

TECHTALENTCENTER

INSPIRING
INNOVATION.
EMPOWERING
TALENT.



UNIVERSITAT POLITÈCNICA
DE CATALUNYA

School of Professional
& Executive Development

Text analytics (I)
Pedro González
Carlos Escolano

Barcelona, Mayo 2023



Pedro González Alonso
Head of Data Science / CTO at nixi1.com
pedrojavi.gonzalez@gmail.com



Computer Sciencie engineer FIB UPC
MIRI master FIB-UPC
AI with Deep Learning – UPC School
MBA – ESADE Business School

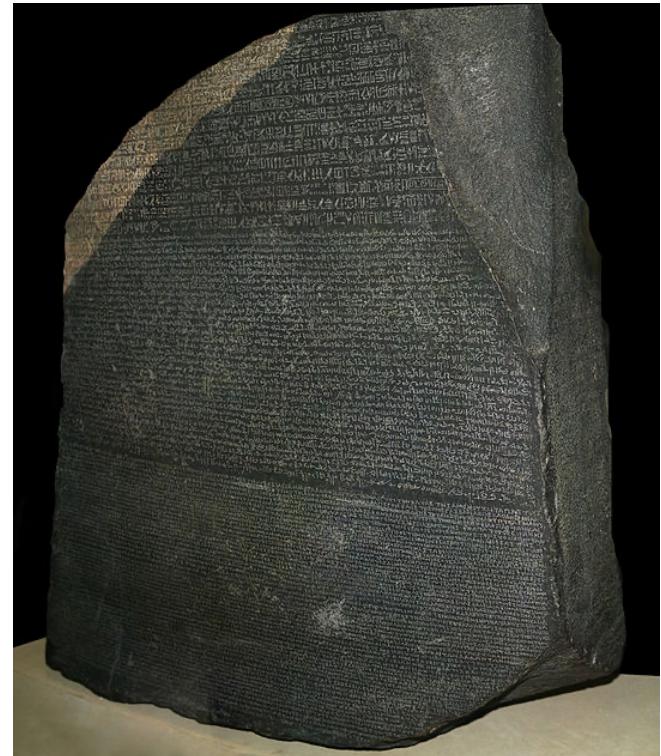


Carlos Escolano
PhD Candidate TSC UPC
carlos.escolano@tsc.upc.edu

Computer science engineer FIB-UPC
MAI master FIB-UPC

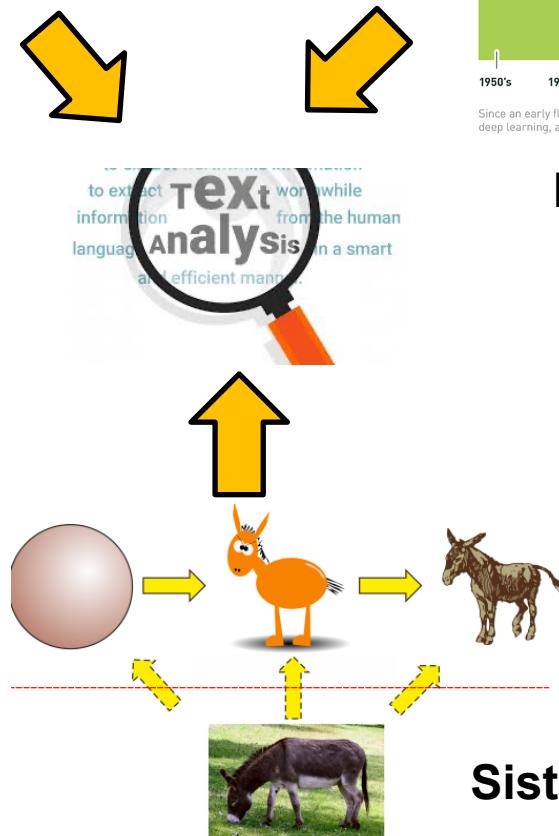
Contenidos

- Contexto
- Problemas y tareas
- Leyes estadísticas en texto
- Preproceso
- Vector models
- Cálculo de similaridades
- Topic Modeling
- Sistemas conversacionales, un ejemplo
- Ejemplo: Clasificación de texto

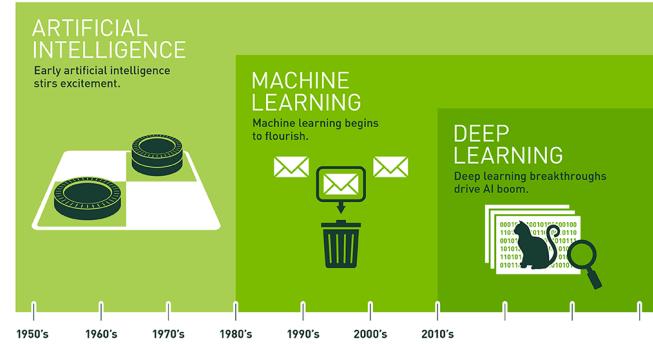




Information Retrieval



Contexto



Inteligencia Artificial

Sistemas basados en reglas



Problemas y tareas

- Búsqueda y recuperación de información
- Q&A
- Clasificación de texto
- Sentiment analysis
- Categorización automática / topic modelling
- Traducción automática
- Speech recognition
- Named Entity Recognition (NER)
- Resolución de correferencias
-



