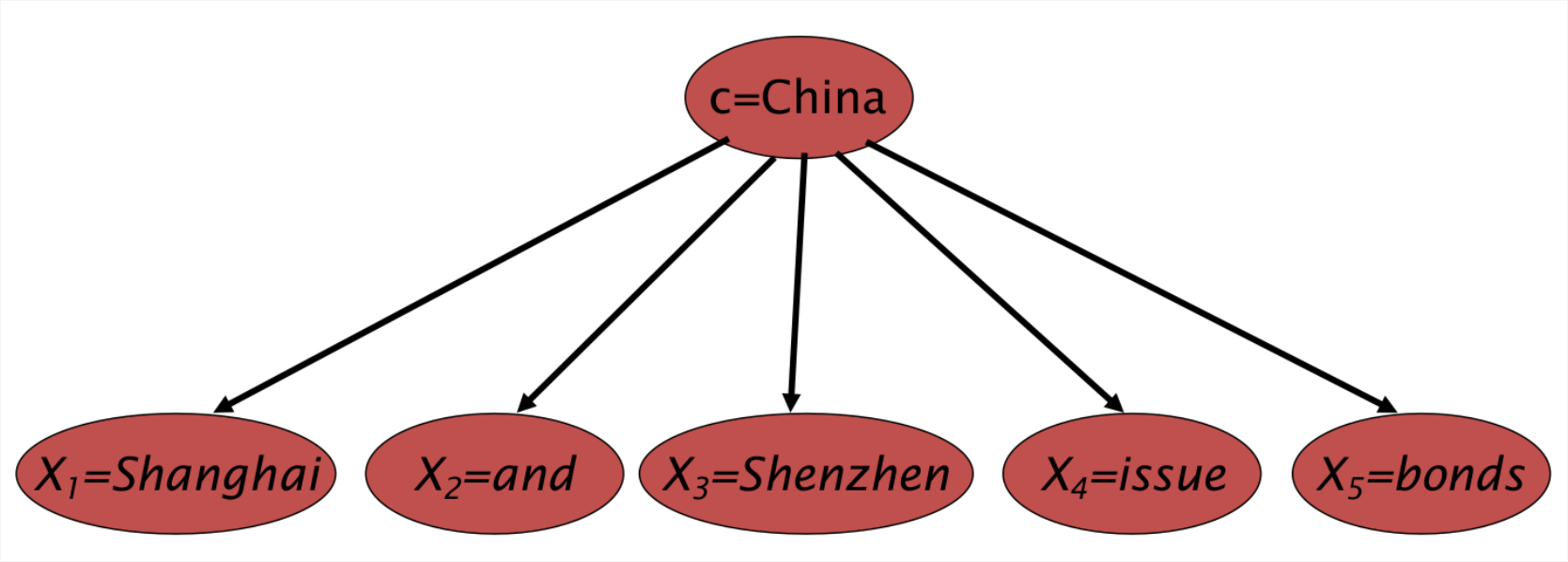
    $docs_j \leftarrow$ todos los documentos con clase c_j

    $P(c_j) \leftarrow \frac{|docs|}{|totaldocumentos|}$
- Calcular los términos $P(w_k|c_j)$
  - $text_j \leftarrow$ un sólo documento que contenga todos los $docs_j$
  - Por cada palabra en w_k en el Vocabulario

    $n_k \leftarrow$ de ocurrencias de w_k en $text_j$

    $P(w_k|c_j) \leftarrow \frac{n_k+\alpha}{n+\alpha|Vocabulary|}$

Generación de Modelo para Naïve Bayes Multinomial



Modelado del Lenguaje con Naïve Bayes Multinomial

- El clasificador Naïve Bayes puede usar cualquier tipo de características ordenadas:
  - URL, emails, diccionarios, características de una red, etc...
- Pero si:
  - Sólo usamos las características de las palabras
  - Usamos todas las palbras en el texto (no un subconjunto)
- Entonces:
  - Naïve bayes tiene una similitud importante con el modelado del lenguaje.

Modelo de Lenguaje: Unigrama

- Asignar a cada palabra $P(word|c)$
- Asignar a cada sentencia: $P(s|c) = \prod P(word|c)$

Class *pos*	
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

I	love	this	fun	film
0.1	0.1	.05	0.01	0.1

P(s | pos) = 0.0000005

Modelo de Lenguaje: Unigrama

- Cuál es la clase a la que pertenece la sentencia s?

Model pos	
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

Model neg	
0.2	I
0.001	love
0.01	this
0.005	fun
0.1	film

I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

P(s|pos) > P(s|neg)

Modelado del Lenguaje con Naïve Bayes Multinomial: Ejemplo

$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Priors:

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

$P(j) = \frac{1}{4}$

Conditional Probabilities:

P(Chinese|*c*) = (5+1) / (8+6) = 6/14 = 3/7

P(Tokyo|*c*) = (0+1) / (8+6) = 1/14

P(Japan|*c*) = (0+1) / (8+6) = 1/14

P(Chinese|*j*) = (1+1) / (3+6) = 2/9

P(Tokyo|*j*) = (1+1) / (3+6) = 2/9

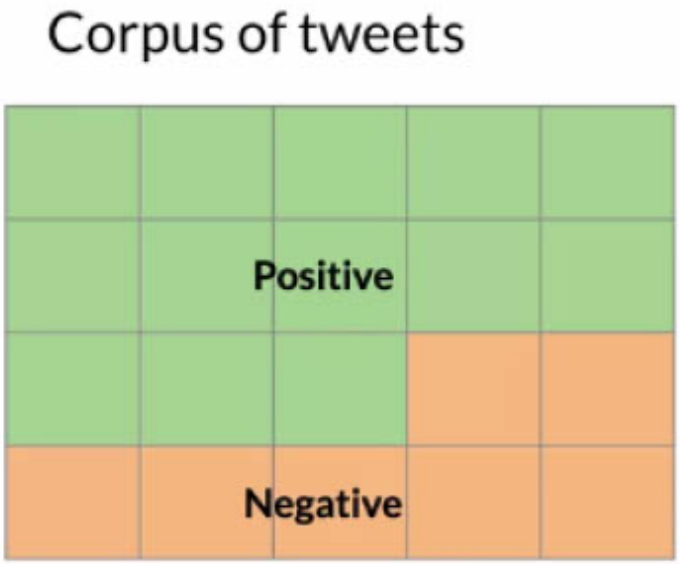
P(Japan|*j*) = (1+1) / (3+6) = 2/9

Choosing a class:

$P(c|d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14 \approx 0.0003$

$P(j|d5) \propto 1/4 * (2/9)^3 * 2/9 * 2/9 \approx 0.0001$

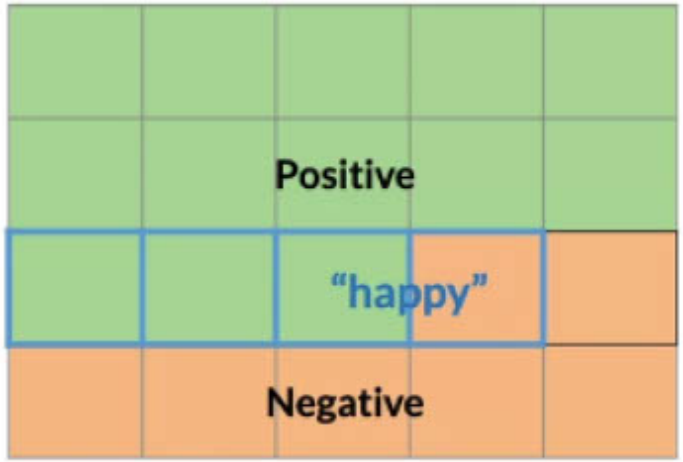
Introduction



Introduction

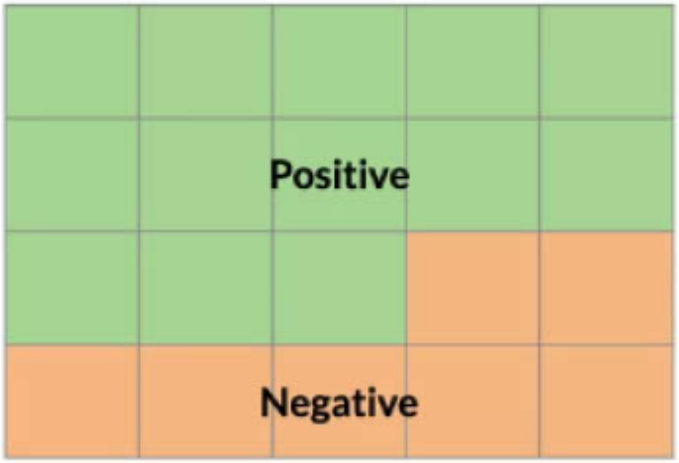
Corpus of tweets

Tweets containing the word "happy"



