

# Procesamiento del Lenguaje Natural (PLN)

Mayo 2024

Germán Alonso Lascurain  
[galonso@datavaluemanagement.es](mailto:galonso@datavaluemanagement.es)

# Índice de contenidos

1. ¿Qué es el procesamiento del lenguaje natural?
2. Entendiendo el PLN
3. Diferencia entre PLN basado en reglas y PLN estadístico
4. El problema de la ambigüedad léxica
5. Conocimiento necesario para el PLN
6. Librerías disponibles para PLN
7. Técnicas para la limpieza de textos
8. Técnicas para la normalización de los datos
9. Terminologías usadas en PLN
10. Incrustaciones de palabras o Word Embeddings
11. Top Modeling y Latent Dirichlet Allocation (LDA)

# 1. ¿Qué es el procesamiento del lenguaje natural?

- PLN es un subcampo de la Inteligencia Artificial que se ocupa de las interacciones entre los ordenadores y los lenguajes humanos (naturales).

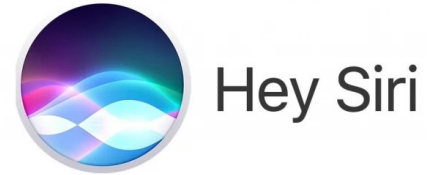
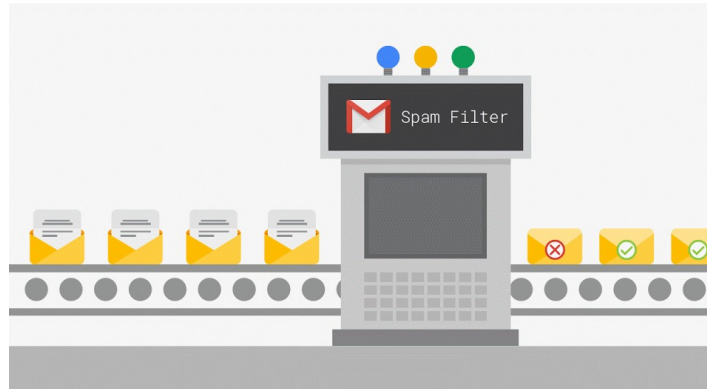
Se usa PLN para crear sistemas de IA que permitan:

- Reconocimiento del habla,
- Resumir documentos,
- Traducción automática,
- Detección de Spam,
- Reconocimiento de Entidades Nombradas,
- Respuesta a preguntas,
- Autocompletar,
- Escritura predictiva, etc.

Hoy en día, la mayoría de nuestros smartphones cuentan con un sistema de reconocimiento de voz. Estos smartphones utilizan PLN para entender el lenguaje natural y dar la respuesta. Además, la mayoría de la gente utiliza ordenadores portátiles cuyo sistema operativo tiene incorporado el reconocimiento de voz.

# 1. ¿Qué es el procesamiento del lenguaje natural?

- Aplicaciones del PLN



## 2. Entendiendo el PLN

- Como humanos, no es una tarea muy difícil realizar el procesamiento del lenguaje natural (PLN), pero aun así, no somos perfectos. A menudo malinterpretamos una cosa por otra y a menudo interpretamos las mismas frases o palabras de manera diferente.
- Por ejemplo, considera las siguientes frases y trata de entender su interpretación de muchas maneras diferentes:
- Ejemplo

Frase: Vi a un estudiante en una colina con un prismático.  
La frase anterior tiene varias interpretaciones:

  - Hay un estudiante en la colina, y lo observé con mi prismático.
  - Hay un estudiante en la colina, y tiene un prismático.
  - Estoy en una colina, y he visto a un estudiante con mi prismático.
  - Estoy en una colina, y vi a un estudiante que tiene un prismático.
  - Hay un estudiante en una colina, y le vi algo con mi prismático.

## 2. Entendiendo el PLN

- A partir de ejemplos como el anterior, podemos observar que el procesamiento del lenguaje **no es "determinista"**:
  - ➔ El mismo lenguaje tiene las mismas interpretaciones, y algo adecuado para una persona puede no serlo para otra.

### 3. Diferencia entre PLN basado en reglas y PLN estadístico

- El Procesamiento del Lenguaje Natural se divide en dos enfoques diferentes:

#### 1. **Procesamiento del lenguaje natural basado en reglas**

Utiliza el razonamiento de sentido común para las tareas de procesamiento.