Leyes estadísticas

Dos leyes empíricas importantes en el en estudio estadístico:

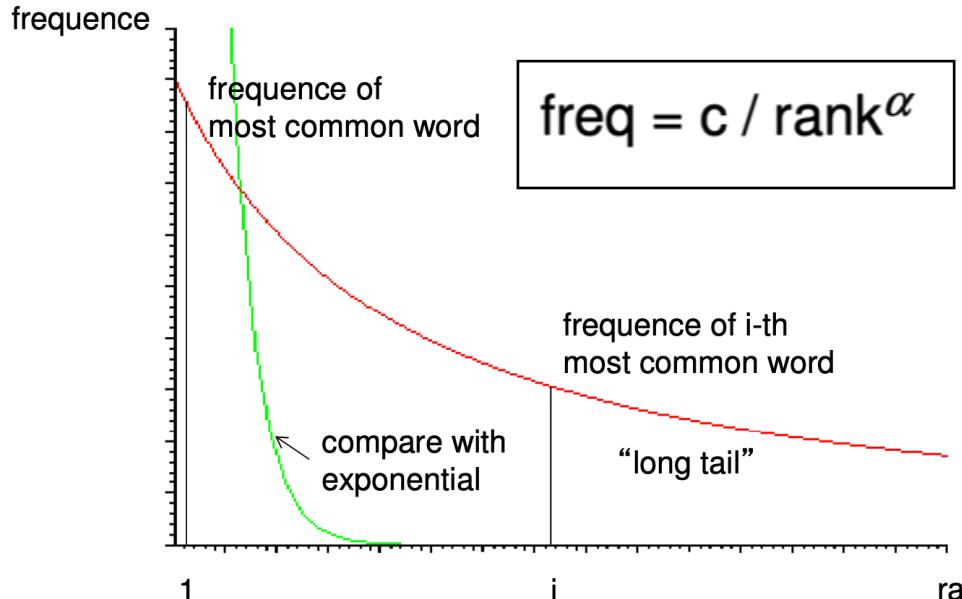
- Ley de Zipf. ¿Cómo se distribuyen las frecuencias de los términos?
- Ley de Heap. ¿Cuántos términos diferentes podemos esperar en una colección de textos?

Leyes estadísticas

Ley de Zipf. ¿Cómo se distribuyen las frecuencias de los términos?

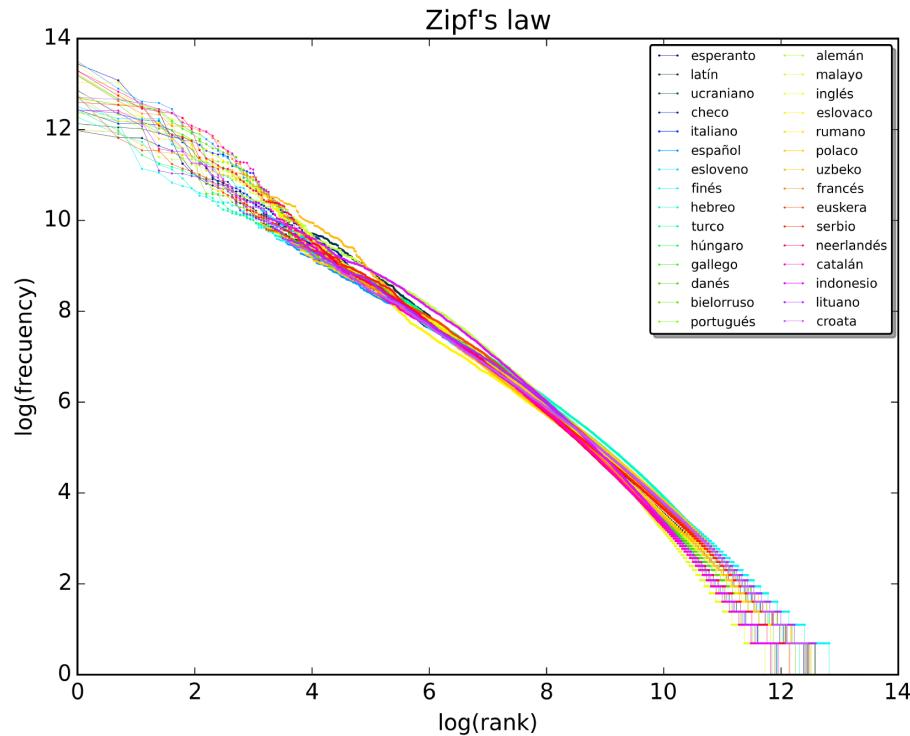
- Formulada por **George Kingsley Zipf**, en los 40's
- La frecuencia del i-ésimo término más frecuente en un corpus, es proporcional a $1/i$
- En otras palabras, el producto Frecuencia(término i) * i, es constante => ambas magnitudes siguen una ley potencial (power law)

- Esta relación se caracteriza por presentar "long tail". Muchos términos tienen una frecuencia baja, pero su masa acumulada es considerable



Leyes estadísticas

Ley de Zipf. ¿Cómo se distribuyen las frecuencias de los términos?



$$\text{freq} = c / \text{rank}^\alpha$$



$$\log(\text{freq}) = \log(c) - \alpha \log(\text{rank})$$

Imagen de Wikipedia. Ranking vs frecuencia de las 10 millones de palabras más frecuentes en 30 wikipedias