Probabilities

Corpus of tweets



A → Positive tweet

$P(A) = P(\text{Positive}) = N_{pos} / N$

Probabilities

Corpus of tweets

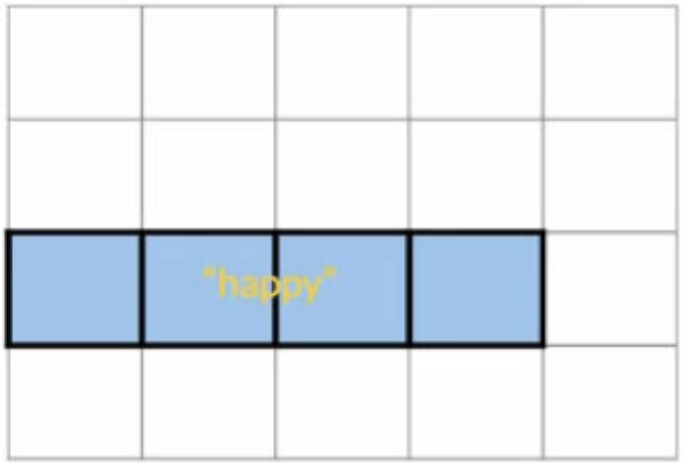
A → Positive tweet

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

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

Probabilities

Tweets containing the word "happy"

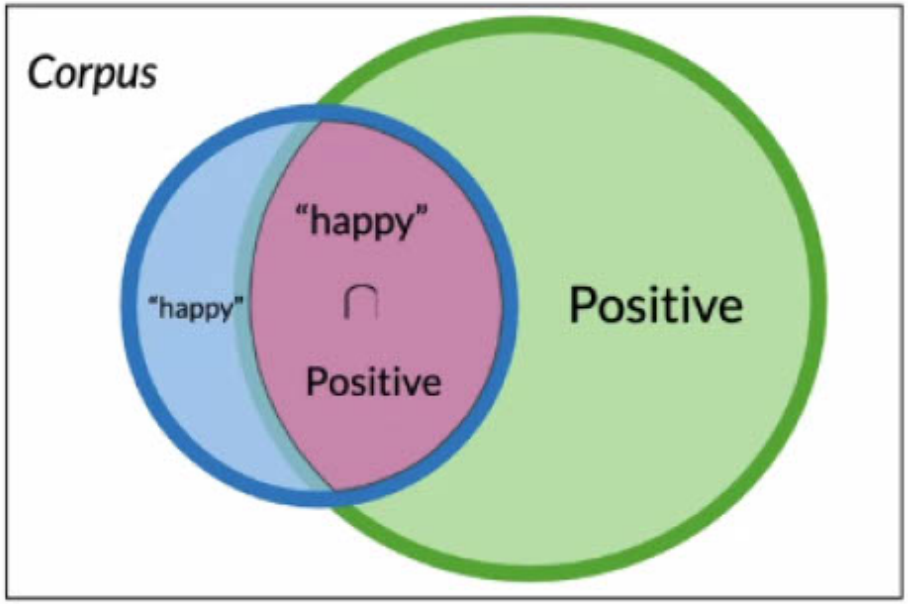


B → tweet contains "happy".

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

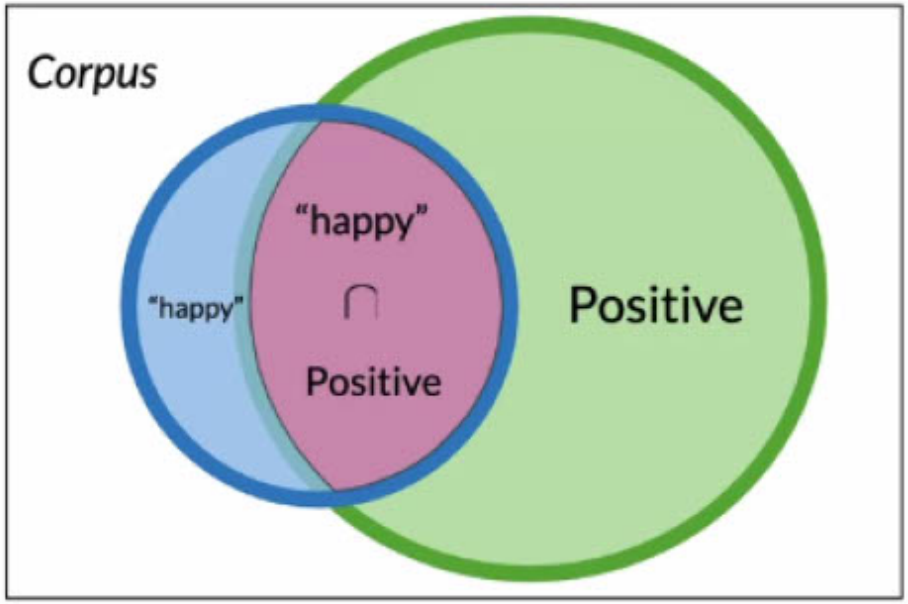
$P(B) = 4 / 20 = 0.2$

Probability of the intersection



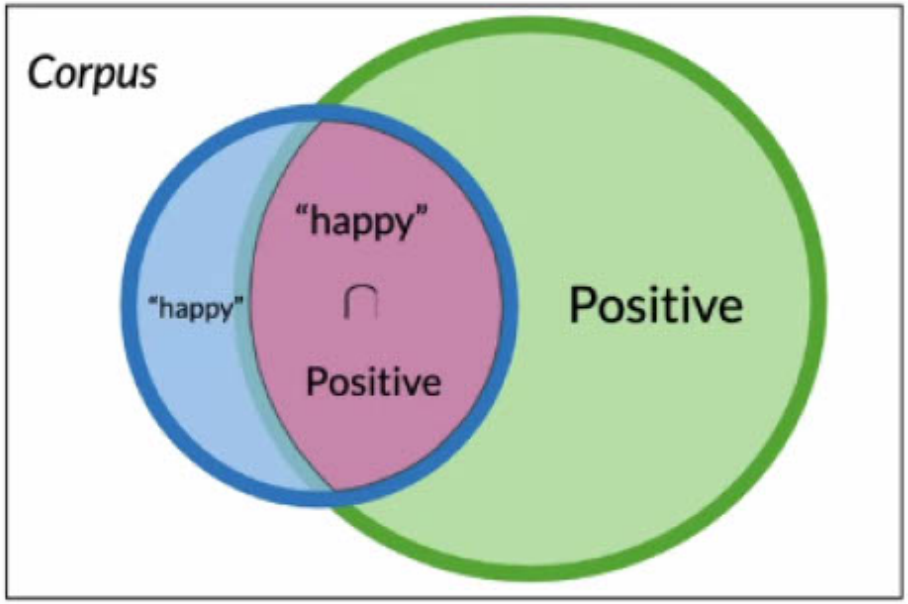
Probability of the intersection

$$P(A \cap B) = P(A, B) =$$



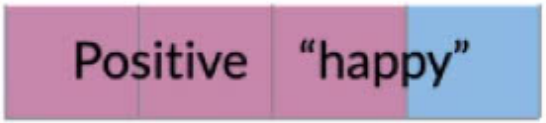
Probability of the intersection

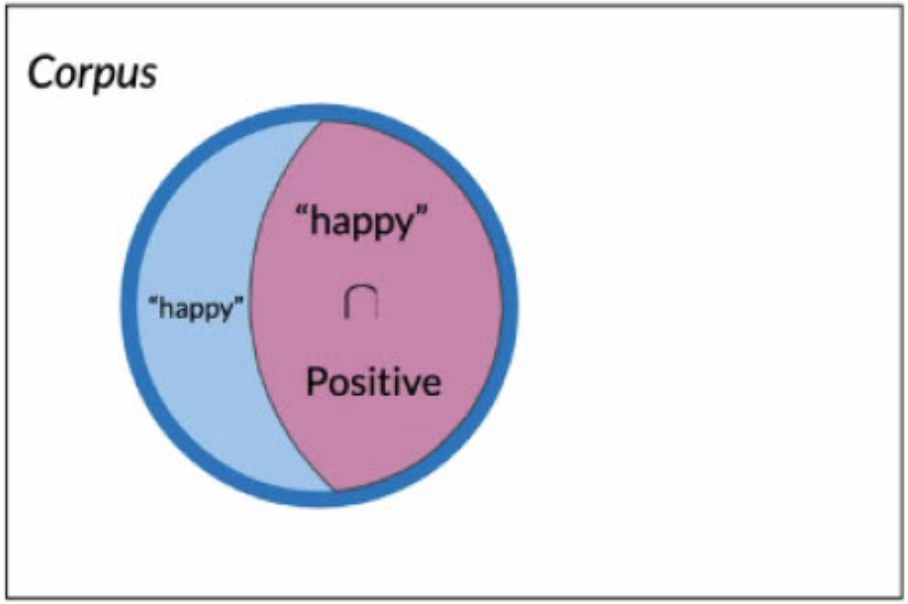
$$P(A \cap B) = P(A, B) = \frac{3}{20} = 0.15$$



