

Procesamiento de Lenguaje Natural

Apunte de Clases (Borrador)

Felipe Bravo Márquez

Felipe Bravo Márquez

Ilustración Portada por Paulette Filla

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN, UNIVERSIDAD DE CHILE

GITHUB.COM/DCCUCHILE/CC6205

Apuntes de clases del curso de Procesamiento de Lenguaje Natural de la Universidad de Chile.

El formato del apunte fue tomado del template de Jasmine Hao.

Borrador, 8 de noviembre de 2023



Índice general

0.1 Ejemplos de Problemas de Clasificación	4
0.2 Modelos Generativos	6
0.3 Naïve Bayes Multinomial	7
0.3.1 Estimación de parámetros	7
0.3.2 Problemas al multiplicar muchas probabilidades	8
0.3.3 Probabilidades cero y el problema de las palabras no vistas	8
0.3.4 Suavizado de Laplace (Add-1) para Naïve Bayes	9
0.4 Ejemplo	9
0.5 Naïve Bayes como modelo de lenguaje	11
0.6 Evaluación	11
0.6.1 La Matriz de Confusión 2x2	12
0.6.2 Evaluación: Exactitud	12
0.6.3 Evaluación: Precisión y Recall	12
0.6.4 ¿Por qué Precisión y Recall?	13
0.6.5 Una Medida Combinada: Medida F	13
0.6.6 Conjuntos de Prueba de Desarrollo ("Devsets")	13
0.6.7 Validación Cruzada: Múltiples Divisiones	13
0.7 Conjuntos de entrenamiento, prueba y validación	14
0.7.1 Matriz de Confusión para clasificación de 3 clases	15

La clasificación, que implica la asignación de un objeto a una categoría específica, desempeña un papel fundamental tanto en la inteligencia humana como en la artificial. En este contexto, la clasificación abarca diversas tareas, que van desde determinar qué letra, palabra o imagen se ha presentado a nuestros sentidos, hasta reconocer caras, voces, clasificar correos electrónicos o calificar tareas.

El propósito subyacente de la clasificación es tomar una única observación, identificar y extraer características relevantes de la misma, y, en última instancia, ubicarla en una de las categorías discretas predefinidas.

Definición 0.0.1 Formalmente, definimos el problema de clasificación de texto, al escenario en que se tiene un documento d y un conjunto fijo de clases $C = \{c_1, c_2, \dots, c_J\}$ y se requiere predecir una clase $c \in C$ para d .

Como se discutió en el Capítulo ??, la construcción de clasificadores mediante reglas manuales no resulta ser un enfoque eficaz. Hoy en día la mayoría de las tareas de clasificación en PLN se abordan mediante enfoques de aprendizaje automático supervisado.

Formalmente, se parte de un conjunto fijo de clases $C = \{c_1, c_2, \dots, c_J\}$ y un conjunto de entrenamiento compuesto por m documentos que han sido etiquetados manualmente como

$$(d_1, c_1), (d_2, c_2), \dots, (d_m, c_m).$$

A través de un proceso de entrenamiento, se desarrolla un clasificador que se define como una función $\gamma: d \rightarrow c$.

Diversos algoritmos de clasificación están disponibles para este propósito, como Naïve Bayes, Regresión logística, Redes neuronales, k-vecinos más cercanos, entre otros. En este capítulo, nos centraremos en el modelo Naïve Bayes. Este capítulo se basa en el material del curso de Daniel Jurafsky, al que se puede acceder a través del siguiente enlace¹.

0.1 Ejemplos de Problemas de Clasificación

La clasificación de texto se puede aplicar a varias tareas, incluyendo:

- Análisis de sentimientos
 - Detección de spam
 - Identificación de autoría
 - Identificación de idioma
 - Asignación de categorías, temas o géneros
- Analicemos estos ejemplos en más detalle:

■ **Ejemplo 0.1** Clasificación de spam. En este problema el objeto a clasificar es un correo electrónico y las etiquetas posibles son SPAM y no SPAM. Las etiquetas se obtienen del mismo usuario que etiqueta los correos.

