

Procesamiento del lenguaje natural



Índice

- Introducción
- N-Gramas
- Recuperación de información
- Clasificación y agrupamiento de documentos

INTRODUCCIÓN

Procesamiento del lenguaje natural

- Área de intersección entre la **Inteligencia Artificial** y la **Lingüística** que estudia la comunicación entre personas y máquinas por medio del lenguaje natural.
 - Sistemas capaces de interpretar o generar documentos
- Algunas de las tareas investigadas son
 - Procesamiento de texto y del habla
 - OCR, speech recognition, text to speech, ...
 - Análisis morfológico, sintáctico, semántico, pragmático
 - Extracción de entidades y sus relaciones
 - Generación de resúmenes
 - Sistemas de diálogo
 - Corrección gramatical
 - Traducción automática
 - Generación de textos

Etapas históricas

- Podemos distinguir 3 grandes etapas históricas
 - PLN simbólico (1950s – 1990s): **Técnicas lingüísticas formales**
 - Se basan en el desarrollo de reglas estructurales que se aplican en las distintas fases del análisis.
 - Encaje de patrones, gramáticas, sistemas de reglas, ontologías, etc.
 - PLN estadístico (1990s-2010s): **Técnicas empiricistas o probabilísticas**
 - Se basan en el estudio de una serie de características de tipo probabilístico asociadas a las distintas fases del análisis del lenguaje.
 - Estas características son extraídas de un conjunto de textos de referencia (corpus)
 - Aprendizaje supervisado, semi-supervisado y no supervisado
 - PLN y redes neuronales (2010s-presente): **Deep Learning**
 - Arquitecturas específicas para trabajar con lenguaje natural
 - Requieren grandes corpus de documentos y poder computacional
 - Grandes avances en los últimos años en áreas como la traducción automática o los sistemas conversacionales

Procesamiento del lenguaje natural

- El problema de los **métodos basados en técnicas lingüísticas formales** es la dificultad de codificar manualmente todo el conocimiento lingüístico necesario (diccionarios, gramáticas, etc.)
 - Esto obliga a trabajar con un lenguaje reducido
 - Este tipo de métodos no suelen contemplar la capacidad de aprendizaje
- En cambio, los **métodos basados en técnicas probabilísticas** aprenden a partir de datos prácticos (corpus de documentos)
- Los sistemas que utilizan métodos necesitan una **fase de entrenamiento** en la que se les debe proporcionar un número suficiente de ejemplos
 - Corpus anotado
- El uso de **redes neuronales profundas** permite encontrar patrones estadísticos complejos en los corpus de documentos

Procesamiento del lenguaje natural

- La existencia de grandes corpus de documentos y el abaratamiento de la capacidad de cómputo a propiciado el auge del aprendizaje automático
- Importantes avances en las dos últimas décadas en áreas como la clasificación de textos, agrupación de documentos, análisis de sentimiento, traducción automática, análisis gramatical...
- Sin embargo, los modelos de lenguaje basados en deep learning son poco interpretables
 - Difíciles de corregir cuando no se comportan como esperamos
 - Abstracciones poco útiles para otras áreas de conocimiento
- Es posible que en el futuro tendamos hacia **aproximaciones mixtas** donde ambas aproximaciones se combinen de alguna forma (por ej. usar técnicas subsimbólicas para generar explicaciones simbólicas)

N-GRAMAS

Modelos de lenguaje probabilísticos

- Un **modelo probabilístico del lenguaje** define una distribución de probabilidad sobre el conjunto de elementos a partir de los valores observados en un **corpus de documentos**
 - Según cual sea el objeto de análisis los elementos pueden ser fonemas, letras, sílabas, o palabras
 - Las frecuencias de aparición de cada uno de los elementos son las que se observen en el corpus
- Son realmente útiles hoy día en multitud de tareas de PLN
 - Texto predictivo: $P(\text{Qué tal } \mathbf{estás}) > P(\text{Qué tal } \mathbf{has comido})$
 - Traducción automática: $P(\text{Voy de visita a su } \mathbf{casa}) > P(\text{Voy de visita a su } \mathbf{hogar})$
 - Corrección ortográfica: $P(\text{Tenemos } \mathbf{calor}) > P(\text{Tenemos } \mathbf{color})$
 - Reconocimiento del habla: $P(\text{Se hizo daño } \mathbf{a sí mismo}) > P(\text{Se hizo daño } \mathbf{así mismo})$