Naive Bayes: Bayes Rule

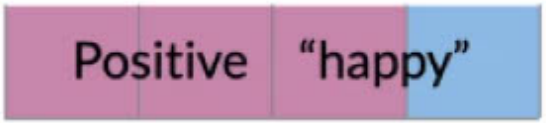
Conditional Probabilities

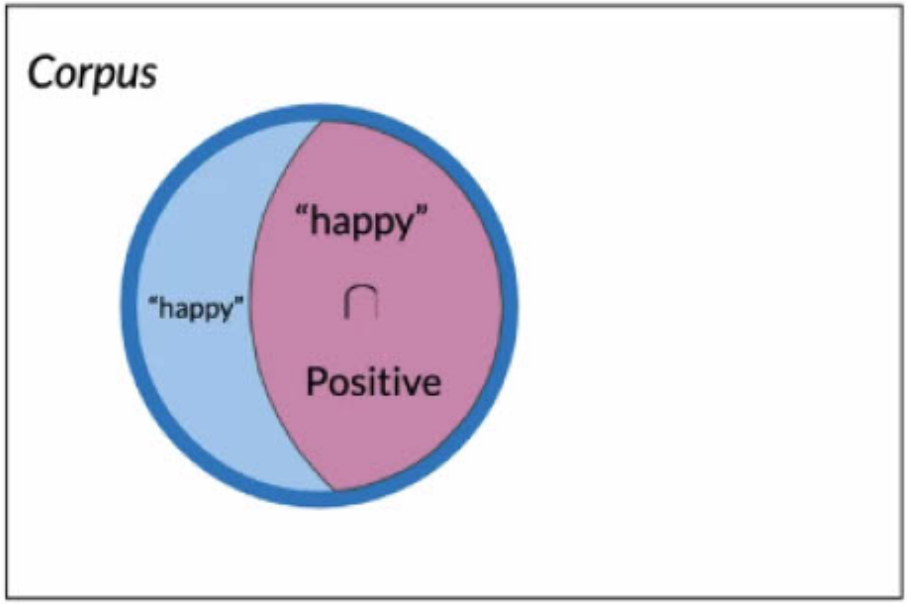




Naive Bayes: Bayes Rule

Conditional Probabilities



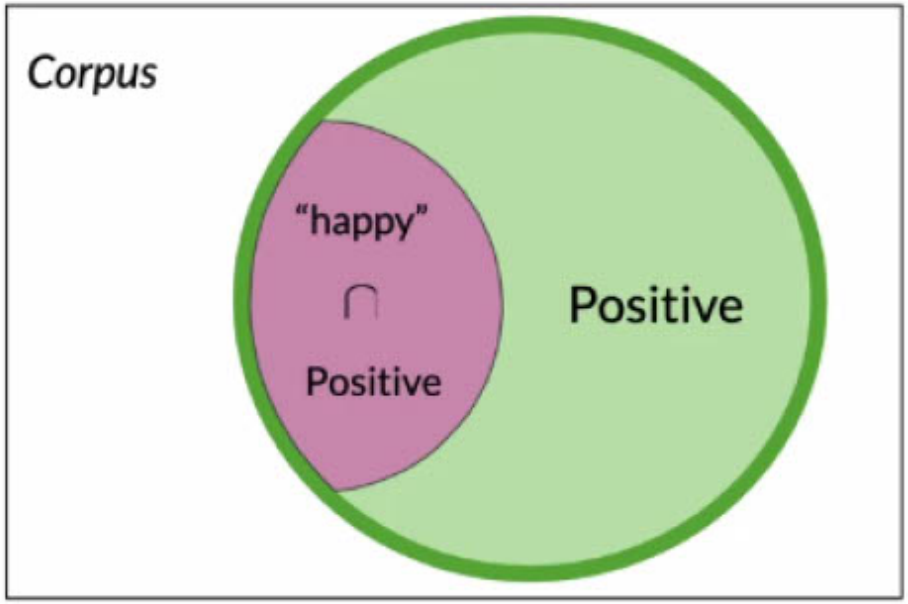


$P(A \mid B) = P(\text{Positive} \mid \text{“happy”})$

$P(A \mid B) = 3 / 4 = 0.75$

Naive Bayes: Bayes Rule

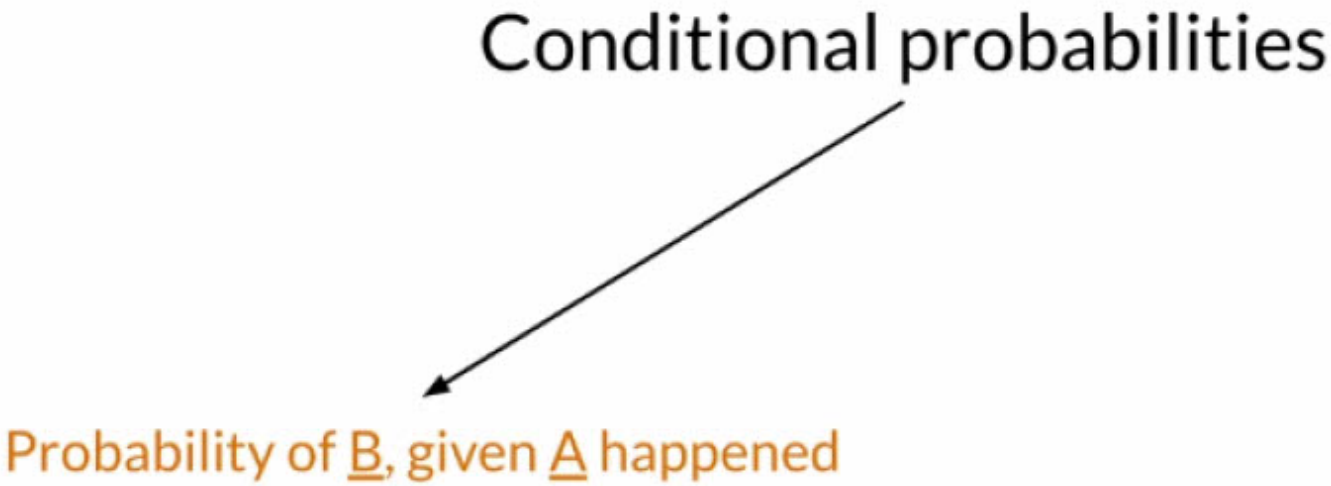
Conditional Probabilities



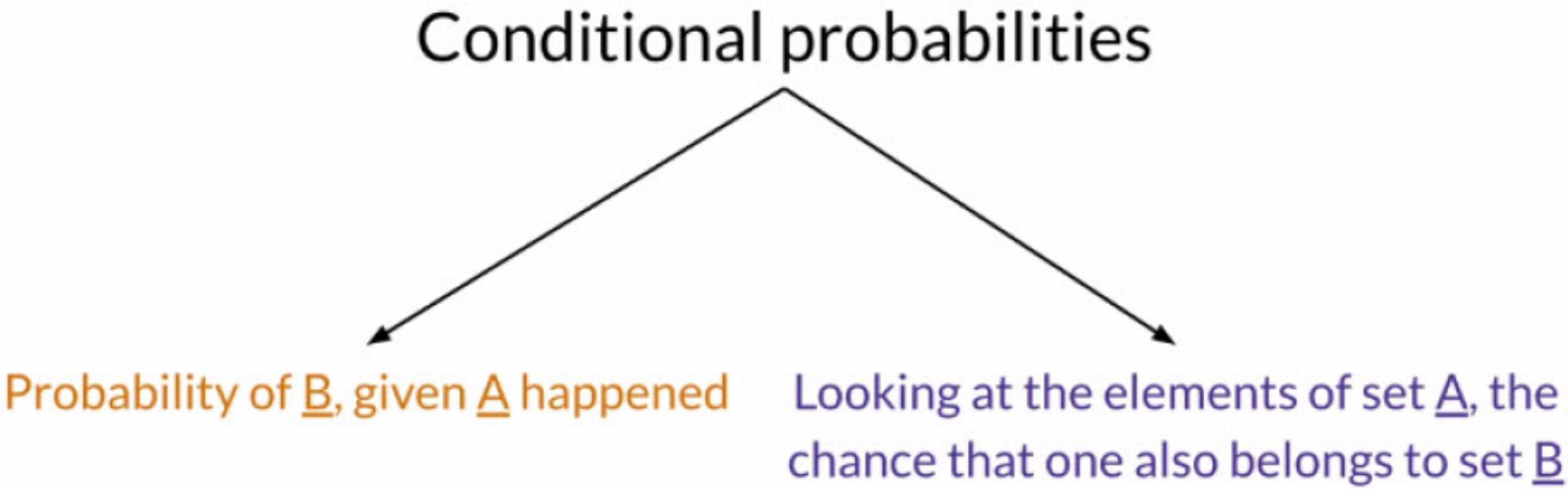
P(B | A) = P(“happy”| Positive)