■ **Ejemplo 0.2** Detección de Autoría. Un ejemplo histórico de detección de autoría se refiere a la autoría de los Ensayos Federalistas de la Constitución de EE. UU. En 1787, se escribieron ensayos anónimos para persuadir a Nueva York de ratificar la Constitución de EE. UU. La autoría de 12 de estos ensayos estuvo en disputa entre James Madison y Alexander Hamilton (Tabla 1) hasta que en 1963, Mosteller y Wallace [?] utilizaron métodos bayesianos para identificar que Hamilton era el autor de los mismos.

■ **Ejemplo 0.3** Clasificación por tópico. Aquí los ejemplos incluyen clasificar una noticia en una categoría temática (ej: deportes, política, economía) o etiquetar automáticamente un artículo en

¹<https://web.stanford.edu/~jurafsky/slp3/4.pdf>

Subject: Important notice!
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greates News!

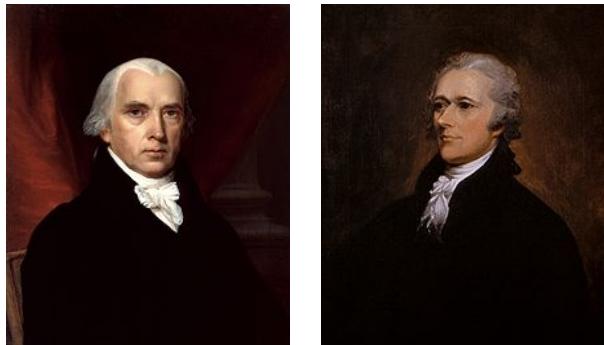
You can now access the latest news by using the link below to login to Stanford University News Forum.

<http://www.123contactform.com/contact-form-StanfordNew1-236335.html>

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information about the new services.

© Stanford University. All Rights Reserved.

Figura 1: Ejemplo de Spam



James Madison

Alexander Hamilton

Cuadro 1: Autores candidatos a autoría.

ciencias de la vida en una categoría del Medical Subject Headings (MeSH) como se ilustra en la Figura 2.

■ **Ejemplo 0.4** Clasificación de sentimientos. La clasificación de sentimientos se utiliza para determinar si un documento exhibe un sentimiento positivo o negativo. Este enfoque se basa en el análisis del tono y las palabras utilizadas en el texto. A continuación, se presentan ejemplos de reseñas de películas y comentarios de restaurantes:

- + ...una película muy **buena**, especialmente por los giros de la trama. ¡Fue **espectacular**!
- - La actuación fue **patética**, destacando las escenas de baile como lo **peor**.
- + ...la salsa de chocolate con almendras dulces es simplemente **increíble** en este lugar. ¡Me **encanta**!
- - ...la pizza estaba **horrible** y tenía un precio **ridículamente** alto.

Este enfoque se aplica en diversos casos, como medir la confianza del consumidor respecto a un producto o predecir resultados electorales y tendencias del mercado en función del sentimiento. Estas aplicaciones suelen ser el resultado de la implementación de modelos de análisis de sentimientos en datos procedentes de las redes sociales.

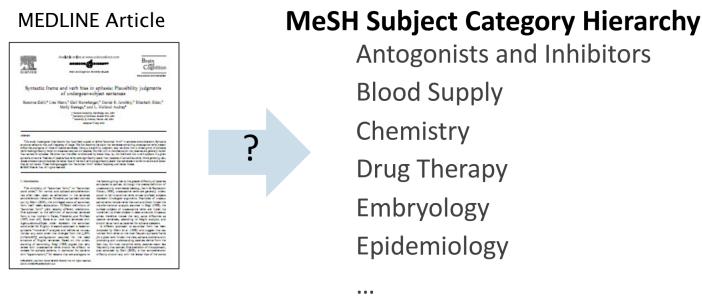


Figura 2: Ejemplo de clasificación por categorías clínicas.

0.2 Modelos Generativos

Nuestra tarea principal consiste en aprender una función f que asigne etiquetas $f(d)$ a las entradas d . El modelo Naïve Bayes sigue un enfoque probabilístico en el que buscamos estimar la probabilidad condicional $p(c|d)$ utilizando ejemplos de entrenamiento para todas las clases o etiquetas (por ejemplo, $c = c_1, c = c_2, \dots, c = c_k$), donde $0 \leq p(c = c_j|d) \leq 1$ para cada j y $\sum_j p(c = c_j|d) = 1$. Luego, para cualquier entrada de prueba \tilde{d} , definimos $f(\tilde{d}) = c_{\text{MAP}} = \arg \max_c p(c|\tilde{d})$ como el estimador MAP (Maximum a Posteriori), lo cual equivale a seleccionar la clase más probable dada la información disponible en los datos.