Por ejemplo,

La temperatura de congelación puede provocar la muerte, o

El café caliente puede quemar la piel de las personas,

Algunas otras tareas de razonamiento de sentido común, etc.

- Sin embargo, este proceso puede llevar más tiempo y requiere un esfuerzo manual.

#### 2. **Procesamiento estadístico del lenguaje natural**

Este tipo de PLN utiliza grandes cantidades de datos y pretende extraer conclusiones de ellos. Para entrenar los modelos de PLN, utiliza algoritmos de aprendizaje automático. Una vez completado el proceso de entrenamiento en grandes cantidades de datos, el modelo entrenado tendrá resultados positivos con la deducción.

## 4. Pros y Cons de ambos métodos

PLN Basado en reglas	PLN Estadístico
Flexible	Fácilmente escalable
Fácil de depurar	Aprende por sí mismo
No requiere mucho entrenamiento	Rápido desarrollo
Comprensión del lenguaje	Gran cobertura
Alta precisión	
Requiere desarrolladores experimentados	Requiere grandes cantidades de datos
El análisis es lento	Complicado de depurar
Cobertura moderada	No entiende el contexto



## 4. El problema de la ambigüedad léxica

- Entendemos por ambigüedad, la **capacidad de ser entendido de más de una manera.**
- 5 tipos de ambigüedad:

### 1. Ambigüedad léxica

La ambigüedad léxica es la que implica la ambigüedad de una sola palabra que puede tener varios significados.

*Se encontraron en el banco de la avenida (en banco como entidad o para sentarse)*

### 2. Ambigüedad sintáctica

La ambigüedad sintáctica se produce cuando una frase se analiza de diferentes maneras.

*El hombre vio a la chica con los prismáticos (la chica tenía unos prismáticos o la vió con ellos)*

## 4. El problema de la ambigüedad léxica

### 3. Ambigüedad semántica

Este tipo de ambigüedad se produce cuando el significado de las propias palabras puede ser malinterpretado. En palabras sencillas, la ambigüedad semántica se produce cuando una oración contiene una palabra o frase ambigua.

*El autobús chocó contra el poste mientras estaba en movimiento (qué estaba en movimiento)*

### 4. Ambigüedad anafórica

La ambigüedad anafórica se produce por el uso de entidades anafóricas en el discurso que sólo pueden resolverse con el contexto.

*¿Dónde viste a Ana? — La vi en el cine (cómo interpretamos La)*

## 4. El problema de la ambigüedad léxica

### 5. Ambigüedad pragmática

Este tipo de ambigüedades se producen cuando el contexto de una frase le da múltiples interpretaciones. En palabras sencillas, podemos decir que estas ambigüedades surgen cuando **el enunciado no es específico.**

*Golpeó el armario con el bastón y lo rompió (no sabemos si se rompió el bastón o el armario)*

## 5. Conocimiento necesario para el PLN

- **Conocimiento fonético y fonológico**

Los conocimientos fonéticos y fonológicos son esenciales para los sistemas basados en el habla, ya que se ocupan de **cómo se relacionan las palabras con los sonidos que las realizan.**

- **Conocimiento morfológico**

Necesario para **conocer los patrones de formación de las palabras (morfemas).**

- **Conocimiento sintáctico**

Necesario para conocer cómo las palabras pueden unirse para formar oraciones correctas.

- **Conocimiento semántico**

Necesario para entender el **significado** de las palabras y oraciones.

- **Conocimiento pragmático**

Necesario para conocer cómo se utilizan las oraciones en diferentes situaciones.

## 5. Conocimiento necesario para el PLN

- **Conocimiento del discurso**

El discurso se refiere a las oraciones conectadas. Incluye el estudio de trozos de lenguaje que son más grandes que una sola frase. **Implica el entendimiento de los vínculos entre frases inmediatamente anteriores que afectan a la interpretación de la frase siguiente.**

- **Conocimiento de las palabras**

Esencial para mejorar la comprensión de la lengua.

## 6. Librerías disponibles para PLN

- **NLTK**

Muy sencilla de usar, aunque no aconsejable para entornos de producción (sí para educación e investigación).

- Funcionalidades:
  - Tokenización.
  - Etiquetado de partes del lenguaje (POS).
  - Reconocimiento de Entidades Nombradas (NER).
  - Clasificación.
  - Análisis de sentimiento.
  - Paquetes de chatbots.
  - Aplicaciones
  - Sistemas de recomendación.
  - Análisis de sentimiento.
  - Construcción de chatbots.