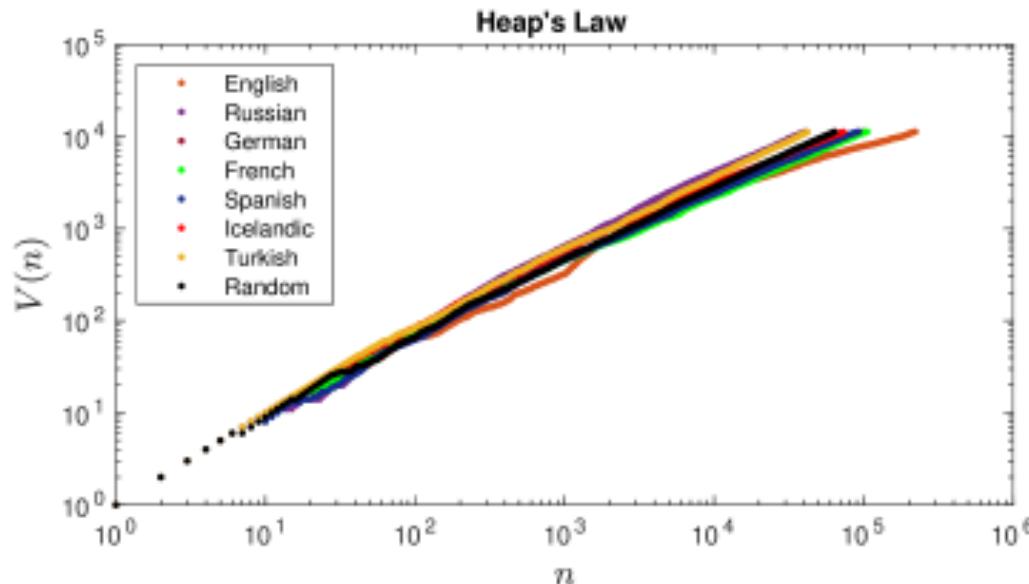
Leyes estadísticas

Ley de Heap. ¿Cuántos términos diferentes podemos esperar en una colección de textos?

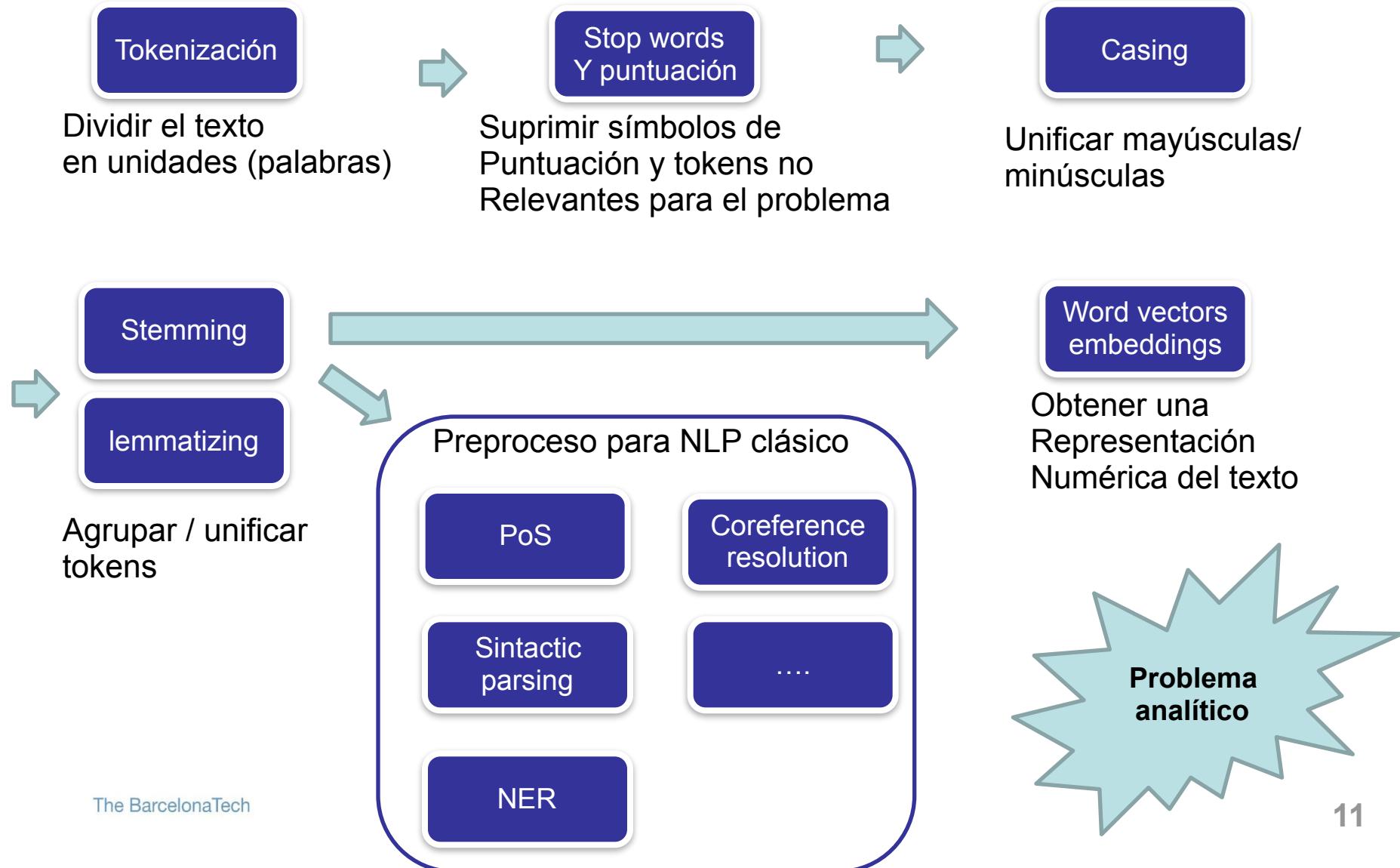
- El número de términos diferentes en un texto de N palabras es proporcional a N^B :