Es importante señalar que existen dos enfoques principales en el aprendizaje automático en términos de la modelización de esta probabilidad condicional $p(c|d)$. Los modelos que intentan estimar $p(c|d)$ directamente de los datos se denominan modelos “discriminativos”, como la regresión logística que exploraremos en el Capítulo ??.

Por otro lado, Naïve Bayes se clasifica como un “modelo generativo”. Bajo este enfoque, se busca construir un modelo para cada clase y clasificar una entrada observando cuál clase tiene la mayor probabilidad de generar el ejemplo objetivo.

Este concepto puede formalizarse mediante el teorema de Bayes. Dado un documento d y una clase c , tenemos:

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

Aquí, $p(d|c)$ representa la “verosimilitud” que representa la probabilidad de que una clase particular “genere” el documento. Por otro lado, $p(c)$ se refiere a la probabilidad “a priori” de la clase y $p(d)$ corresponde a la probabilidad del documento, también conocida como “evidencia”.

Dado que solo estamos interesados en encontrar la clase más probable para la clasificación, podemos eliminar el denominador, ya que se mantiene constante para todas las clases. Esto se expresa como:

$$c_{\text{MAP}} = \arg \max_c p(d|c)p(c)$$

En resumen, el estimador MAP se utiliza para encontrar la clase más probable según la expresión anterior.

Alternativamente, podemos entender los modelos generativos como un intento de estimar la distribución conjunta $p(d, c)$ a partir de los ejemplos de entrenamiento. Utilizando la definición de probabilidad condicional, tenemos $p(d, c) = p(c)p(d|c)$, lo cual es equivalente al numerador que maximizamos previamente utilizando el teorema de Bayes.

0.3 Naïve Bayes Multinomial

Naïve Bayes, o el modelo Bayesiano “ingenuo” [?], se presenta como un modelo de clasificación generativo en el que un documento d se representa como una bolsa de palabras x_1, x_2, \dots, x_n . Similar al modelo vectorial explorado en el Capítulo ??, en este modelo se omiten las posiciones de las palabras dentro del documento.

La verosimilitud del modelo probabilístico se expresa de la siguiente manera:

$$p(d|c) = p(x_1, x_2, \dots, x_n|c)$$

Luego, se aplica el supuesto de independencia condicional, que establece que las palabras son independientes entre sí cuando se condicionan a la clase. Esto permite factorizar la verosimilitud como el producto de factores para cada palabra del vocabulario:

$$p(x_1, x_2, \dots, x_n|c) = p(x_1|c) \cdot p(x_2|c) \cdot p(x_3|c) \cdot \dots \cdot p(x_n|c) = \prod_i p(x_i|c)$$

Con esta factorización, la estimación MAP del modelo para clasificar un documento se formula de la siguiente manera:

$$c_{\text{MAP}} = \arg \max_{c \in C} \prod_i p(x_i|c)p(c)$$

0.3.1 Estimación de parámetros

Para estimar los parámetros del modelo Naive Bayes multinomial, se utiliza el método de estimación por máxima verosimilitud, asumiendo una distribución multinomial para $p(x_1, x_2, \dots, x_n|c)$. Esto implica la creación de un “mega-dокументo” para cada clase c_j mediante la concatenación de todos los documentos de esa clase. Luego, se calcula la frecuencia de todas las palabras w_i del vocabulario en el mega-dокументo para cada clase $\text{count}(w_i, c_j)$.

La probabilidad estimada $\hat{p}(w_i|c_j)$ de la palabra w_i dada la clase c_j se obtiene dividiendo el conteo de ocurrencias de w_i en el mega-documento de la clase c_j por el total de palabras en el mega-documento:

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

La probabilidad previa de una clase c_j , $p(c_j)$, se estima de la siguiente manera:

$$\hat{p}(c_j) = \frac{N_{c_j}}{N_{\text{total}}}$$

donde N_{c_j} es el número de documentos en la clase c_j y N_{total} es el número total de documentos.

