uc3m | Universidad Carlos III de Madrid

HULAT Human Language & Accessibility Technologies

OpenCourseWare

Procesamiento de Lenguaje Natural con Aprendizaje Profundo,

Máster en Ciencia y Tecnología Informática

Tema 1.3. Métricas de Evaluación en Procesamiento de Lenguaje natural

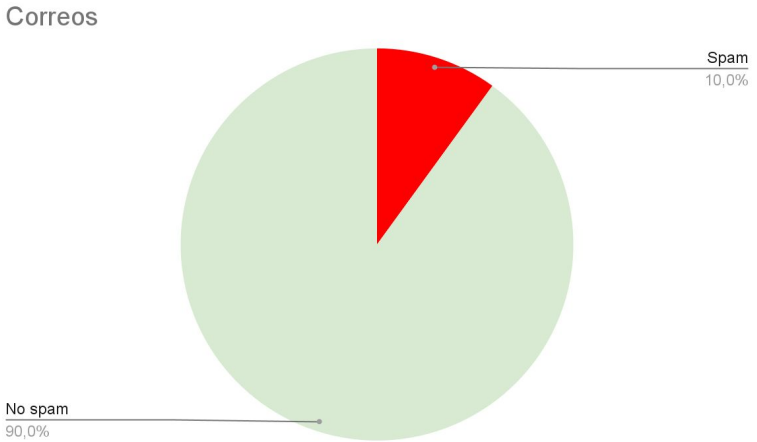
Isabel Segura Bedmar

Objetivos

- En este tema, vamos a estudiar algunas de las **métricas** más populares que se utilizan para **evaluar los sistemas de PLN**, en particular, aquellos que son tareas de **clasificación**.
- Vamos a aprender qué es la **matriz de confusión**.
- Estudiaremos las métricas de **accuracy**, **precisión**, **recall** y **F1**.
- También vamos a estudiar las métricas para problemas de multi-clasificación: **macro, weighted-macro** y **micro.**.

Para estudiar las métricas

- Propondremos como ejemplo un sistema cuyo objetivo es detectar **correos spam** (clasificación de textos), es decir, clasificar si un correo es spam o no.
- Vamos a suponer que nuestro dataset está formado por **10,000 correos**:
  - 9.000 no spam
  - 1000 spam



Clasificación binaria

- Si las clases no están balanceadas , la **clase mayoritaria** representa "**normal**" y la clase minoritaria representa "**anormal**".
- Es preferible un **buen resultado en la clase minoritaria** a un buen resultado en ambas clases.
- En nuestro ejemplo,
  - la clase **minoritaria** (**positiva**) es la de los correos spam.
  - la clase **mayoritaria** (**negativa**) es la de los correos no spam.

Matriz de confusión

- Ningún algoritmo va a ser perfecto, por tanto:
  - Los correos que no son spam pero que nuestro algoritmo los clasifica como spam son **falsos positivos** (false positives).
  - Los correos que son spam pero que nuestro algoritmo no es capaz de identificar (es decir, son clasificados como no spam) son **falsos negativos** (false negatives).

Matriz de confusión

- Es de esperar que el algoritmo también tiene aciertos:
  - Los correos que son spam y que además han sido correctamente clasificados por nuestro algoritmo son **ciertos positivos** (true positives).
  - Los correos que no son spam y que han sido clasificados como no spam por nuestro algoritmo son **ciertos negativos** (true positives).

    .

Matriz confusión para clasificación binaria

Valores reales

Predicciones del algoritmo	**Positivo (correos spam)**	**Negativo (correos no spam)**
Positivo (correos spam)	Ciertos positivos (True positives, **TP**)	Falsos positivos (False positives, **FP**)
Negativo (correos no spam)	Falsos negativos (False negatives, **FN**)	Ciertos negativos (True negatives, TN)

Matriz confusión (ejemplo)

- Supongamos que nuestro algoritmo es capaz de identificar 700 correos spam, pero erróneamente clasifica como spam otros 700 correos.