$$\text{Términos} = k * N^B$$

($k = 30..10$ y $B=0.5..0.8$ en textos en inglés)



Preprocessing





Preprocessing

- **Tokenización:** Separar las sentencias en unidades constituyentes.
- **Expansión n-grams:** Agrupar tokens en n-grams de diferente n (muy útil para NER o coreference resolution)
- **Stop words:** Eliminar los tokens que no son relevantes para la resolución del problema (dependiente del problema)
- **Casing:** Unificar mayúsculas y minúsculas. Por “fuerza bruta” (todo a minúsculas) o utilizando modelos que aprendan que palabras se escriben con mayúsculas en un corpus)
- **Stemming.** Reducir las palabras a su raíz (computation, computer -> comput).
 - Dependiente del idioma
 - No necesariamente obtiene una palabra válida del diccionario
 - No precisa contexto, se puede calcular a partir de la palabra aplicando las reglas asociadas al idioma



Preprocessing

- **Lemmatizing:** Calcular la forma común de una palabra flexionada (conjugación, género, numero,).
 - Dependiente del contexto.
 - Requiere diccionarios y modelos propios del idioma/lenguaje.
 - Siempre se obtiene una palabra válida en el diccionario.
- **PoS:** Identificar las clases sintácticas de las palabras. Nombres, determinantes, verbos,..
- **NER:** Identificar entidades propias (localizaciones, personas, etc). Requieren bases de datos propias del dominio de aplicación y/o modelos entrenados para hacer NER
- **Sintactic Parsing:** Obtener el árbol sintáctico acorde con la definición de una determinada gramática formal <http://nlp.stanford.edu:8080/parser/>



Preprocessing

- **Correference resolution:** Identificar menciones a la misma entidad en un texto. (Pedro Sánchez anuncia el fin del estado de alarma. El presidente, informó a los medios ayer)
- **Vector models:** Obtener una representación numérica del texto

Vector models

Bag of words: Una sentencia se representa como un vector de valores enteros de dimensión igual al número de tokens diferentes del corpus.

Cada elemento representa la aparición o no (binario) o el número de apariciones (Word count) del token correspondiente en la sentencia.

Produce representaciones muy poco densas (sparse vectors) dado que la mayoría de tokens no aparece en una misma sentencia.

the dog is on the table

0	0	1	1	0	1	1	1
are	cat	dog	is	now	on	table	the

Fuente: <https://machinelearnings.co/text-classification-using-neural-networks-f5cd7b8765c6>



Vector models

TF-IDF: En lugar de codificar valores binarios o enteros para cada token, se calcula un valor continuo que incorpora la relevancia de cada token en la sentencia en la que aparece. Incorpora la frecuencia de aparición del token en el corpus, y en el documento/sentencia que se está analizando.

Para un corpus de D documentos, para cada documento d_i , se calcula el vector $td\text{-}idf_i$ para cada término w_j del documento d_i

$$tf\text{-}idf(w_j, d_i) = tf(w_j, d_i) * idf(w_j, D)$$

Donde:

$$tf(w_j, d_i) = freq(w_j, d_i) / max(freq(w_k, d_i))$$

term frequency: frecuencia del término w_i en el documento d_i normalizada por la frecuencia del término con frecuencia máxima. En ocasiones se utiliza en el denominador el número total de términos en el documento.

$$idf(w_j, D) = \log(D/(docs(w_j)))$$

Inverse Document Frecuence: log del total de documentos dividido por el número de documentos en los que aparece el término w_j . La versión “smooth” (suavizada) de IDF, suma 1 al denominador y suma uno tras calcular el logaritmo

Vector models

```
[107]: ex_tfidf = [  
    "la lechuga estaba encima de la mesa",  
    "el martillo estaba encima de la mesa",  
    "el diario estaba encima de la mesa"  
]
```

	lechuga	martillo	diario	mesa	encima
TF(0)	0,333	0,000	0,000	0,333	0,333
TF(1)	0,000	0,333	0,000	0,333	0,333
TF(2)	0,000	0,000	0,333	0,333	0,333
IDF	0,477	0,477	0,477	0,000	0,000
IDF (smooth)	1,176	1,176	1,176	0,875	0,875
TFIDF(1)	0,392	0,000	0,000	0,292	0,292
TFIDF(2)	0,000	0,392	0,000	0,292	0,292
TFIDF(3)	0,000	0,000	0,392	0,292	0,292

Aunque mesa, tiene la misma frecuencia que el resto de términos, aparece en todos los documentos, por lo que su relevancia en un documento determinado es menor que aquellos que aparecen solo en dicho documento.