P(B | A) = 3 / 13 = 0.231

Naive Bayes: Bayes Rule

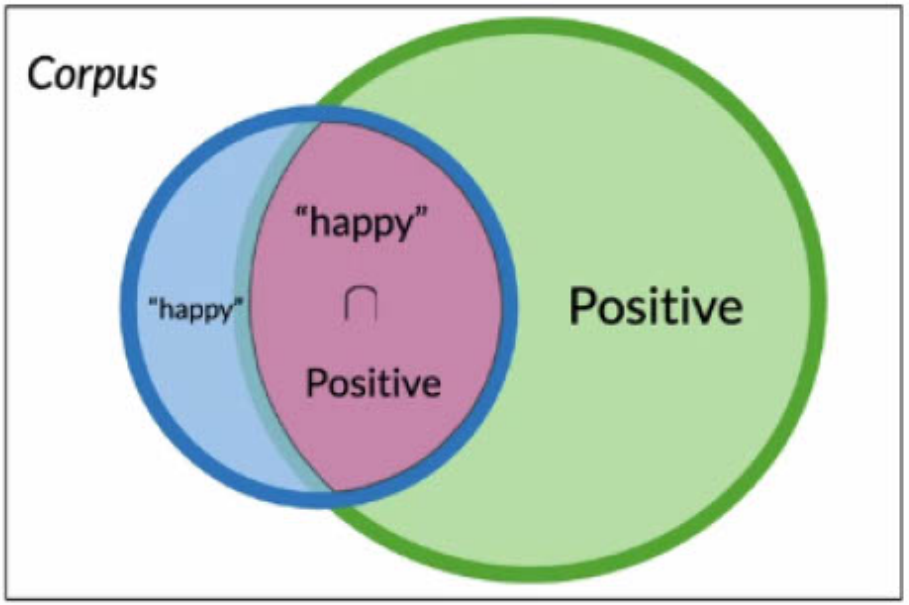


Naive Bayes: Bayes Rule



Naive Bayes: Bayes Rule

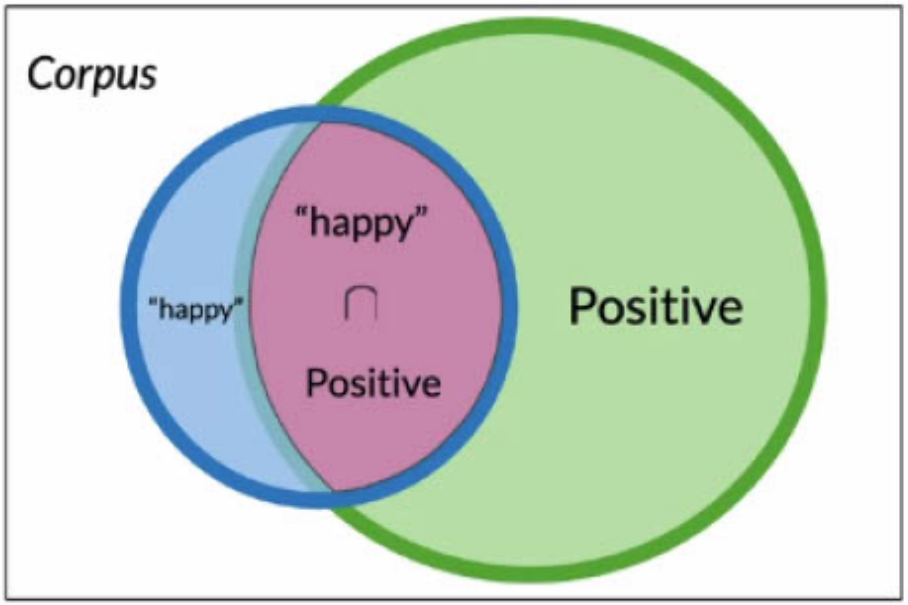
Conditional probabilities



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

Naive Bayes: Bayes Rule

Conditional probabilities

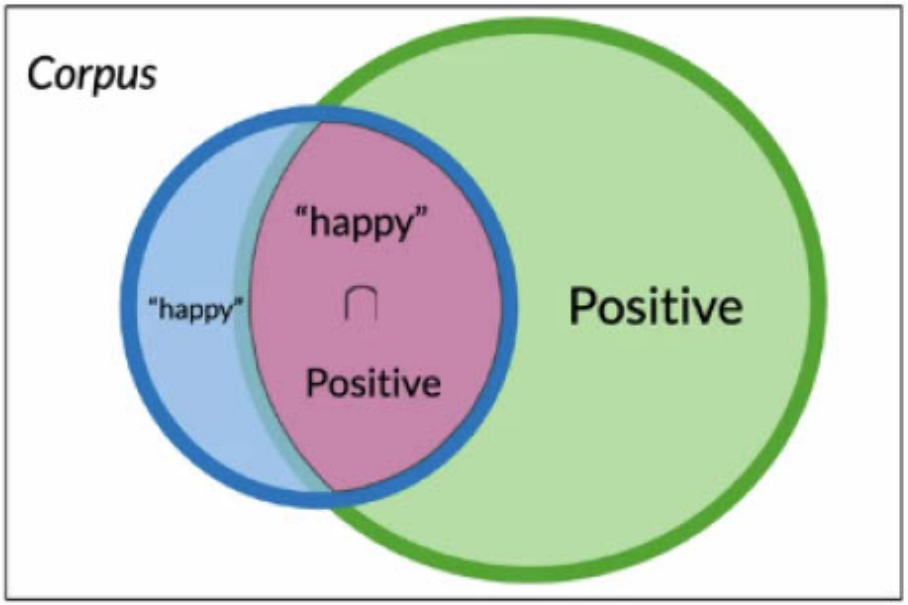


$$P(\text{Positive}|\text{“happy”}) =$$

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

Naive Bayes: Bayes Rule

Conditional probabilities



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

Naive Bayes: Bayes Rule

Bayes' rule

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

Naive Bayes: Bayes Rule

Bayes' rule

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

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

Naive Bayes: Bayes Rule

Bayes' rule

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

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

Naive Bayes: Bayes Rule

Bayes' rule

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

Bayes' rule

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

$$P(X|Y)$$

Bayes' rule

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

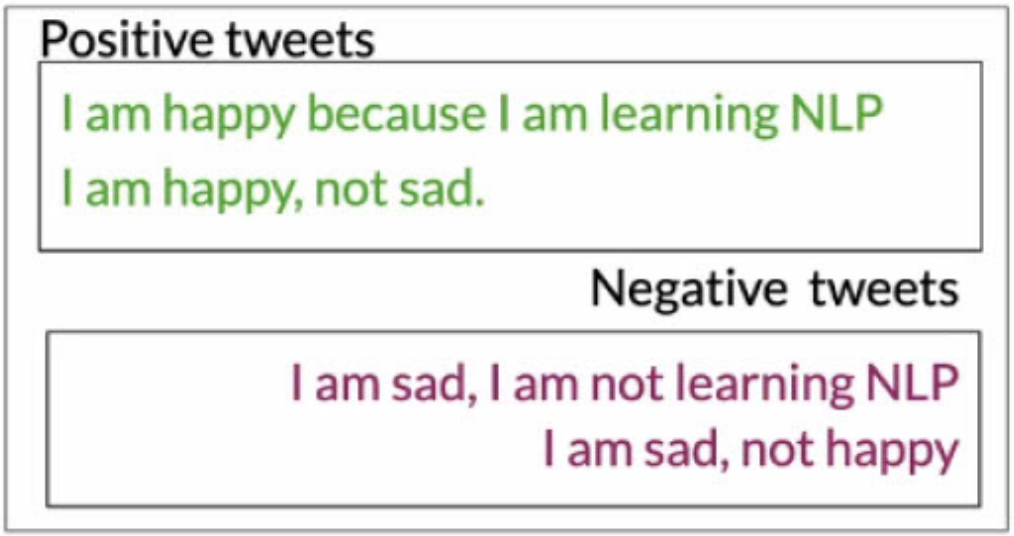
$$\boxed{P(X|Y) = P(Y|X)}$$

Bayes' rule

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

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

Naïve Bayes for Sentiment Analysis



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
I
am
happy
because
learning
NLP
sad
not

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
I	3
am	3
happy	2
because	1
learning	1
NLP	1
sad	1
not	1

Naive Bayes: Summary

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

Naive Bayes: Summary

Naïve Bayes for Sentiment Analysis

Positive tweets

I am happy because I am learning NLP

I am happy, not sad.