Natural Language Analysis  
with Python NLTK

## 6. Librerías disponibles para PLN

- **spaCy**

De código abierto, está diseñada para ser de naturaleza rápida y lista para la producción. Se centra en proporcionar software para el uso de producción.

- Funcionalidades:
  - Tokenización.
  - Etiquetado de partes del lenguaje (POS).
  - Reconocimiento de entidades con nombre (NER).
  - Clasificación.
  - Análisis de sentimientos.
  - Análisis de dependencia.
  - Vectores de palabras.
  - Aplicaciones
  - Autocompletar y autocorregir.
  - Análisis de reseñas.
  - Resumir.

The logo for spaCy, featuring the word "spaCy" in a blue, sans-serif font. The "spa" is in a lighter blue, and "Cy" is in a darker blue.

## 6. Librerías disponibles para PLN

- **gensim**

Gensim, un framework de PNL en Python, se utiliza generalmente en el modelado de temas y la detección de similitudes. No es una biblioteca de PNL de propósito general, pero maneja muy bien las tareas que se le asignan.

- Funcionalidades
  - Análisis semántico latente.
  - Factorización de matrices no negativas.
  - TF-IDF.
  - Aplicaciones
    - Conversión de documentos en vectores.
    - Búsqueda de similitudes textuales.
    - Resumen de textos.
    - Para más información, consulte la documentación oficial



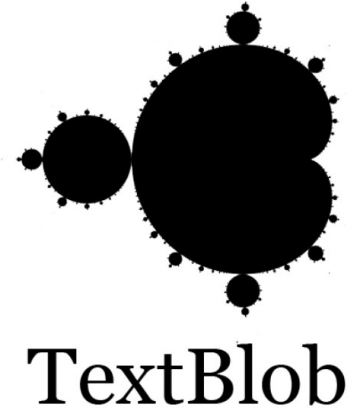


## 6. Librerías disponibles para PLN

### ● **TextBlob**

TextBlob es una librería de Python diseñada principalmente para el procesamiento de datos textuales.

- Funcionalidades
  - Etiquetado de parte del discurso.
  - Extracción de frases sustantivas.
  - Análisis de sentimiento.
  - Clasificación.
  - Traducción de idiomas.
  - Análisis sintáctico.
  - Integración de redes de palabras.
- Aplicaciones
  - Análisis de sentimientos.
  - Corrección ortográfica.
  - Traducción y detección de idiomas.



## 6. Librerías disponibles para PLN



- **Stanza**

Stanza es una colección de herramientas para el análisis lingüístico de muchas lenguas. Empezando por el texto en bruto hasta el análisis sintáctico y el reconocimiento de entidades, Stanza aporta modelos de PLN de última generación.

- Funcionalidades
  - Red neuronal completa para el análisis de texto robusto, incluyendo tokenización, expansión de tokens de múltiples palabras (MWT), lematización, etiquetado de parte del habla (POS) y características morfológicas, análisis sintáctico de dependencias y reconocimiento de entidades con nombre;
  - Modelos neuronales preformados que soportan más de 80 idiomas

## 7. Técnicas para la limpieza de textos

- Previo al análisis de textos, y como en cualquier proyecto de machine learning o deep learning son necesarios una serie de pasos para mejorar la calidad de los datos. En el caso de textos, necesitamos preprocesarlos y limpiarlos.
  - **Convertir el texto a minúsculas**
  - **Eliminación de las etiquetas HTML**

Cuando recogemos los datos de texto mediante técnicas como el web scraping o el screen scraping, éstos contienen mucho ruido. Por lo tanto, podemos eliminar las etiquetas HTML innecesarias y conservar la información textual útil para los siguientes pasos de nuestro análisis.
  - **Eliminación de los caracteres acentuados**

Normalmente, en cualquier dato de texto, puede haber caracteres o letras acentuadas, especialmente si sólo queremos analizar el idioma inglés. Por lo tanto, tenemos que convertir y normalizar estos caracteres en caracteres ASCII. Por ejemplo, convirtiendo é en e.

## 7. Técnicas para la limpieza de textos

### ➤ **Expansión de las contracciones**

Podemos decir que las contracciones son versiones abreviadas de palabras o sílabas. O más sencillamente, una contracción es una abreviatura utilizada para representar una secuencia de palabra

### ➤ **Eliminación de caracteres especiales y símbolos no alfanuméricos**

Normalmente, para eliminarlos podemos utilizar expresiones regulares sencillas.