Vector models

Los modelos bag of Word generan vectores de codificación cuya dimensión depende del número de tokens diferentes en el corpus.

En algunos casos nos interesaría poder controlar la dimensionalidad del los vectores de clasificación. Por ejemplo, en un problema de clasificación supervisada usando clasificadores clásicos. Para ello podemos utilizar varias estrategias:

- Seleccionar un número de tokens determinado para codificar e ignorar el resto.
- Utilizar el “hashing trick”. Usar una función de hash para controlar la dimensión de la tabla de mappings => colisiones (varios tokens comparten la misma codificación)



Cálculo de similaridades

El índice de jaccard, mide la similaridad de dos conjuntos mediante el cociente de las cardinalidades de la intersección y la unión:

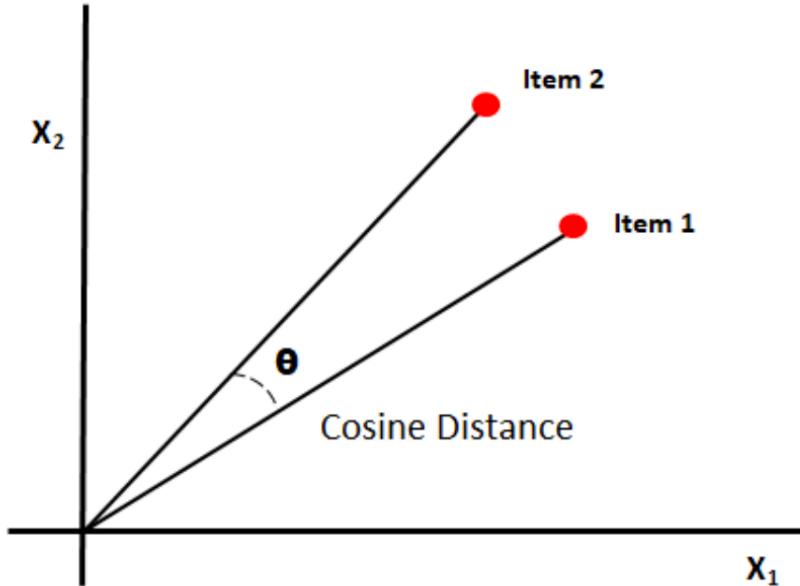
Sean p , q dos representaciones de vector model binario de dos sentencias, el índice de Jaccard se define como:

$$\text{Jaccard}(d, q) = \frac{|d \cap q|}{|d \cup q|}$$

$$|d \cap q| = \sum_{t=1}^T d_t \cdot q_t$$

Cálculo de similaridades

Cosine Distance/Similarity



Dado un modelo de word vector, la distancia de coseno es una métrica normalizada [0,1] que nos da una medida directa de la semejanza de dos sentencias

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$



Clasificación

Un problema de clasificación de textos no es diferente de otros problemas de clasificación supervisada vistos durante el curso:

- Un dataset de entrada con textos a clasificar (spam/no spam, tema, opinión positiva / negativa,...)
- Un conjunto de clases conocidas $c = \{C_1, C_2, \dots, C_p\}$
- Una función $f(x)$ que mapea documentos a clases.



Clasificación

Clasificador bayesiano ingenuo (Naïve Bayes)

- Para un documento d con tokens (x_1, x_2, \dots, x_n) y una clase c :

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

- Asignar un documento a una única clase usando maximum likelihood:

$$\begin{aligned} c_{ML} &= \arg \max_{c \in C} P(c|d) \\ &= \arg \max_{c \in C} \frac{P(d|c)P(c)}{P(d)} \\ &= \arg \max_{c \in C} P(d|c)P(c) \\ &= \arg \max_{c \in C} P(x_1, x_2, \dots, x_n | c)P(c) \end{aligned}$$



Clasificación

Asunciones

- Los modelos bag of words asumen que la posición de los tokens no es importante
- Independencia de las features
- Misma contribución de todas las x_i al resultado

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



Clasificación

Funcionamiento

- Calculamos $P(C_j)$ mediante su frecuencia en el corpus

$$P(C_j) = \frac{|docs_j|}{|docs|}$$

- Calculamos $P(w_i|c_j)$ para los términos w_i

$$P(w_k|c_j) = \frac{count(w_k, c_j) + 1}{\sum(count(w, \textcolor{red}{c}) + 1)}$$

número de documentos en los que w está presente cuando la clase del documento es c