		Valores reales	
		Positivo (correos spam)	**Negativo (correos no spam)**
Predicciones del algoritmo	**Positivo (estimado como spam)**	TP = 700	FP = 700
	Negativo (estimado como no spam)	FN = 300	TN = 8300

Precisión

- Mide cómo de preciso es el algoritmo: cuántas de las instancias que han sido clasificadas como positivas, realmente son ejemplos de la clase positiva. Su fórmula es la siguiente:

$$precision = \frac{TP}{TP+FP}$$

Precisión

- En nuestro ejemplo, la precisión es 700 / (700 +700) = 0.50.
- Es decir, nuestro algoritmo se equivocará un 50% de las veces cuando predice que un correo es spam.
- Date cuenta que es un algoritmo bastante malo. No mucho mejor que tirar una moneda ;).

	Positivo (correos spam)	**Negativo (correos no spam)**
Positivo (estimado como spam)	TP = 700	FP = 700
Negativo (estimado como no spam)	FN = 300	TN = 8300

Recall (Exhaustividad)

- Calcula qué porcentaje de ejemplos positivos es capaz de identificar. Es decir, es el ratio entre el número total de aciertos del algoritmo para la clase positiva (minoritaria) y el número total de ejemplos positivos.

$$recall = \frac{TP}{TP+FN}$$

Recall (Exhaustividad)

- En nuestro ejemplo, el recall es 700 / (700 +300) = 0.70.
- Es decir, nuestro algoritmo es capaz de identificar el 70% de los correos que son spams.
- Tiene un recall aceptable, pero una precisión mala (50%).

	Positivo (correos spam)	**Negativo (correos no spam)**
Positivo (estimado como spam)	TP = 700	FP = 700
Negativo (estimado como no spam)	FN = 300	TN = 8300

F1 (valor F)

- Recall y precisión suelen competir.
- Lo deseable es que nuestro algoritmo tenga la mayor precisión y recall posible (aunque a veces están reñidos)
- Los resultados de precisión y recall se pueden combinar en un único valor, su media armónica, que nos va a permitir elegir el mejor algoritmo, teniendo en cuenta ambas métricas:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

F1 (valor F)

- En nuestro ejemplo, F1 = 2 * (0.50 * 0.70) / 1.2 = 0.58
- Date cuenta que el valor F1, asume que nos importa de igual forma la precisión y el recall.
- Sin embargo, esto no tiene que ser siempre así.
- Por ejemplo, si estamos desarrollando un algoritmo para detectar el plagio en exámenes, ¿nos interesa que tenga mayor precisión o mayor recall?.

F1 (valor F)

$$F_\beta = (1 + \beta^2) * \frac{precision * recall}{((\beta^2 * precision) + recall}$$

- beta = 1, mismo peso a precisión y recall.
- beta = 0.5, se prioriza la precisión.
- beta = 2, se prioriza el recall.

Accuracy (Exactitud)

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- Mide el porcentaje de casos que el algoritmo acierta en ambas clases (es decir, los true positives pero también los true negatives).
- Su principal carencia es que suele dar una **estimación poco correcta**, si las clases **no están bien balanceadas.**

Accuracy (Exactitud)

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

	Positivo (correos spam)	**Negativo (correos no spam)**
Positivo (estimado como spam)	TP = 700	FP = 700
Negativo (estimado como no spam)	FN = 300	TN = 8300

- En nuestro caso, acc = 9000 / 10000 = 0.9.
- Una exactitud del 90%!!!.
- Parece un buen resultado, sin embargo, su F1 era 58% y una precisión bastante pobre 50% para la clase positiva.

Un caso aún más extremo!!!