Modelos de lenguaje probabilísticos

- Vamos a considerar que nuestros elementos son palabras, pero lo que veamos aplica igual para fonemas, sílabas, etc.
- Un **modelo probabilístico del lenguaje** permite entre otras cosas
 - Calcular la probabilidad de encontrar una frase o secuencia de palabras determinada
 - $P(\text{Yo, quiero, comer, macarrones, con, tomate})$
 - Calcular la probabilidad de la siguiente palabra
 - $P(\text{tomate} \mid \text{Yo, quiero, comer, macarrones, con})$
- Como el texto es secuencial, podemos pensar que la probabilidad de una palabra depende de todas las anteriores
 - Para ello necesitamos refrescar ciertas nociones de probabilidad

Teoría de probabilidad aplicada a PLN

- Probabilidad condicionada: La probabilidad de B habiendo observado A
 - $P(B|A) = P(A, B) / P(A)$
- La probabilidad de que una palabra sea (por ejemplo) 'perro', sabiendo que la primera palabra es 'el', es la fracción de veces que 'el' aparece seguido de 'perro' en nuestro corpus
 - $P(\text{perro} | \text{el}) = \text{numVeces}(\text{el}, \text{perro}) / \text{numVeces}(\text{el})$
- La probabilidad condicionada se reescribe como $P(A, B) = P(A) P(B | A)$
 - Si añadimos elementos: $P(A, B, C, D) = P(A)P(B|A) P(C|A, B)P(D|A, B, C)$
 - La probabilidad de cada palabra depende de todas las anteriores
- Esto se generaliza mediante la regla de la cadena

$$P(w_1, \dots, w_n) = P(w_1) P(w_2 | w_1) P(w_3 | w_1, w_2) \dots P(w_n | w_1, \dots, w_{n-1})$$

Relajando la hipótesis de la regla de la cadena

- Según la regla de la cadena, la probabilidad de una palabra depende de todas las anteriores
 - Sin embargo, esta hipótesis es impracticable porque no hay un corpus tan grande para asignar probabilidad a las posibles combinaciones de palabras
 - Siempre habrá alguna combinación que no esté presente en el corpus
 - Se puede relajar la hipótesis de la regla de la cadena, haciendo así factibles los cálculos y obteniendo excelentes resultados
- En lugar de considerar que la probabilidad de un elemento depende de todos los anteriores, supone que solamente los $n-1$ elementos anteriores tienen efecto sobre las probabilidades del siguiente elemento i -ésimo.
 - Hipótesis de Markov: $P(w_i | w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-(n-1)}, \dots, w_{i-1})$
- Esta hipótesis no tiene en cuenta que en lenguaje puede haber dependencias de “larga distancia”, sin embargo, funciona muy bien para algunas tareas

Modelos probabilístico de los n-gramas

- El modelo n-grama es uno de los modelos estadísticos del lenguaje más simples pero más útiles.
 - Se puede aplicar a fonemas, letras, sílabas, palabras... aquí nos centraremos en palabras
- El modelo n-grama utiliza la hipótesis de Markov para indicar que la dependencia es con los n-1 anteriores. Por ejemplo:
 - Modelo bigrama ($n = 2$): $P(w_n | w_{n-1})$
 - Modelo trigramas ($n = 3$): $P(w_n | w_{n-1}, w_{n-2})$

Ejemplo de estimación de bigramas

- Supongamos este corpus
 - <s> Yo quiero patatas </s>
 - <s> Yo quiero patatas con carne </s>
 - <s> No quiero carne </s>
 - <s> Quiero carne con patatas </s>
 - <s> Hoy quiero carne </s>
 - <s> Quiero dormir </s>

Estimación óptima por
máxima verosimilitud

$$P(w_n | w_{n-1}) = \frac{\text{frec}(w_{n-1}, w_n)}{\text{frec}(w_{n-1})}$$

<s> indica “inicio de oración”

</s> indica “final de oración”