número de documentos en los que w está presente



Bases de datos

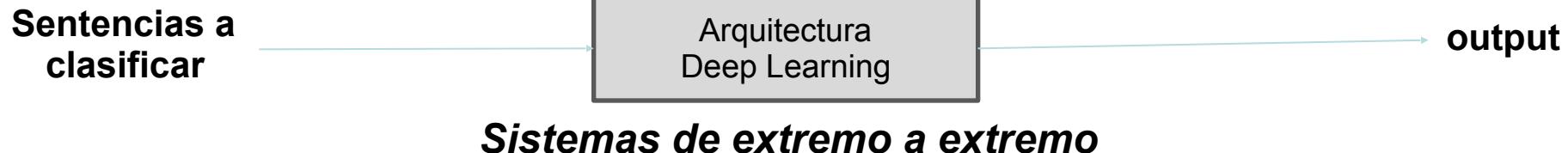
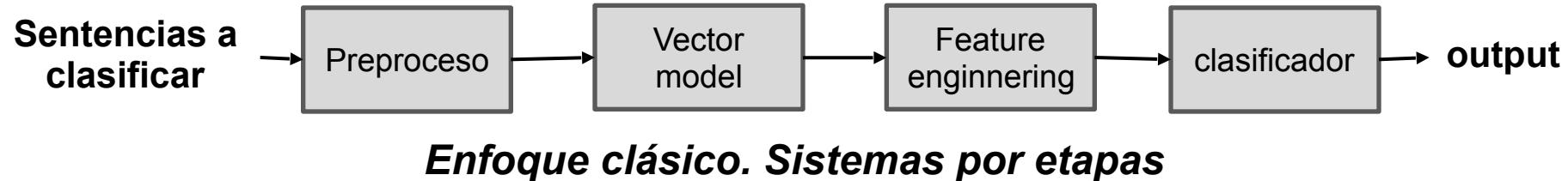
- Base de datos orientada al almacenamiento, procesamiento y búsqueda de texto basado en Apache Lucene.
- Proporciona pipelines de procesamiento predefinidos y personalizables para tratar los textos que se incorporan a la base de datos (tokenizadores, stemmers, filtros,...)
- Ofrece funciones de búsqueda muy rápidas y escalables. Autocompletado, búsqueda flexible, búsqueda por relevancia, etc
- Cada vez más usado para almacenar datos estructurados



elasticsearch

Pipelines vs End to end

Dado un problema determinado (ej:clasificación) podemos afrontar el problema mediante sistemas End to End o sistemas por etapas





Pipelines vs End to end

- Con el desarrollo de las técnicas y arquitecturas Deep Learning, se están consolidando para muchos problemas de text analytics las soluciones de extremo a extremo.
- En lugar de realizar procesos de diseñados manualmente para preprocesar los datos, se utiliza la capacidad de las redes profundas para extraer automáticamente nuevas features. De esta forma, las arquitecturas profundas, combinan niveles orientados a obtener nuevas features con niveles que utilizan esos features maps para resolver el problema, en un mismo flujo de **backpropagation**
- ,Debemos olvidarnos de lo que sabemos?. La aplicación del conocimiento del dominio de aplicación, es siempre determinante. Para muchos problemas, los métodos tradicionales son suficientes
- Como conclusión de este primer bloque, combinaremos las técnicas de pre-proceso de texto, con modelos que ya conocemos para aplicarlos a clasificación supervisada



Topic Modeling

Un tipo de modelaje estadístico que utiliza aprendizaje automático no supervisado para descubrir temas abstractos en un texto o conjunto de textos

Esta técnica de text mining utiliza las estructuras y relaciones semánticas del texto para identificar grupos (clusters) de documentos sin datos de entrenamiento y sin tags o definiciones predefinidas

Dado un conjunto de documentos, Topic Modeling los analiza para identificar temas comunes y en base a estos proporcionar una agrupación en clusters



Topic Modeling

Las técnicas de topic modelling están basadas en la siguiente asunción básica

- Cada documento consiste en una combinación de tópicos
- Cada tema consiste en una colección de palabras.

Los modelos de tópicos se construyen sobre la idea de que la semántica del corpus está gobernada por **ciertas variables “latentes”** u **“ocultas”** que no pueden ser observadas. El objetivo es descubrir estas variables latentes, que son al final los tópicos que nos interesan

Dos de las técnicas más populares para Topic Modeling son:

- LSA: Latent Semantic Analysis
- LDA: Latent Dirichlet Allocation



Topic Modeling. LSA

La matriz de entrada (Matriz de documentos-términos) se descompone en el producto de tres matrices mediante SVD, que factoriza la matriz de entrada A en tres matrices U, Δ, V

$$\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \Delta_{[r \times r]} (\mathbf{V}_{[m \times r]})^T$$