- Calcula las métricas para un **algoritmo que clasifique todos los correos como no spam**. Su matriz de confusión será:

		Valores reales
Predicciones del algoritmo	**Positivo (spam)**	**Negativo (no spam)**
Positivo (estimado como spam)	TP = 0	FP = 0
Negativo (estimado como no spam)	FN = 1000	TN = 9000

Solución

- $P_{spam} = R_{spam} = F1_{spam} = 0$ para la clase positiva (correos spam)

- $P_{no\ pam} = 9000 / 10000 = 0.9$
- $R_{no\ spam} = 9000 / 9000 = 1$
- $F1_{no\ spam} = 2*(0.9*1)/(0.9+1) = 0.94$
- Acc = 9000 / 10000 = 0.9

	Positivo (spam)	**Negativo (no spam)**
Positivo (estimado como spam)	TP = 0	FP = 0
Negativo (estimado como no spam)	FN = 1000	TN = 9000

Solución

- El algoritmo que clasifica todos los correos como no spam, tiene un accuracy de 90!!!.
- Además, la F1 para su clase negativa (no spam) es también 0.94.
- Sin embargo, su F1 es 0 en la clase positiva (P=R=0).
- Es un pésimo sistema, aunque su accuracy y F1 en la clase no negativa son muy altas.
- **Accuracy no es fiable** cuando las **clases no** están **balanceadas**.

Métricas para multi-clasificación

- Precisión, recall y F1 pueden ser fácilmente extendidas para problemas de **multi-clasificación**.
- Para cada clase, se calculan **precisión, recall y F1**, y de esta forma se conoce el desempeño del algoritmo **para cada clase.**
- ¿Pero cómo puedo dar una **estimación global** del desempeño del algoritmo?

Métricas para multi-clasificación

- Estudiaremos las siguientes métricas:
  - macro
  - weighted macro
  - micro

Métricas versión macro

- La **macro precisión** es la media aritmética de las precisiones para las diferentes clases.
- El **macro recall** es la media aritmética de las recall de cada clase.
- La **macro F1** será la media aritmética de las F1 de cada clase

$$macro\_precision = \frac{\sum_{i=1}^{N} precision_i}{N}$$

$$macro\_recall = \frac{\sum_{i=1}^{N} recall_i}{N}$$

$$macro\_f1 = \frac{\sum_{i=1}^{N} F1_i}{N}$$

donde N es el número de clases

Métricas versión macro

- Vamos a suponer que tenemos un algoritmo para el análisis de sentimiento con 5 clases, y que ha obtenido los siguientes resultados:

	precision	recall	f1-score
1	1.00	0.16	0.27
2	1.00	0.05	0.10
3	1.00	0.35	0.52
4	1.00	0.21	0.35
5	0.77	1.00	0.87

Métricas versión macro

- **macro P** = (1.00 + 1.00 + 1.00 + 1.00 +0.77) / 5 = **0.95**
- **macro R** = (0.16 + 0.05 + 0.35 + 0.21 + 1.00) 5 = **0.35**
- **macro F1** = (0.27 + 0.10 + 0.52 + 0.35 + 0.87) / 5 = **0.42**

	precision	recall	f1-score
1	1.00	0.16	0.27
2	1.00	0.05	0.10
3	1.00	0.35	0.52
4	1.00	0.21	0.35
5	0.77	1.00	0.87

Métricas versión weighted macro

- Cada métrica es ponderada con el peso de su clase.

$$weighted\_macro\_precision = \frac{\sum_{i=1}^{N} \lambda_i * precision_i}{N}$$

$$weighted\_macro\_recall = \frac{\sum_{i=1}^{N} \lambda_i * recall_i}{N}$$

$$weighted\_macro\_f1 = \frac{\sum_{i=1}^{N} \lambda_i * F1_i}{N}$$

N = número total de instancias en el conjunto de evaluación

λ_i = número de instancias de la clase i en el conjunto de evaluación

Métricas versión weighted macro

- **macro P** = (32*1.00 + 19*1.00 + 31*1.00 + 91*1.00 +457*0.77) / 630 = **0.83**
- **macro R** = (32*0.16 + 19*0.05 + 31*0.35 + 91*0.21 + 457*1.00)/ 630 = **0.78**
- **macro F1** = (32*0.27 + 19*0.10 + 31*0.52 + 91*0.35 + 457*0.87) / 630 = **0.42**

	precision	recall	f1-score	número de instancias
1	1.00	0.16	0.27	32
2	1.00	0.05	0.10	19
3	1.00	0.35	0.52	31
4	1.00	0.21	0.35	91
5	0.77	1.00	0.87	457