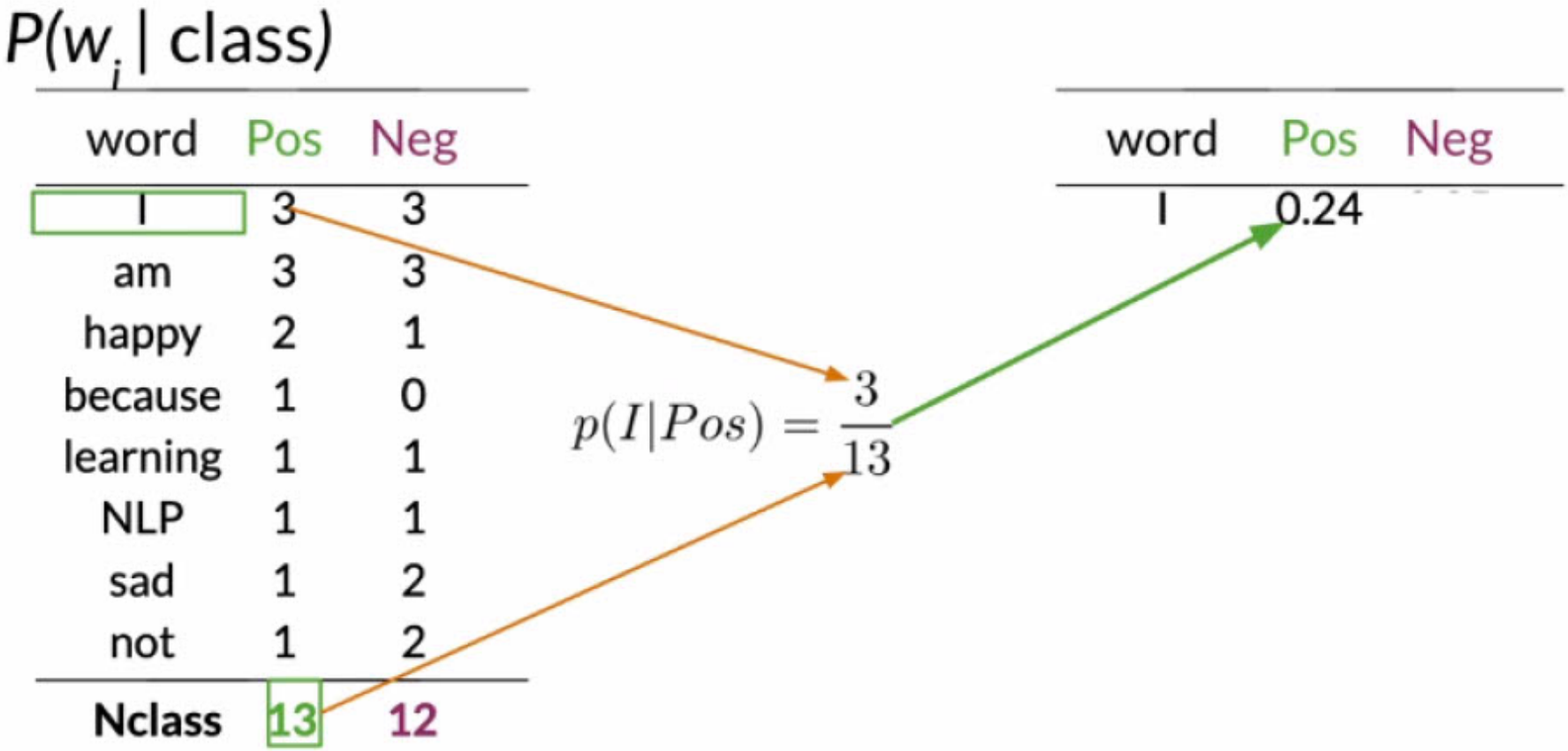
Negative tweets

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
N_{class}	13	12

Naive Bayes: Summary



Naive Bayes: Summary

$P(w_i \mid \text{class})$

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
Nclass	13	12

$$p(I|Pos) = \frac{3}{13}$$

word	Pos	Neg
I	0.24	

Naive Bayes: Summary

$P(w_i \mid \text{class})$

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
Nclass	13	12

$$p(I|Neg) = \frac{3}{12}$$

word	Pos	Neg
I	0.24	0.25

Naive Bayes: Summary

$P(w_i \mid \text{class})$

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
Nclass	**13**	**12**

word	Pos	Neg
I	0.24	0.25
am	0.24	0.25
happy	0.15	0.08
because	0.08	0.00
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.17
not	0.08	0.17

Naive Bayes: Summary

$P(w_i \mid \text{class})$

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
Nclass	**13**	**12**

word	Pos	Neg
I	0.24	0.25
am	0.24	0.25
happy	0.15	0.08
because	0.08	0.00
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.17
not	0.08	0.17
Sum	**1**	**1**

Naive Bayes: Summary

$P(w_i \mid \text{class})$

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

$$\frac{0.20}{0.20}$$

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

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

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

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

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

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

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

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

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

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

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

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

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

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

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

Naïve Bayes

Tweet: I am happy today; I am learning.

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

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

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

Summary

- Naive Bayes inference condition rule for binary classification

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

- Table of probabilities

Laplacian Smoothing

$$P(w_i|class) = \frac{freq(w_i, class)}{N_{class}}$$

class $\in$ {Positive, Negative}

N_{class} = frequency of all words in class

V = number of unique words in vocabulary

Laplacian Smoothing

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

class ∈ {Positive, Negative}

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

N_{class} = frequency of all words in class

V = number of unique words in vocabulary

Introducing $P(w_i \mid \text{class})$ with smoothing

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
Nclass	**13**	**12**

word	Pos	Neg

Introducing $P(w_i \mid \text{class})$ with smoothing

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
Nclass	**13**	**12**

$V = 8$

word	Pos	Neg
I		

Introducing $P(w_i \mid \text{class})$ with smoothing

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
Nclass	13	12

$$P(I|Pos) = \frac{3+1}{13+8}$$

V = 8

word	Pos	Neg
I		

Introducing $P(w_i \mid \text{class})$ with smoothing

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
Nclass	**13**	**12**

$$P(I|Pos) = \frac{3+1}{13+8}$$

V = 8

word	Pos	Neg
I	0.19	

Introducing $P(w_i \mid \text{class})$ with smoothing

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
Nclass	13	12

$$P(I|Neg) = \frac{3+1}{12+8}$$

V = 8

word	Pos	Neg
I	0.19	

Introducing $P(w_i \mid \text{class})$ with smoothing

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
Nclass	13	12

$$P(I|Neg) = \frac{3+1}{12+8}$$

V = 8

word	Pos	Neg
I	0.19	0.20

Introducing $P(w_i \mid \text{class})$ with smoothing

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
Nclass	13	12

$V = 8$

word	Pos	Neg
I	0.19	0.20
am	0.19	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

Introducing $P(w_i \mid \text{class})$ with smoothing

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
Nclass	13	12

$V = 8$

word	Pos	Neg
I	0.19	0.20
am	0.19	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
Sum	1	1

Summary

- Laplacian smoothing to avoid $P(w_i|class) = 0$
- Naïve Bayes formula

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

Ratio of probabilities

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

$$\text{ratio}(w_i) = \frac{P(w_i \mid \text{Pos})}{P(w_i \mid \text{Neg})}$$

Ratio of probabilities

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

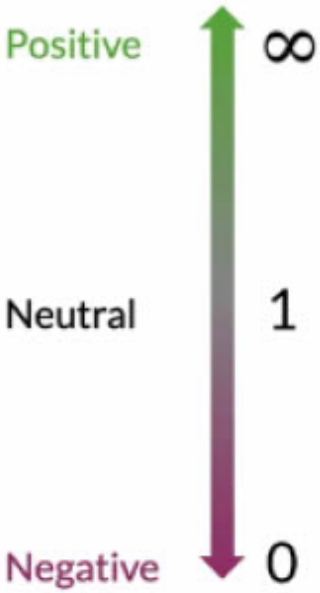
$$\text{ratio}(w_i) = \frac{P(w_i \mid \text{Pos})}{P(w_i \mid \text{Neg})}$$