Una vez que el modelo Naïve Bayes (NB) ha sido entrenado, se puede emplear para clasificar un nuevo documento \tilde{d} de la siguiente manera:

$$c_{\text{NB}} = \arg \max_{c_j} p(c_j) \prod_{i \in \text{posiciones}} p(x_i | c_j)$$

Básicamente, se debe iterar a través de las palabras en las posiciones del documento y calcular las probabilidades $p(c_j)$ y $p(x_i | c_j)$ para todas las clases y palabras del documento, para finalmente retornar la clase más probable. Estas probabilidades se derivan de los conteos almacenados durante el proceso de entrenamiento.

Cuando el documento de prueba \tilde{d} contiene palabras que son desconocidas, es decir, no se encuentran en los datos de entrenamiento ni en el vocabulario, optamos por omitirlas. En este contexto, no asignamos ninguna probabilidad a estas palabras desconocidas durante el proceso de clasificación.

0.3.2 Problemas al multiplicar muchas probabilidades

La multiplicación de muchas probabilidades puede llevar a un underflow de punto flotante², especialmente cuando se manejan probabilidades pequeñas. Por ejemplo, $0,0006 \times 0,0007 \times 0,0009 \times 0,01 \times 0,5 \times 0,000008 \dots$. Esto resulta en productos con valores cercanos a cero.

Para abordar este problema, podemos recurrir a los logaritmos, ya que $\log(ab) = \log(a) + \log(b)$. En lugar de multiplicar las probabilidades, sumamos los logaritmos de las probabilidades. Así, el clasificador Naive Bayes multinomial se puede expresar utilizando logaritmos de la siguiente manera:

$$c_{\text{NB}} = \arg \max_{c_j \in C} \left(\log(P(c_j)) + \sum_{i \in \text{posiciones}} \log(P(x_i | c_j)) \right)$$

Al aplicar logaritmos, el clasificador Naive Bayes se convierte en un modelo lineal (consulte el Capítulo ??), donde la predicción es el argmax de la suma de los logaritmos de las probabilidades y las entradas (logaritmos de las probabilidades condicionales). En otras palabras, Naive Bayes se convierte en un clasificador lineal que opera en el espacio logarítmico.

0.3.3 Suavizamiento de Laplace

De forma similar a los modelos de lenguaje de n-gramas mencionados en el Capítulo ??, Naïve Bayes también enfrenta problemas de probabilidades nulas.

Supongamos que estamos trabajando en un problema de clasificación de sentimientos con clases positivo y negativo. Por alguna razón, la palabra “fantástico” no aparece en ningún documento negativo. Luego, imaginemos que queremos clasificar el documento \tilde{d} = “Qué fantástico, acabo de aburrirme como nunca viendo la peor película de mi vida”. Claramente, este documento es negativo, pero si usamos la estimación de máxima verosimilitud, la probabilidad $\hat{p}(\text{“fantástico”} | \text{negativo})$ sería:

$$\hat{p}(\text{“fantástico”} | \text{negativo}) = \frac{\text{count(“fantástico”, negativo)}}{\sum_{w \in V} \text{count}(w, \text{negativo})}$$

²https://en.wikipedia.org/wiki/Arithmetic_underflow

En este caso, el conteo de la palabra “fantástico” en los documentos negativos es cero, lo que resulta en una probabilidad cero:

$$\hat{p}(\text{“fantástico”} \mid \text{positivo}) = \frac{0}{\sum_{w \in V} \text{count}(w, \text{positivo})} = 0$$

El problema es que esta probabilidad cero anularía toda la evidencia proporcionada por otras palabras muy indicativas de la clase negativa, como “aburrirme” y “peor”, y el clasificador Naïve Bayes terminaría clasificando el ejemplo en la clase “positiva”:

$$c_{\text{MAP}} = \arg \max_c \left(\hat{p}(c) \prod_i \hat{p}(x_i \mid c) \right)$$