La librería sklearn ya hace estos cálculos para nosotros

```
1 from sklearn.metrics import classification_report
2 print( classification_report(y_test, predictions))
```

	precision	recall	f1-score	número de instancias
1	1.00	0.16	0.27	32
2	1.00	0.05	0.10	19
3	1.00	0.35	0.52	31
4	1.00	0.21	0.35	91
5	0.77	1.00	0.87	457
accuracy			0.78	630
macro avg	0.95	0.35	0.42	630
weighted avg	0.83	0.78	0.72	630

Métricas versión micro

- El cálculo de las micros se calculará en base a los TP, FP y FN de cada clase.

$$micro\_precision = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + \sum_{i=1}^{N} FP_i}$$

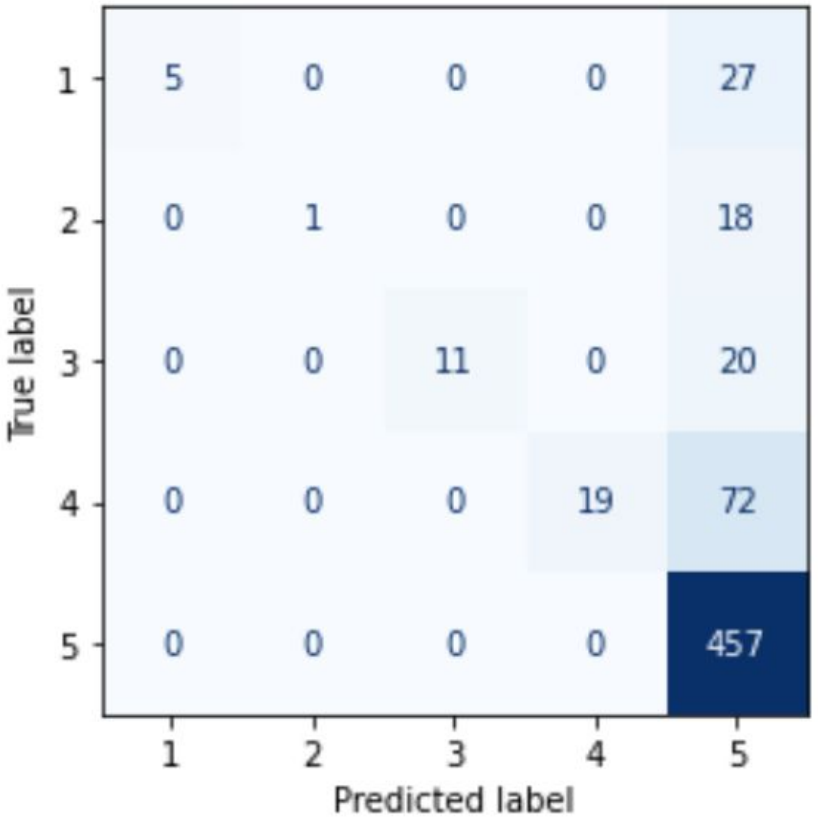
$$micro\_recall = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + \sum_{i=1}^{N} FN_i}$$

$$micro\_F1 = 2 * \frac{micro\_precision * micro\_recall}{micro\_precision + micro\_recall}$$

donde N es el número de clases

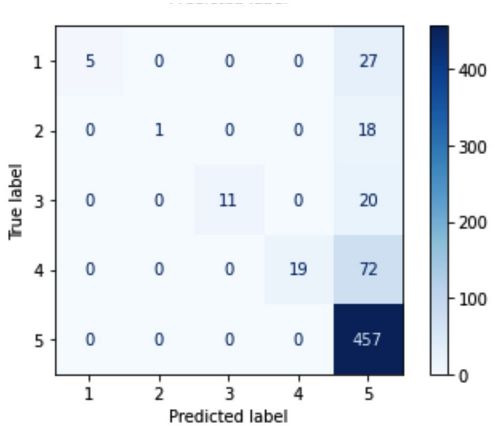
Métricas versión micro

- Supongamos que nuestro algoritmo tiene la siguiente matriz de confusión:



Métricas versión micro

- micro precisión = (5 + 1 + 11 + 19 +457) / (5 + 1 + 11 + 19 +457 + 0 + 0 + 0 + 72 + 20 + 18 + 27) = 493 / 630 = 0.78
- micro recall = (5 + 1 + 11 + 19 +457) / (5 + 1 + 11 + 19 +457) + (27+18+20+72)= 493 / 630 = 0.78
- micro F1 =2 * (0.78 * 0.78) / (0.78+0.78) = 0.78
  Siempre son el mismo valor!!!