A es la matriz de documentos-términos

Δ es la matriz diagonal de los valores singulares de A

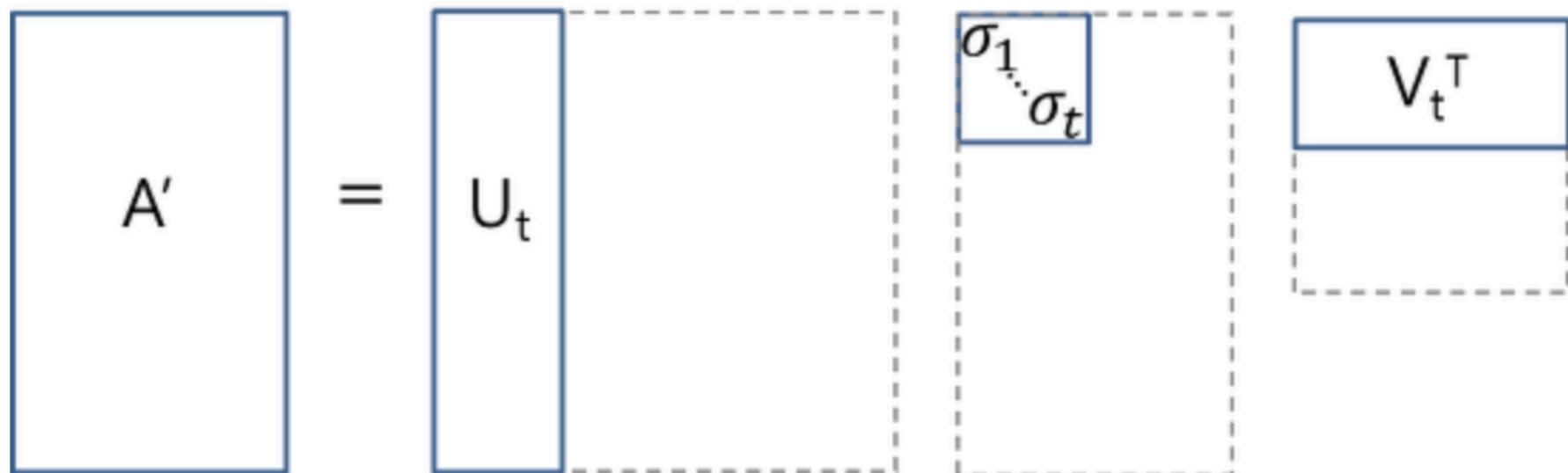
U es la matriz de documentos-topics

V es la matriz de términos-tópicos

Los vectores columna en U, V corresponden a los tópicos

Topic Modeling. LSA

En la práctica, definimos un hiperparámetro t que se corresponde con el número de tópicos que queremos identificar.





Topic Modeling. LSA

Interpretación

Los vectores columna de V nos indican la “importancia” de cada término (palabra) en el tópico. Esto nos permite darle semántica a los tópicos.

Los vectores columna de U nos indican la “importancia” de cada tópico en los documentos.

Las implementaciones (TruncatedSVD en sklearn) nos facilitan la interpretación y visualización de los tópicos.

(ejemplo demostración)

Topic Modeling. LDA

LDA asume que los documentos son generados mediante un proceso estadístico generativo (distribución de Dirichlet) de forma que cada documento es una mezcla de tópicos y cada tópico es una mezcla de palabras

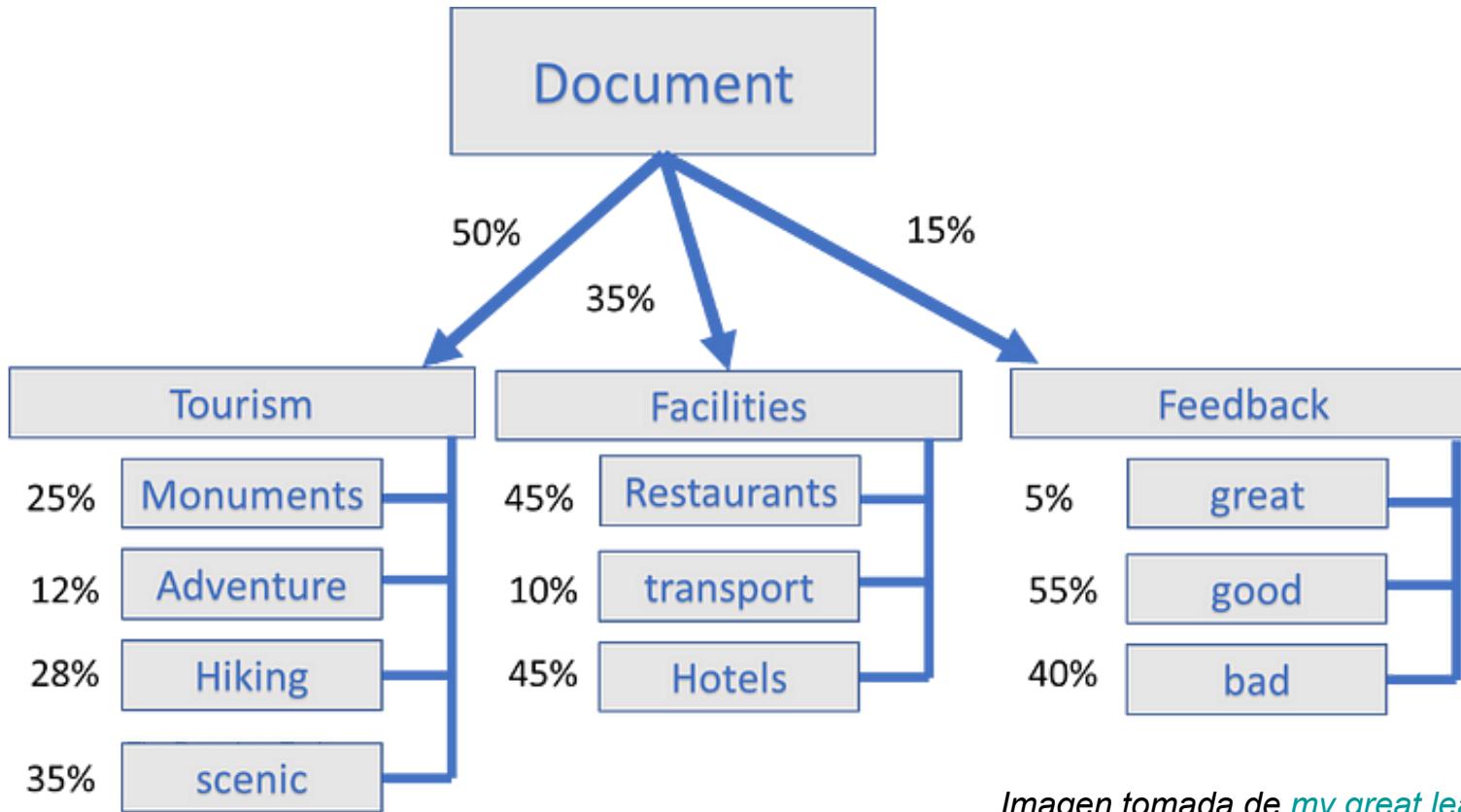


Imagen tomada de [my great learning](#)