Para abordar el problema de las probabilidades cero, podemos utilizar la técnica de suavizamiento de Laplace, que consiste en agregar un conteo adicional a todas las combinaciones de palabras y clases. La estimación suavizada $\hat{p}(w_i \mid c)$ quedaría de la siguiente manera:

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

El denominador se ajusta agregando el tamaño del vocabulario V para garantizar una normalización adecuada. Al hacerlo, evitamos las probabilidades cero y permitimos que cierta masa de probabilidad se distribuya a palabras no vistas. Esta técnica aborda el problema de las probabilidades nulas, ya que ahora cualquier palabra del vocabulario, incluso si nunca se ha visto con alguna clase, recibirá una probabilidad condicional mayor a cero. Como pueden ver, esta técnica tiene una semejanza importante con las técnicas de interpolación lineal y de descuento vistas en el Capítulo de modelos de lenguaje (Capítulo ??).

0.4 Ejemplo

Supongamos que tenemos el siguiente conjunto de entrenamiento:

Entrenamiento:

Categoría	Texto
Perú	Lima Lima Trujillo
Chile	Santiago Concepción Concepción
Perú	Perú Lima Trujillo América
Chile	Chile América Santiago

Y queremos clasificar el siguiente ejemplo:

Test:

Categoría	Texto
?	América Lima Perú Santiago

En este ejemplo, $|V| = 7$, y tenemos dos clases: $c_1 = \text{Perú}$ y $c_2 = \text{Chile}$.

id_palabra	palabra	count(w_i , Perú)	count(w_i , Chile)
w_1	América	1	1
w_2	Chile	0	1
w_3	Concepción	0	2
w_4	Lima	3	0
w_5	Perú	1	0
w_6	Santiago	0	2
w_7	Trujillo	2	0

La tabla de conteos que calculamos de los datos de entrenamiento quedaría de la siguiente manera:

Ahora, debemos calcular las probabilidades condicionales suavizadas:

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

para todas las palabras y clases.

El denominador de esta expresión para cada clase corresponde al largo del “mega-documento” que junta todos los documentos de la misma clase más el tamaño del vocabulario, que sería $7 + 7 = 14$ para Perú y $6 + 7 = 13$ para Chile.

id_palabra	palabra	$\hat{p}(w_i c = \text{Perú})$	$\hat{p}(w_i c = \text{Chile})$
w_1	América	$\frac{1+1}{14} = 0,14$	$\frac{1+1}{13} = 0,15$
w_2	Chile	$\frac{0+1}{14} = 0,07$	$\frac{1+1}{13} = 0,15$
w_3	Concepción	$\frac{0+1}{14} = 0,07$	$\frac{2+1}{13} = 0,23$
w_4	Lima	$\frac{3+1}{14} = 0,29$	$\frac{0+1}{13} = 0,08$
w_5	Perú	$\frac{1+1}{14} = 0,14$	$\frac{0+1}{13} = 0,08$
w_6	Santiago	$\frac{0+1}{14} = 0,07$	$\frac{2+1}{13} = 0,23$
w_7	Trujillo	$\frac{2+1}{14} = 0,21$	$\frac{0+1}{13} = 0,08$

Ahora necesitamos estimar la probabilidad a priori de cada clase:

$$\hat{p}(c_j) = \frac{N_{c_j}}{N_{\text{total}}}$$

Ambas clases ocurren en 2 documentos sobre un total de 4. Entonces, $\hat{p}(c = \text{Perú}) = 0,5$ y $\hat{p}(c = \text{Chile}) = 0,5$.

Ahora debemos calcular:

$$c_{\text{NB}} = \arg \max_{c_j} p(c_j) \prod_{i \in \text{posiciones}} p(x_i | c_j)$$

Esto implica iterar posiciones para ambas clases del documento de prueba:

$$p(\tilde{d} | c = \text{Perú}) \approx 0,14 \times 0,29 \times 0,14 \times 0,07 \times 0,5 = 0,00019894$$

$$p(\tilde{d} | c = \text{Chile}) \approx 0,15 \times 0,08 \times 0,08 \times 0,23 \times 0,5 = 0,0001104$$

Entonces, clasificamos el documento en la clase “Perú”.

0.5 Naive Bayes como modelo de lenguaje

Cuando utilizamos características de palabras individuales y consideramos todas las palabras en el texto, el naive Bayes tiene una similitud importante con la modelización del lenguaje.