### ➤ **Corrección de errores tipográficos**

Para hacer el mapa correcto se necesita un diccionario mediante el cual mapeamos las palabras a su forma correcta basándonos en la similitud.

### ➤ **Normalización**

A menudo, los datos de texto contienen palabras o frases que no están presentes en ningún diccionario estándar como los acrónimos.

## 8. Técnicas para la normalización de los datos

- **Stemming – Extracción raíz**

El stemming consiste en **quitar y reemplazar sufijos y prefijos de la raíz de la palabra.**

El stemming es una forma rápida pero un poco torpe de tomar las raíces.

Para eso utilizaremos SnowballStemmer de NLTK. El algoritmo es bastante simple y se puede ver en este link. Cada idioma tiene sus reglas, en inglés NLTK utiliza el algoritmo de stemming de Porters, en Español toma algunas reglas que, en resumen, remueven diversos sufijos que se atribuyen a acciones (ar, er, ir, ía, en, es, etc), terminaciones plurales, generos y otros →

Palabras: reír, rió, ríe, risa

Todas las palabras anteriores se convertirán en risa, que es su raíz.

Una vez pasada la palabra por el algoritmo de stemming devuelve un **stem**, que **es la palabra sin la terminación o sufijos.**

- Ayuda a muchas aplicaciones como:

- Clasificar textos
- Agrupar textos, y
- Recuperación de información

## 8. Técnicas para la normalización de los datos

- **Lematización**

La lematización es un proceso lingüístico que consiste en, dada una forma flexionada (es decir, en plural, en femenino, conjugada, etc), hallar el lema correspondiente.

**El lema de una palabra es la palabra que nos encontraríamos como entrada en un diccionario tradicional:** singular para sustantivos, masculino singular para adjetivos, infinitivo para verbos.

Por ejemplo, decir es el lema de **dije**, pero también de **diré o dijéramos**; **guapo** es el lema de **guapas**; **mesa** es el lema de **mesas**.

- Lematizar implica estandarizar, desambiguar, segmentar y, en caso de usar programas de lematización automática, también etiquetar.

## 9. Terminologías usada en PLN

- **Documento**

Conjunto de datos con la información a procesar.

- **Corpus**

Colección de todos los documentos presentes en nuestro conjunto de datos.

- **Característica (Feature)**

Cada palabra única del corpus se considera una característica.

- **Ejemplo:**

Frases:

- El perro odia al gato. Le encanta salir a jugar.
- Al gato le encanta jugar con una pelota.

Podemos construir un corpus a partir de los 2 documentos anteriores simplemente combinándolos.

→ Corpus = “El perro odia al gato. Le encanta salir a jugar. Al gato le encanta jugar con una pelota.”

→ Features = ['El', 'perro', 'odia', 'al', 'gato', 'le', 'encanta', 'salir', 'jugar'] ← VECTOR DE CARACTERÍSTICAS

Nota: Eliminamos la 'a' por tener un sólo caracter

## 10. Incrustaciones de palabras o Word Embeddings

- Es una técnica que consiste en **representar palabras con vectores de números**.
- Ello nos permite mejorar significativamente las tareas de descubrimiento de conocimiento y recomendación de contenido.

*En esta nube, el lugar donde se encuentran las palabras es importante ya que determina su significado. Como vemos en la imagen arriba, la cual es una visualización de esta nube, tenemos en una sección palabras asociadas a animales y en otra encontramos palabras asociadas a alimentos.*

*Dentro del grupo de palabras de animales, también vemos que las palabras asociadas a mascotas se encuentran más cercanas entre ellas, ej. gato y perro, en comparación con las palabras que hacen referencia a animales salvajes, ej. jirafa y elefante.*

*No solo eso, en la imagen podemos observar otro tipo de patrones, por ejemplo, vemos que la distancia entre las palabras hombre y rey, es la misma que la distancia entre las palabras mujer y reina.*





## 10. Incrustaciones de palabras o Word Embeddings

- **Tipos de word embeddings**

1. Basadas en frecuencia o en la estadística
2. Basadas en la predicción

Para aplicar este tipo de incrustaciones podemos aplicar las siguientes técnicas

- One-Hot Encoding (OHE)
- Count Vectorizer
- Bag of Words (BOW)
- N-gramas
- Term Frequency-Inverse Document Frequency (Vectorización TF-IDF)

## 10. Incrustaciones de palabras o Word Embeddings

- **One Hot Encoding (OHE)**

En esta técnica, **representamos cada palabra única en el vocabulario** estableciendo un token único con valor 1 y el resto 0 en otras posiciones del vector.