Topic Modeling. LDA

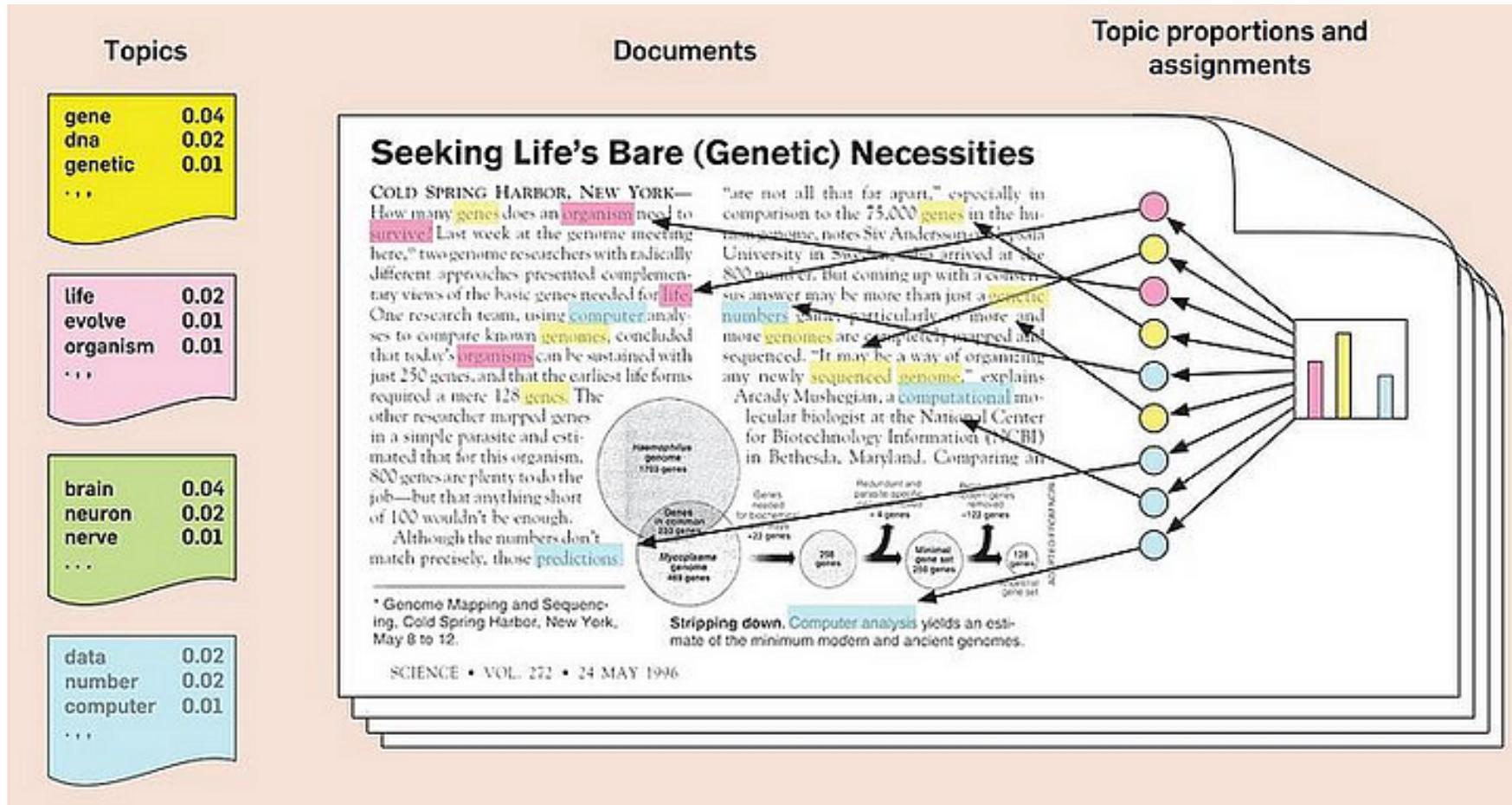


Imagen tomada de [Kim et al.](#)



Clasificación

Ejercicio práctico 1.

En la carpeta de la sesión tenéis un notebook con ejemplos de preproceso, Word vector y clasificación (spam), muy básico. Proponemos mejorar los resultados de la clasificación:

- Intensificando el preproceso (stemming, stop_word removing, etc)
- Ensayando otros métodos de vectorización (TF/IDF, HashingVectorizer) y ajustando el tamaño del vector resultante
- Utilizando otros modelos de clasificación supervisada (en el ejemplo NB) como RandomForest, SVM, logística regression,...



Topic Modeling

Ejercicio práctico 2.

Explore el dataset Reuters mediante Topic Modeling con LSA