Específicamente, un modelo naive Bayes se puede ver como un conjunto de modelos de lenguaje de unigramas específicos de cada clase, en el que el modelo para cada clase instancia un modelo de lenguaje de unigrama.

Las características de verosimilitud del modelo naive Bayes asignan una probabilidad a cada palabra $P(\text{word}|c)$, y el modelo también asigna una probabilidad a cada oración:

$$P(s|c) = \prod_{i \in \text{positions}} P(w_i|c)$$

Consideremos un modelo naive Bayes con las clases positiva (+) y negativa (-) y los siguientes parámetros del modelo:

w	$P(w +)$	$P(w -)$
I	0.1	0.2
love	0.1	0.001
this	0.01	0.01
fun	0.05	0.005
film	0.1	0.1
...

Cada una de las dos columnas anteriores instancian un modelo de lenguaje que puede asignar una probabilidad a la oración "I love this fun film":

$$P("I love this fun film" | +) = 0,1 \times 0,1 \times 0,01 \times 0,05 \times 0,1 = 0,0000005$$

$$P("I love this fun film" | -) = 0,2 \times 0,001 \times 0,01 \times 0,005 \times 0,1 = 0,000000010$$

Como sucede, el modelo positivo asigna una probabilidad más alta a la oración:

$$P(s|\text{pos}) > P(s|\text{neg})$$

Cabe destacar que esto es solo la parte de verosimilitud del modelo naive Bayes; una vez que multiplicamos por la probabilidad a priori, un modelo naive Bayes completo podría tomar una decisión de clasificación diferente.

0.6 Evaluación

- Consideremos solo tareas de clasificación de texto binario.
- Imagina que eres el CEO de Delicious Pie Company.
- Quieres saber lo que la gente está diciendo sobre tus pasteles.
- Por lo tanto, construyes un detector de tweets de "Delicious Pie" con las siguientes clases:
 - Clase positiva: tweets sobre Delicious Pie Co.
 - Clase negativa: todos los demás tweets.

	Sistema Positivo	Sistema Negativo
Oro Positivo	Verdadero Positivo (VP)	Falso Negativo (FN)
Oro Negativo	Falso Positivo (FP)	Verdadero Negativo (VN)

0.6.1 La Matriz de Confusión 2x2

Recall (también conocido como **Sensibilidad** o **Tasa de Verdaderos Positivos**):

$$\text{Recall} = \frac{VP}{VP + FN}$$

Precisión:

$$\text{Precisión} = \frac{VP}{VP + FP}$$

Exactitud:

$$\text{Exactitud} = \frac{VP + VN}{VP + FP + VN + FN}$$

0.6.2 Evaluación: Exactitud

¿Por qué no usamos la exactitud como nuestra métrica?

Imagina que vimos 1 millón de tweets:

- 100 de ellos hablaban sobre Delicious Pie Co.
- 999,900 hablaban de otra cosa.

Podríamos construir un clasificador tonto que simplemente etiquete todos los tweets como "no sobre pasteles":

- ¡¡¡Obtendría una exactitud del 99.99 %!!! ¡¡¡Wow!!!
- ¡Pero sería inútil! ¡No devuelve los comentarios que estamos buscando!

Por eso usamos precisión y recall en su lugar.

0.6.3 Evaluación: Precisión y Recall

Precisión mide el porcentaje de elementos que el sistema detectó (es decir, los elementos que el sistema etiquetó como positivos) que son realmente positivos (según las etiquetas de oro humanas).

$$\text{Precisión} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Positivos}}$$

Recall mide el porcentaje de elementos que el sistema identificó correctamente de todos los elementos que deberían haber sido identificados.

$$\text{Recall} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}}$$

0.6.4 ¿Por qué Precisión y Recall?

Considera nuestro clasificador tonto de pasteles que simplemente etiqueta nada como "sobre pasteles".

- Exactitud = 99.99 % (etiqueta correctamente la mayoría de los tweets como no relacionados con pasteles)
 - Recall = 0 (no detecta ninguno de los 100 tweets relacionados con pasteles)
- La precisión y el recall, a diferencia de la exactitud, enfatizan los verdaderos positivos:
- Se centran en encontrar las cosas que se supone que debemos buscar.