Ratio of probabilities

word	Pos	Neg	ratio
I	0.20	0.20	1
am	0.20	0.20	1
happy	0.14	0.10	1.4
because	0.10	0.10	1
learning	0.10	0.10	1
NLP	0.10	0.10	1
sad	0.10	0.15	0.6
not	0.10	0.15	0.6

$$\text{ratio}(w_i) = \frac{P(w_i \mid \text{Pos})}{P(w_i \mid \text{Neg})}$$

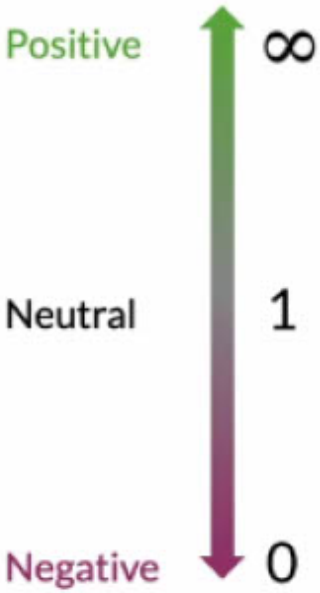
Ratio of probabilities



word	Pos	Neg	ratio
I	0.20	0.20	1
am	0.20	0.20	1
happy	0.14	0.10	1.4
because	0.10	0.10	1
learning	0.10	0.10	1
NLP	0.10	0.10	1
sad	0.10	0.15	0.6
not	0.10	0.15	0.6

$$\text{ratio}(w_i) = \frac{P(w_i \mid \text{Pos})}{P(w_i \mid \text{Neg})}$$

Ratio of probabilities



word	Pos	Neg	ratio
I	0.20	0.20	1
am	0.20	0.20	1
happy	0.14	0.10	1.4
because	0.10	0.10	1
learning	0.10	0.10	1
NLP	0.10	0.10	1
sad	0.10	0.15	0.6
not	0.10	0.15	0.6

$$\text{ratio}(w_i) = \frac{P(w_i \mid \text{Pos})}{P(w_i \mid \text{Neg})}$$

$$\approx \frac{freq(w_i, 1) + 1}{freq(w_i, 0) + 1}$$

Naïve Bayes' inference

class $\in$ *{pos, neg}*
w -> *Set of m words in a tweet*

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

Naïve Bayes' inference

class $\in$ *{pos, neg}*
w -> *Set of m words in a tweet*

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

Naïve Bayes' inference

class $\in$ *{pos, neg}*
w -> *Set of m words in a tweet*

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

- A simple, fast, and powerful baseline
- A probabilistic model used for classification

Log Likelihood

- Products bring risk of underflow

Log Likelihood

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

- Products bring risk of underflow
- $log(a * b) = log(a) + log(b)$

Log Likelihood

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

- Products bring risk of underflow
- $log(a * b) = log(a) + log(b)$
- $log(\frac{P(pos)}{P(neg)} \prod_{i=1}^{n} \frac{P(w_i|pos)}{P(w_i|neg)})$

Log Likelihood

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

- Products bring risk of underflow
- $log(a * b) = log(a) + log(b)$
- $log(\frac{P(pos)}{P(neg)} \prod_{i=1}^{n} \frac{P(w_i|pos)}{P(w_i|neg)}) \Rightarrow log\frac{P(pos)}{P(neg)} + \sum_{i=1}^{n} log\frac{P(w_i|pos)}{P(w_i|neg)}$

log prior + log likelihood

Log Likelihood

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

- Products bring risk of underflow
- $log(a * b) = log(a) + log(b)$
- $log(\frac{P(pos)}{P(neg)} \prod_{i=1}^{n} \frac{P(w_i|pos)}{P(w_i|neg)}) \Rightarrow log\frac{P(pos)}{P(neg)} + \sum_{i=1}^{n} log\frac{P(w_i|pos)}{P(w_i|neg)}$

log prior + log likelihood

Calculating Lambda

tweet: I am happy because I am learning.

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

word	Pos	Neg
I	0.05	0.05
am	0.04	0.04
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

Calculating Lambda

tweet: I am happy because I am learning.

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

$$\lambda(I) = log\frac{0.05}{0.05}$$

word	Pos	Neg	λ
I	0.05	0.05	
am	0.04	0.04	
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	

Calculating Lambda

tweet: I am happy because I am learning.

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

$$\lambda(I) = log\frac{0.05}{0.05} = log(1) = 0$$

word	Pos	Neg	λ
I	0.05	0.05	
am	0.04	0.04	
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	

Calculating Lambda

tweet: I am happy because I am learning.

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

$$\lambda(I) = log\frac{0.05}{0.05} = log(1) = 0$$

word	Pos	Neg	λ
I	0.05	0.05	0
am	0.04	0.04	
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	

Calculating Lambda

tweet: I am happy because I am learning.

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

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

word	Pos	Neg	λ
I	0.05	0.05	0
am	0.04	0.04	
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	

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	
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	

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	

Summing the Lambdas

doc: I am happy because I am learning.

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

$$\lambda(\text{happy}) = log\frac{0.09}{0.01} \approx 2.2$$

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	

Summing the Lambdas

doc: I am happy because I am learning.

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

$$\lambda(\text{happy}) = log\frac{0.09}{0.01} \approx 2.2$$

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	
learning	0.03	0.01	
NLP	0.02	0.02	
sad	0.01	0.09	
not	0.02	0.03	

Summing the Lambdas

doc: I am happy because I am learning.

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

$$\lambda(\text{happy}) = log\frac{0.09}{0.01} \approx 2.2$$

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

Summary

- Word sentiment

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

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

Log Likelihood

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 =

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

Log Likelihood

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

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

Log Likelihood

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

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

Log Likelihood

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

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

Log Likelihood

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

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

Log Likelihood

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

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

Log Likelihood

doc: I am happy because I am learning.

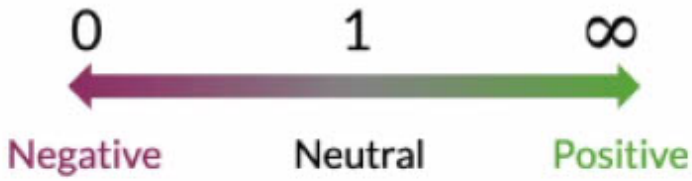
$$\sum_{i=1}^{m} log \frac{P(w_i|pos)}{P(w_i|neg)} = \sum_{i=1}^{m} \lambda(w_i)$$

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

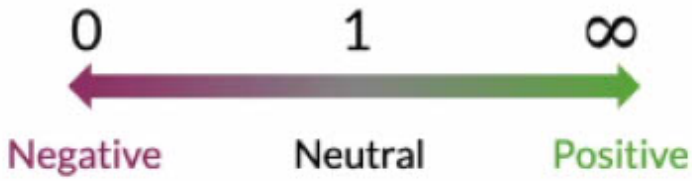
Log Likelihood

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

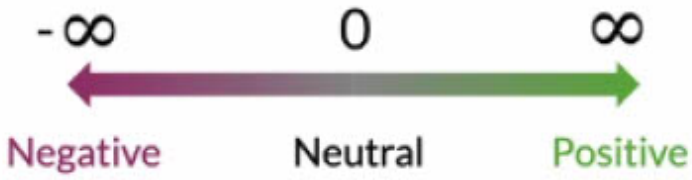


Log Likelihood

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

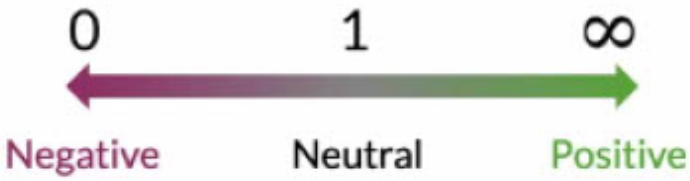


$$\sum_{i=1}^{m} log \frac{P(w_i|pos)}{P(w_i|neg)} > 0$$



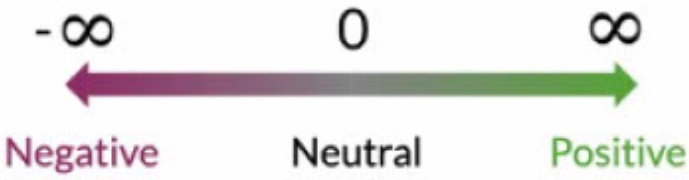
Log Likelihood

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



$$\sum_{i=1}^{m} log \frac{P(w_i|pos)}{P(w_i|neg)} > 0$$

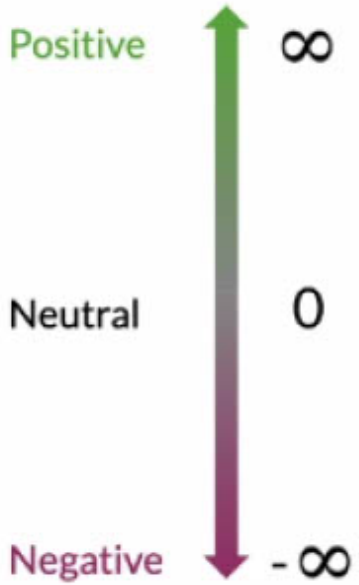
3.3 > 0



Summary

Tweet sentiment:

$$log \prod_{i=1}^{m} ratio(w_i) = \sum_{i=1}^{m} \lambda(w_i) \quad > 0$$



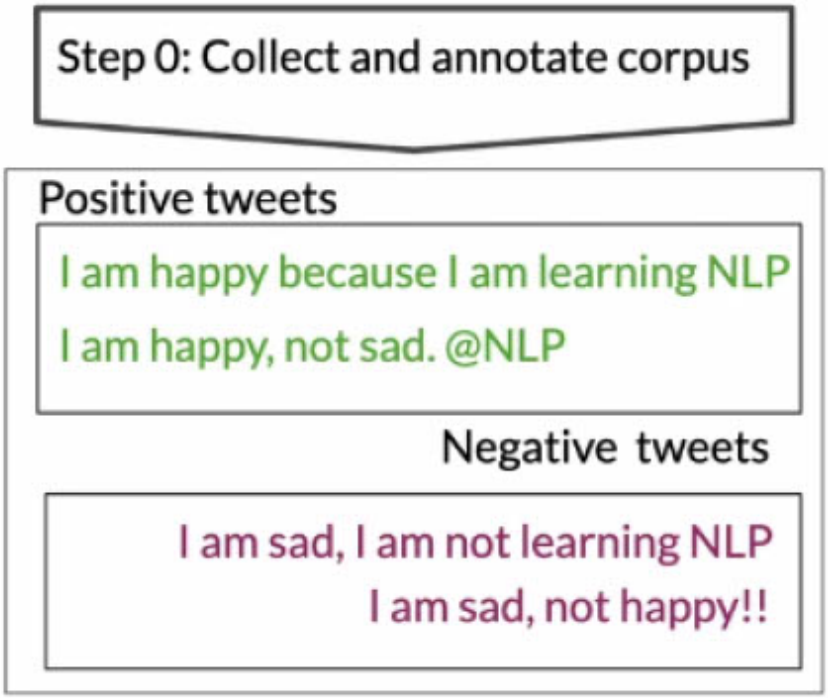
Outline

- Five steps for training a Naïve Bayes model

Training Naïve Bayes

Step 0: Collect and annotate corpus

Training Naïve Bayes



Training Naïve Bayes

Step 0: Collect and annotate corpus

Positive tweets

I am happy because I am learning NLP

I am happy, not sad. @NLP

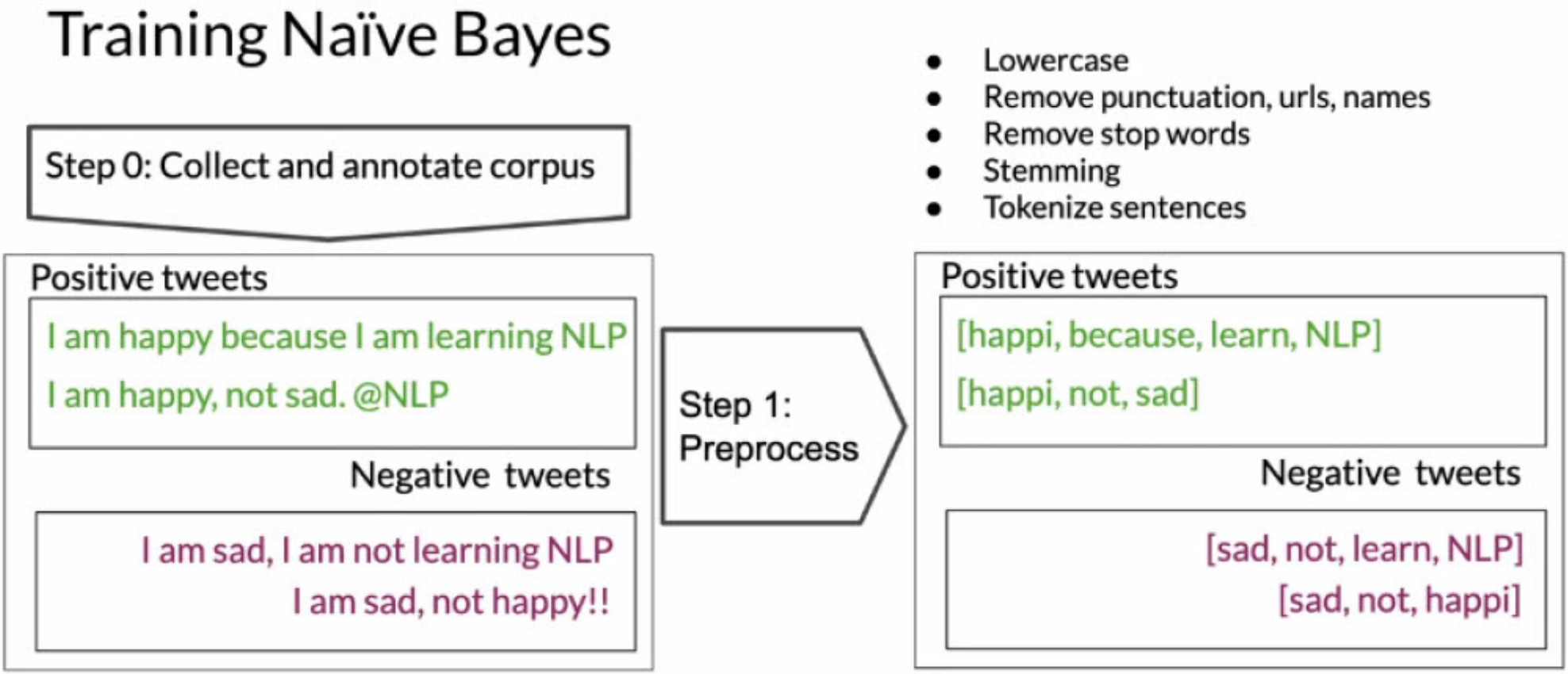
Negative tweets

I am sad, I am not learning NLP

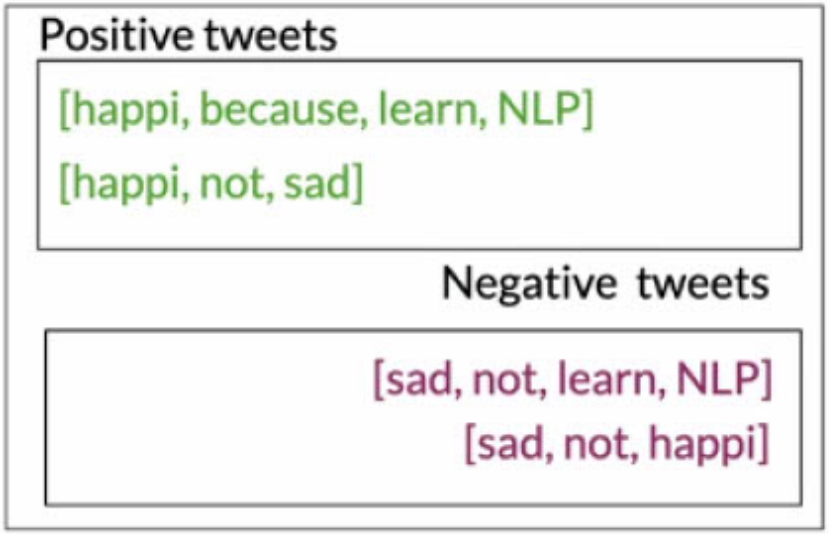
I am sad, not happy!!