En palabras sencillas, la representación vectorial de un vector codificado OHE se representa en forma de 1 y 0, donde 1 representa la posición en la que existe la palabra y 0 en todas las demás.

Frase: I am teaching NLP in Python

Diccionario: ['I', 'am', 'teaching', 'NLP', 'in', 'Python', 'resto de palabras del diccionario']

Vector OHE para NLP: [0,0,0,1,0,0, ...]

**El vector OHE, también se denomina vector de características.**



1	0	0	1	1
0	0	1	1	1
1	1	0	1	0

## 10. Incrustaciones de palabras o Word Embeddings

- **One Hot Encoding (OHE). Desventajas**

- ➔ El Tamaño del vector es igual al conteo de palabras únicas en el vocabulario.
- ➔ No transmite información sobre el contexto (No capta las relaciones entre las distintas palabras)

## 10. Incrustaciones de palabras o Word Embeddings

### ● Count Vectorizer

Es una de las formas más sencillas de hacer vectorización de textos.

	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

Word Vector (Passage Vector) →

Document Vector ↑

1. **Crea una matriz de términos del documento**, que es un conjunto de variables ficticias que indican si una determinada palabra aparece en el documento.
2. **Registramos la frecuencia con la que aparecen los términos**, y las columnas están dedicadas a cada palabra del corpus.
3. Se crea una matriz de términos del documento en la que **las celdas individuales denotan la frecuencia de esa palabra en un documento concreto**, lo que también se conoce como **frecuencia de términos**, y las columnas están dedicadas a cada palabra del corpus.

## 10. Incrustaciones de palabras o Word Embeddings

- **Bag of Words (BOW)**

Esta técnica de vectorización **convierte el contenido del texto en vectores de características numéricas.**

**Bag of Words se crea un vector donde cada entrada representa la frecuencia de una palabra en el texto.**

La ordenación de las palabras generalmente se ignora, lo que significa que la información sobre la estructura y el orden se pierde.

Requiere de 2 operaciones:

1. Tokenización
2. Creación de vectores de características

```
Texto 1: "Este es un ejemplo."  
Texto 2: "Otro ejemplo es este."
```

```
Bag of Words:  
{ "este": 2, "es": 2, "un": 1,  
  "ejemplo": 2, "otro": 1 }
```

## 10. Incrustaciones de palabras o Word Embeddings

- **Bag of Words (BOW). Inconvenientes**
  1. No preserva el orden de las palabras
  2. No permite hacer inferencias útiles para tareas posteriores de PLN

## 10. Incrustaciones de palabras o Word Embeddings

### ● Vectorización de N-gramas

La vectorización de n-gramas en Procesamiento del Lenguaje Natural (PLN) es una técnica que extiende el enfoque de Bag of Words (BoW) y Count Vectorizer para **capturar secuencias de palabras contiguas de longitud n (n-gramas) en lugar de solo palabras individuales**. Un n-grama es una secuencia continua de n elementos de una cadena de texto, que generalmente son palabras.

1. Genera una matriz de términos del documento, y cada celda representa un recuento.
2. Las columnas representan todas las columnas de palabras adyacentes de longitud n.
3. La vectorización por recuento es un caso especial de N-Gram donde  $n=1$ .
4. Los N-gramas consideran la secuencia de n palabras en el texto; donde n es (1,2,3.. ) como 1-grama, 2-grama. para el par de tokens. A diferencia de BOW, mantiene el orden de las palabras.

Ejemplo:

*Estoy estudiando Python NLP* → Tiene 4 palabras y  $n=4$

*Si  $n=2$ , pej bigrama* → ['Estoy estudiando', 'estudiando Python', 'Python NLP']

*Si  $n=3$ , pej trigram* → ['Estoy estudiando Python', 'estudiando Python NLP']

*Si  $n=4$ , pej cuatrigrama* → ['Estoy estudiando Python NLP']

## 10. Incrustaciones de palabras o Word Embeddings

- **Vectorización de N-gramas**

El procedimiento general es **tomar el valor de N tan pequeño como sea posible** y que ofrezca información suficiente.