0.6.5 Una Medida Combinada: Medida F

La medida F es un número único que combina la precisión (P) y el recall (R), definida como:

$$F_{\beta} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

La medida F, definida con el parámetro β , pondera diferencialmente la importancia del recall y la precisión.

- $\beta > 1$ favorece al recall
- $\beta < 1$ favorece a la precisión

Cuando $\beta = 1$, la precisión y el recall son iguales, y tenemos la medida F_1 equilibrada:

$$F_1 = \frac{2PR}{P + R}$$

0.6.6 Conjuntos de Prueba de Desarrollo ("Devsets")

- Para evitar el sobreajuste y proporcionar una estimación más conservadora del rendimiento, comúnmente utilizamos un enfoque de tres conjuntos: conjunto de entrenamiento, conjunto de desarrollo y conjunto de prueba.

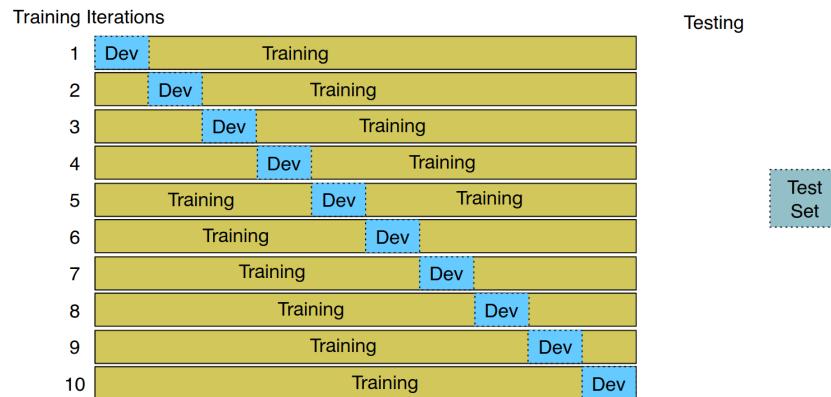


- **Conjunto de entrenamiento:** Se utiliza para entrenar el modelo.
- **Conjunto de desarrollo:** Se utiliza para ajustar el modelo y seleccionar los mejores hiperparámetros.
- **Conjunto de prueba:** Se utiliza para informar el rendimiento final del modelo.
- Este enfoque garantiza que el modelo no esté ajustado específicamente al conjunto de prueba, evitando el sobreajuste.
- Sin embargo, crea una paradoja: queremos la mayor cantidad de datos posible para el entrenamiento, pero también para el conjunto de desarrollo.
- ¿Cómo dividimos los datos?

0.6.7 Validación Cruzada: Múltiples Divisiones

- La validación cruzada nos permite utilizar todos nuestros datos para el entrenamiento y la prueba sin tener un conjunto de entrenamiento, conjunto de desarrollo y conjunto de prueba fijos.

- Elegimos un número k y dividimos nuestros datos en k subconjuntos disjuntos llamados pliegues.
- En cada iteración, uno de los pliegues se selecciona como conjunto de prueba mientras que los $k - 1$ pliegues restantes se utilizan para entrenar el clasificador.
- Calculamos la tasa de error en el conjunto de prueba y repetimos este proceso k veces.
- Finalmente, promediamos las tasas de error de estas k ejecuciones para obtener una tasa de error promedio.
- Por ejemplo, la validación cruzada de 10 pliegues implica entrenar 10 modelos con el 90 % de los datos y probar cada modelo por separado.
- Las tasas de error resultantes se promedian para obtener la estimación final del rendimiento.
- Sin embargo, la validación cruzada requiere que todo el corpus sea ciego, lo que impide examinar los datos para sugerir características o comprender el comportamiento del sistema.
- Para abordar esto, se crea un conjunto de entrenamiento y un conjunto de prueba fijos, y se realiza la validación cruzada de 10 pliegues dentro del conjunto de entrenamiento.
- La tasa de error se calcula convencionalmente en el conjunto de prueba.