Step 1: Preprocess

- Lowercase
- Remove punctuation, urls, names
- Remove stop words
- Stemming
- Tokenize sentences

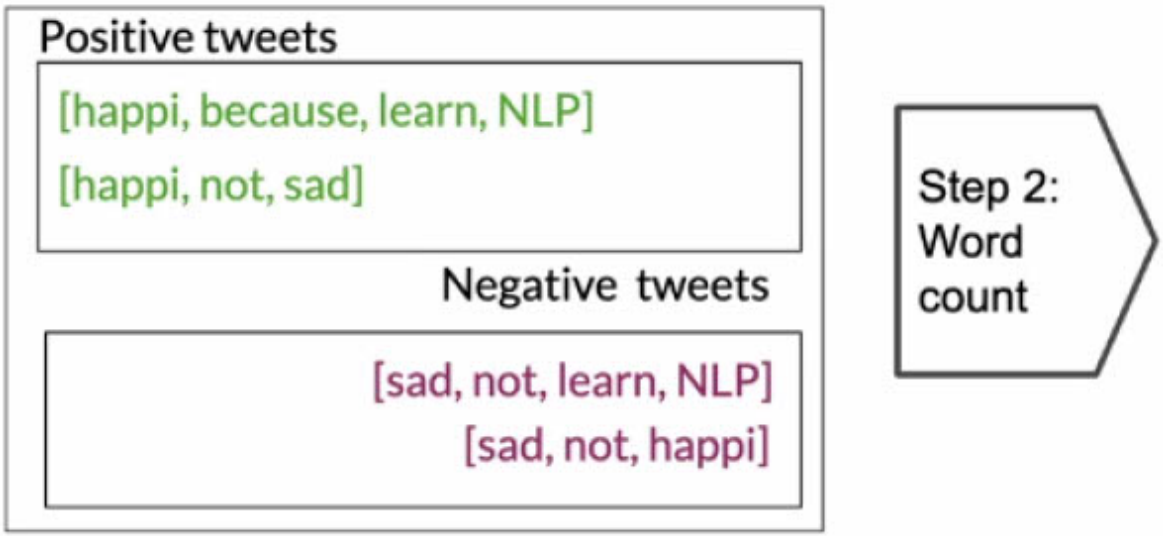

Training Naïve Bayes
Step 0: Collect and annotate corpus
Positive tweets
I am happy because I am learning NLP
I am happy, not sad. @NLP
Negative tweets
I am sad, I am not learning NLP
I am sad, not happy!!
Step 1: Preprocess
Lowercase
Remove punctuation, urls, names
Remove stop words
Stemming
Tokenize sentences
Positive tweets
[happi, because, learn, NLP]
[happi, not, sad]
Negative tweets
[sad, not, learn, NLP]
[sad, not, happi]

Training Naïve Bayes

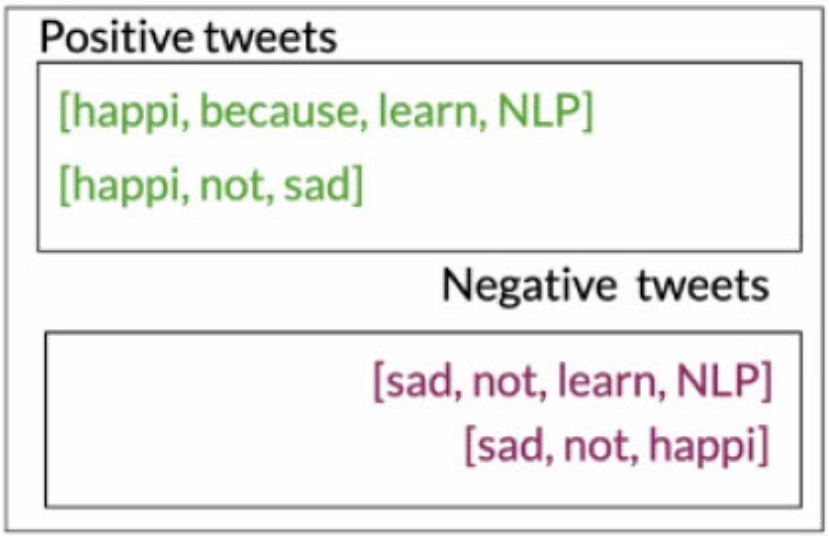


Training Naïve Bayes

freq(w, class)



Training Naïve Bayes



Step 2: Word count

freq(w, class)

word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

Training Naïve Bayes

freq(w, class)

word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

Training Naïve Bayes

freq(w, class)

word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

Step 3:
$P(\mathrm{w}|\mathrm{class})$

$$V_{\mathrm{class}} = 6$$

$$\frac{\mathrm{freq}(\mathrm{w}, \mathrm{class}) + 1}{N_{\mathrm{class}} + V_{\mathrm{class}}}$$

Training Naïve Bayes

freq(w, class)

word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

Step 3: $P(\text{w}|\text{class})$

$$V_{\text{class}} = 6$$

$$\frac{\text{freq}(\text{w}, \text{class}) + 1}{N_{\text{class}} + V_{\text{class}}}$$

word	Pos	Neg
happy	0.23	0.15
because	0.15	0.07
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.17
not	0.08	0.17

Training Naïve Bayes

freq(w, class)

word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

Step 3: $P(\text{w}|\text{class})$

$$V_{\text{class}} = 6$$

$$\frac{\text{freq}(\text{w}, \text{class}) + 1}{N_{\text{class}} + V_{\text{class}}}$$

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

Step 4: Get lambda

word	Pos	Neg
happy	0.23	0.15
because	0.15	0.07
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.17
not	0.08	0.17

Training Naïve Bayes

freq(w, class)

word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

Step 3: $P(\text{w}|\text{class})$

$$V_{\text{class}} = 6$$

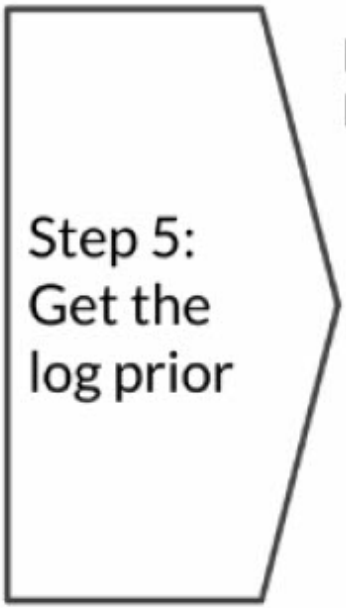
$$\frac{\text{freq}(\text{w}, \text{class}) + 1}{N_{\text{class}} + V_{\text{class}}}$$

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

Step 4: Get lambda

word	Pos	Neg	λ
happy	0.23	0.15	0.43
because	0.15	0.07	0.6
learning	0.08	0.08	0
NLP	0.08	0.08	0
sad	0.08	0.17	-0.75
not	0.08	0.17	-0.75

Training Naïve Bayes



D_{pos} = Number of positive tweets
D_{neg} = Number of negative tweets

$$\text{logprior} = log\frac{D_{pos}}{D_{neg}}$$

Training Naïve Bayes

Step 5: Get the log prior

D_{pos} = Number of positive tweets
D_{neg} = Number of negative tweets

$$\text{logprior} = log\frac{D_{pos}}{D_{neg}}$$

Training Naïve Bayes

Step 5: Get the log prior

D_{pos} = Number of positive tweets
D_{neg} = Number of negative tweets

$$\text{logprior} = log\frac{D_{pos}}{D_{neg}}$$

If dataset is balanced, $D_{pos} = D_{neg}$ and logprior = 0.

Predict using Naïve Bayes

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

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$
- Tweet: I passed the NLP interview.

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$
- Tweet: [I, pass, the, NLP, interview]

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$
- Tweet: [I, pass, the, NLP, interview]

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$
- Tweet: [I, pass, the, NLP, interview]

$$score = -0.01 + 0.5 - 0.01 + 0$$

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$
- Tweet: [I, pass, the, NLP, interview]

$$score = -0.01 + 0.5 - 0.01 + 0 + logprior$$

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$
- Tweet: [I, pass, the, NLP, interview]

$$score = -0.01 + 0.5 - 0.01 + 0 + logprior = 0.48$$

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$
- Tweet: [I, pass, the, NLP, interview]

$$score = -0.01 + 0.5 - 0.01 + 0 + logprior = 0.48$$

$$pred = score > 0$$

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

Predict using Naïve Bayes

- log-likelihood dictionary $\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$
- $logprior = log\frac{D_{pos}}{D_{neg}} = 0$
- Tweet: [I, pass, the, NLP, interview] 

$$score = -0.01 + 0.5 - 0.01 + 0 + logprior = 0.48$$

$$pred = score > 0$$

word	λ
I	-0.01
the	-0.01
happi	0.63
because	0.01
pass	0.5
NLP	0
sad	-0.75
not	-0.75

References I

Manning, C. D., Raghavan, P., and Schütze, H. (2008).
Introduction to Information Retrieval.
Cambridge University Press, New York, NY, USA.

Salton, G., Wong, A., and Yang, C.-S. (1975).
A vector space model for automatic indexing.
Communications of the ACM, 18(11):613–620.

Zipf, G. K. (1935).
The Psychobiology of Language.
Houghton-Mifflin, New York, NY, USA.