**Valores demasiado pequeños** ofrecen pocas **características**, pero más **específicas del texto** como podría ser vocabulario único. En conjuntos de datos grandes, puede implicar trabajar con conjuntos de datos extensos.

**Valores demasiado grandes** de N, capturaremos **relaciones más complejas y de largo alcance**. Lo usaremos para detectar estructuras gramaticales complejas. **Permite capturar el contexto** y puede ayudar a predecir la siguiente palabra en un contexto. Todo ello a costa de mayor complejidad computacional (mayor cantidad de combinaciones posibles).



## 10. Incrustaciones de palabras o Word Embeddings

- **Vectorización de N-gramas. Inconvenientes**

1. Tiene demasiadas características (N-gramas)
2. Debido que devuelve demasiados N-gramas, el conjunto es demasiado disperso y computacionalmente costoso.
3. No es fácil elegir el valor de N

## 10. Incrustaciones de palabras o Word Embeddings

- **Vectorización TF-IDF**

1. **TF** → Frecuencia de término o palabra. **Frecuencia con la que ocurre la palabra en un documento.**

$$tf(t, d) = \frac{n_t}{\sum_k n_k}$$

donde:  $n_t$  corresponde al número de veces que aparece un término en el documento

Sum  $n_k$  corresponde al total de términos que hay en el documento.

Como cada documento puede tener una longitud diferente, un término puede aparecer más en documentos grandes que en cortos y todos los términos tienen la misma importancia.

## 10. Incrustaciones de palabras o Word Embeddings

- **Vectorización TF-IDF**

2. **Idf** → Frecuencia inversa de documento. **Permite evaluar cómo de importante es un determinado término en el documento.**

Este valor no varía entre documentos, sólo entre palabras.

Necesitamos saber cuántos documentos hay en nuestro corpus (representados en la fórmula con la letra D) y en cuántos aparece la palabra.

$$idf(t, D) = \log \frac{|D|}{|\{ d_i \in D \mid t \in d_i \}|}$$

Elementos como 'el', 'la', 'eso', ... pueden aparecer muchas veces en un documento, sin embargo tienen poca importancia.

## 10. Incrustaciones de palabras o Word Embeddings

- **Vectorización TF-IDF**

**3. Frecuencia inversa del documento: Valora cuán común o raro es un término en el conjunto de documentos.** Cuanto más raro, mayor será su frecuencia inversa.

- Un **valor alto** de TF-IDF para un término en un documento indica que el término es **importante para ese documento**, ya que es frecuente en ese documento y poco común en otros documentos del conjunto.
- Un **valor bajo** de TF-IDF sugiere que el término es **común en el conjunto de documentos** y, por lo tanto, no es discriminativo para ese documento específico.

Supongamos que tenemos un documento con 100 palabras, donde la palabra gato aparece 3 veces.

$$TF = 3 / 100 = 0.03$$

Ahora supongamos que tenemos 10 millones de documentos y gato aparece en 1000 de ellos.

$$idf = \log (10.000.000 / 1.000) = 4$$

El peso que se asignaría a la palabra gato sería:

$$0.03 * 4 = 0.12$$

## 10. Incrustaciones de palabras o Word Embeddings

### ● Vectorización TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
corpus = ["This burger is very tasty and affordable.", " This burger is not tasty and is affordable.", "This burger is
vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform(corpus)
feature_names = vectorizer.get_feature_names()
print(f"Feature names \n{feature_names}")
matrix = vectors.todense()
list_dense = matrix.tolist()
df=pd.DataFrame(list_dense, columns=feature_names)
print(df)
```

Feature names

['affordable', 'and', 'burger', 'delicious', 'is', 'not', 'tasty', 'this', 'very']

	affordable	and	burger	delicious	is	not	tasty	\
0	0.414896	0.414896	0.322204	0.000000	0.322204	0.000000	0.414896	
1	0.346117	0.346117	0.268791	0.000000	0.537582	0.455102	0.346117	
2	0.000000	0.000000	0.282851	0.478909	0.282851	0.000000	0.000000	

	this	very
0	0.322204	0.414896
1	0.268791	0.000000
2	0.282851	0.728445

## 11. Top Modeling y Latent Dirichlet Allocation (LDA)

El 'Topic Modeling' es un tipo de modelado estadístico para encontrar los diferentes temas contenidos en documentos.

Consideraremos documentos en sentido amplio, pudiendo éstos ser una frase, un párrafo, un documento, ...

- El algoritmo LDA nos permite clasificar un determinado texto en un documento, dentro de un tema.
- Genera un tema por documento y asigna palabras a un tema específico.

## 11. Top Modeling y Latent Dirichlet Allocation (LDA)

- Pasos a aplicar:
  - **Tokenización del contenido.** Dividiendo el texto en sentencias y las sentencias en palabras. Aquí aplicaremos (entre otros), eliminación de signos de puntuación.
  - **Eliminación de palabras** con menos de 3 caracteres.
  - **Eliminación de stopwords.**
  - **Lematización** de palabras (obtención del lema, singular para sustantivos, masculino singular para adjetivos e infinitivo para verbos).
  - **Stemming** de palabras (obtener la raíz de la palabra).

## 11. Top Modeling y Latent Dirichlet Allocation (LDA)

- **Funcionamiento del algoritmo.**

- 1) Definición del Problema:

- LDA **se utiliza para modelar colecciones de textos**, como documentos, artículos o mensajes. El objetivo es descubrir temas subyacentes en estos textos y entender cómo se distribuyen estos temas en los documentos.

- 2) Representación del Texto:

- **Cada documento en la colección se representa como una bolsa de palabras** (bag-of-words). En esta representación, se ignora el orden de las palabras y solo se considera la frecuencia de cada palabra en el documento.

- 3) Asignación de Tópicos:

- **LDA asume que cada documento es una mezcla de varios temas, y cada palabra en el documento está asociada con uno de esos temas.** El objetivo es descubrir estas asignaciones de temas. Por ejemplo, un artículo de noticias podría ser una mezcla de temas como política, economía y deportes.



## 11. Top Modeling y Latent Dirichlet Allocation (LDA)

- **Funcionamiento del algoritmo.**

- 4) Modelo Probabilístico:

- LDA utiliza un modelo probabilístico generativo para explicar cómo se generan los documentos. Según el modelo, se **asume que hay un número fijo de temas en el corpus y que cada documento es una mezcla de estos temas**. Además, se asume que cada palabra en un documento se elige de uno de los temas de esa mezcla.

- 5) Distribución de Dirichlet:

- Para modelar la mezcla de temas en un documento y la distribución de palabras en un tema, LDA utiliza la Distribución de Dirichlet. **La Distribución de Dirichlet modela la probabilidad de que un documento contenga ciertos temas y la probabilidad de que una palabra pertenezca a cierto tema.**

- 6) Inferencia de temas:

- La inferencia en LDA se realiza mediante métodos de inferencia probabilística (bayes, gibbs, ...). **El objetivo es calcular la probabilidad posterior de los temas y las palabras dados los documentos observados.** Esto se realiza utilizando algoritmos como el Muestreo de Gibbs o el Muestreo de Metrópolis-Hastings.

## 11. Top Modeling y Latent Dirichlet Allocation (LDA)

- **Funcionamiento del algoritmo.**

- 7) Entrenamiento del Modelo:

- Durante el entrenamiento del modelo, se ajustan los parámetros del modelo para maximizar la probabilidad de observar los documentos en el corpus.

- 8) Resultados:

- Después de entrenar el modelo, se pueden examinar los resultados para entender qué temas se encuentran en los documentos y cómo se distribuyen en el corpus. **Cada documento tiene una distribución de probabilidad sobre los temas, y cada tema tiene una distribución de probabilidad sobre las palabras.**

## 11. Top Modeling y Latent Dirichlet Allocation (LDA)

- **Uso de los resultados**

- Descubrir los tópicos principales en el corpus.
- Clasificar documentos en función de los tópicos predominantes.
- Encontrar documentos relacionados por tópico.
- Resumir grandes conjuntos de documentos al identificar los tópicos clave.
- Es importante destacar que **LDA asume que los documentos son mezclas de tópicos y que estas mezclas generan las palabras en los documentos.** Sin embargo, **los tópicos** mismos son latentes, lo que significa que **no se conocen de antemano; se descubren durante el proceso de entrenamiento del modelo.** El algoritmo LDA se basa en el principio de Dirichlet para modelar estas relaciones entre tópicos, palabras y documentos.

## 12. **Corpus**es para practicar

[http://www.nltk.org/nltk\\_data/](http://www.nltk.org/nltk_data/)