0.7 Conjuntos de entrenamiento, prueba y validación

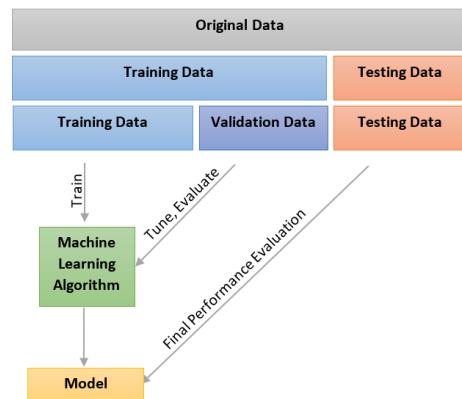
Cuando entrenamos un modelo, nuestro objetivo es producir una función $f(\vec{x})$ que mapee correctamente las entradas \vec{x} a las salidas \hat{y} según lo evidenciado por el conjunto de entrenamiento. La evaluación del rendimiento en los datos de entrenamiento puede ser engañosa, ya que nuestro objetivo es entrenar una función capaz de generalizar a ejemplos no vistos. Una forma común de abordar esto es dividir el conjunto de entrenamiento en subconjuntos de entrenamiento y prueba (80 % y 20 % respectivamente). Se entrena el modelo en el subconjunto de entrenamiento y se calcula la precisión en el subconjunto de prueba.

Sin embargo, este enfoque tiene una limitación. En la práctica, a menudo se entranan varios modelos, se comparan sus calidades y se selecciona el mejor. Si se selecciona el mejor modelo en función de la precisión en el subconjunto de prueba, se obtendrá una estimación excesivamente optimista de la calidad del modelo. No se sabe si la configuración elegida del clasificador final es buena en general o simplemente es buena para los ejemplos particulares en los subconjuntos de prueba.

La metodología aceptada es utilizar una división de tres vías de los datos en conjuntos de

entrenamiento, validación (también llamado desarrollo) y prueba³. Esto proporciona dos conjuntos apartados: un conjunto de validación (también llamado conjunto de desarrollo) y un conjunto de prueba. Todos los experimentos, ajustes, análisis de errores y selección de modelos deben realizarse

basados en el conjunto de validación. Luego, una única ejecución del modelo final sobre el conjunto de prueba proporcionará una buena estimación de su calidad esperada en ejemplos no vistos. Es importante mantener el conjunto de prueba lo más limpio posible, realizando la menor cantidad de experimentos posible en él. Incluso algunos defienden que no se deben mirar siquiera los ejemplos en el conjunto de prueba, para evitar sesgar el diseño del modelo.



0.7.1 Matriz de Confusión para clasificación de 3 clases

			gold labels			
			urgent	normal	spam	
system output	urgent	8	10	1		precision _u = $\frac{8}{8+10+1}$
	normal	5	60	50		precision _n = $\frac{60}{5+60+50}$
	spam	3	30	200		precision _s = $\frac{200}{3+30+200}$
			recall _u = $\frac{8}{8+5+3}$	recall _n = $\frac{60}{10+60+30}$	recall _s = $\frac{200}{1+50+200}$	

Cómo combinar métricas binarias (Precisión, Recall, F_1) de más de 2 clases para obtener una métrica única:

- Macro-promedio:
 - Calcular las métricas de rendimiento (Precisión, Recall, F_1) para cada clase individualmente.
 - Promediar las métricas en todas las clases.
- Micro-promedio:
 - Recopilar las decisiones para todas las clases en una matriz de confusión.

³Un enfoque alternativo es la validación cruzada, pero no se escala bien para entrenar redes neuronales profundas.

Fuente: <https://www.codeproject.com/KB/AI/1146582/validation.PNG>

- Calcular la Precisión y el Recall a partir de la matriz de confusión.

		Class 1: Urgent		Class 2: Normal		Class 3: Spam		Pooled	
		true	true	true	true	true	true	true	true
		urgent	not	normal	not	spam	not	yes	no
system	urgent	8	11	system	60	55	system	200	33
system	not	8	340	system	40	212	system	51	83

precision = $\frac{8}{8+11} = .42$ precision = $\frac{60}{60+55} = .52$ precision = $\frac{200}{200+33} = .86$ microaverage precision = $\frac{268}{268+99} = .73$
 macroaverage precision = $\frac{.42+.52+.86}{3} = .60$