¿Por qué la micro P, micro R y micro F1 son el mismo valor?

- Podemos demostrar si P = R?

$$
\begin{aligned}
P &= R \\
\frac{\sum_{l \in L} TP_l}{\sum_{l \in L} (TP_l + FP_l)} &= \frac{\sum_{l \in L} TP_l}{\sum_{l \in L} (TP_l + FN_l)} \\
\sum_{l \in L} (TP_l + FP_l) &= \sum_{l \in L} (TP_l + FN_l) \\
\sum_{l \in L} TP_l + \sum_{l \in L} FP_l &= \sum_{l \in L} TP_l + \sum_{l \in L} FN_l \\
\sum_{l \in L} FP_l &= \sum_{l \in L} FN_l
\end{aligned}
$$

- Bastaría con demostrar que la suma de todos los falsos positivos es igual a la suma de todos los negativos.

En multi-clasificación, micro P, micro R y micro F1 tienen el mismo valor, ¿por qué?

- Supongamos que un texto del test ha sido clasificado por el algoritmo como 5, pero su clase real es 3.
  - Si estamos contabilizando los errores y aciertos de la clase 5, esta predicción, será un falso positivo, porque hemos asignado la clase 5 a un texto cuya clase real es un 3.
  - Por otro lado, si estamos contabilizando los errores y aciertos de la clase 3, la predicción será un falso negativo porque es una instancia de la clase 3 que no se ha clasificado correctamente (se clasificó como 5).
- Por tanto, el **número de falsos positivos y falsos negativos siempre estará equilibrado,** ya que por cada falso positivo para una clase, habrá un falso negativo para otra clase.

En multi-clasificación, micro F1 = accuracy

- En un problema de multi-clasificación, además de FP = FN, también TP = TN.

$$accuracy = \frac{\sum_{i=1}^{N} TP_i + \sum_{i=1}^{N} TN_i}{\sum_{i=1}^{N} TP_i + \sum_{i=1}^{N} TN_i + \sum_{i=1}^{N} FP_i + \sum_{i=1}^{N} FN_i}$$

$$= \frac{2 * \sum_{i=1}^{N} TP_i}{2 * \sum_{i=1}^{N} TP_i + 2 * \sum_{i=1}^{N} FP_i} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + \sum_{i=1}^{N} FP_i}$$

$$= micro\_precision = micro\_recall = micro\_f1$$

Resumen

- En clasificación binaria y clases no balanceadas, la **accuracy** no nos proporciona una información real sobre el rendimiento de nuestro algoritmo.
- En clasificación **binaria** y clases **no balanceadas**, el objetivo es obtener buenos resultados en la **clase minoritaria**.
- En los problemas de **multi-clasificación**, usaremos las métricas macro y las macro-average.

Resumen

- En multi-clasificación, las tres métricas micro precisión, recall y f1 tienen siempre el mismo valor. El número de falsos positivos y falsos negativos está siempre equilibrado.
- En realidad, en multi-clasificación, la micro-F1 es el accuracy.
- Las métricas **micro** son apropiadas para problemas de **multi-etiquetado** (una instancia puede ser clasificada con una o varias etiquetas).

Resumen

- La librería sklearn ya proporciona numerosas funciones para calcular estas métricas!!!.
- Aquí puedes encontrar un listado de estas métricas: https://scikit-learn.org/stable/modules/model_evaluation.html

OpenCourseWare
Procesamiento de Lenguaje Natural con Aprendizaje Profundo,

Gracias!!!

https://github.com/isegura