- Estos son las probabilidades de algunos bigramas del corpus
 - $P(\text{yo} | \text{<s>}) = \text{frec}(\text{<s>}, \text{yo}) / \text{frec}(\text{<s>}) = 2/6 = 0,333$
 - $P(\text{no} | \text{<s>}) = \text{frec}(\text{<s>}, \text{no}) / \text{frec}(\text{<s>}) = 1/6 = 0,167$
 - $P(\text{quiero} | \text{yo}) = \text{frec}(\text{yo}, \text{quiero}) / \text{frec}(\text{yo}) = 2/2 = 1$
 - $P(\text{patatas} | \text{quiero}) = \text{frec}(\text{quiero}, \text{patatas}) / \text{frec}(\text{quiero}) = 2/6 = 0,333$
 - $P(\text{carne} | \text{quiero}) = \text{frec}(\text{quiero}, \text{carne}) / \text{frec}(\text{quiero}) = 3/6 = 0,5$
 - $P(\text{dormir} | \text{quiero}) = \text{frec}(\text{quiero}, \text{carne}) / \text{frec}(\text{quiero}) = 1/6 = 0,167$
- Según este modelo, la continuación más segura de <s>Yo quiero...
 - **$P(\text{carne} | \text{quiero}) > P(\text{patatas} | \text{quiero}) > P(\text{dormir} | \text{quiero})$**
 - La respuesta cambia si consideramos trigramas □ $P(\text{patatas} | \text{yo}, \text{quiero}) = 1$

Estimación de la probabilidad de una frase

- Probabilidades de bigramas obtenidos de un corpus supuesto
 - $P(\text{yo} \mid \langle s \rangle) = 0,25$ $P(\text{quiero} \mid \langle s \rangle) = 0,75$
 - $P(\text{quiero} \mid \text{yo}) = 0,5$ $P(\text{tengo} \mid \text{yo}) = 0,2$ $P(\text{soy} \mid \text{yo}) = 0,3$
 - $P(\text{ser} \mid \text{quiero}) = 0,8$ $P(\text{tomar} \mid \text{quiero}) = 0,2$
 - $P(\text{café} \mid \text{tomar}) = 0,6$ $P(\text{leche} \mid \text{tomar}) = 0,3$ $P(\text{distancia} \mid \text{tomar}) = 0,1$
 - $P(\text{artista} \mid \text{ser}) = 0,9$ $P(\text{informático} \mid \text{ser}) = 0,1$
- $P(\text{Yo quiero tomar café}) =$
 - $P(\text{yo} \mid \langle s \rangle) P(\text{quiero} \mid \text{yo}) P(\text{tomar} \mid \text{quiero}) P(\text{café} \mid \text{tomar}) = 0,25 * 0,5 * 0,2 * 0,6 = 0,015$
 - La frase entera puede no estar en el corpus. Lo normal es que no esté.
 - ¡Si un bigrama no está la probabilidad de la frase es cero!
- La frase que se genera siguiendo la opción más probable es: “*Quiero ser artista*”
 - $P(\text{quiero} \mid \langle s \rangle) P(\text{ser} \mid \text{quiero}) P(\text{artista} \mid \text{ser}) = 0,75 * 0,8 * 0,9 = 0,54$

Estimación de probabilidad en casos raros

- Como hemos visto cualquier n-grama que no esté en el corpus recibe probabilidad 0, el alisado de Laplace alivia los problemas de estimación de probabilidades en casos raros
- El **alisado de Laplace** calcula cualquier probabilidad condicional considerando que ha habido unas observaciones adicionales virtuales de todos y cada uno de los n-gramas posibles
- Siendo AB un bigrama (observado o no), el valor de la probabilidad condicional $P(B|A)$ alisado según Laplace

$$P(B|A) = \frac{\text{frec}(A, B) + t}{\text{frec}(A) + t * m}$$

Donde

- t es el número de observaciones virtuales adicionales
- m es el número de monogramas (palabras) existentes en el corpus
- Las probabilidades totales siguen sumando 1

Alisado por interpolación lineal

- El **alisado por interpolación lineal** es ligeramente más sofisticado
- La interpolación lineal usa la probabilidad incondicional $P(w_2)$ calculada a partir de los datos para hacer que la probabilidad condicional $P(w_2|w_1)$ se parezca a ella, de la siguiente forma

$$P_{\text{Int}}(w_2|w_1) = \alpha P(w_2|w_1) + (1 - \alpha)P(w_2)$$

donde $\alpha \in [0,1]$ regula el peso que se le da a la probabilidad condicional y la probabilidad no condicionada

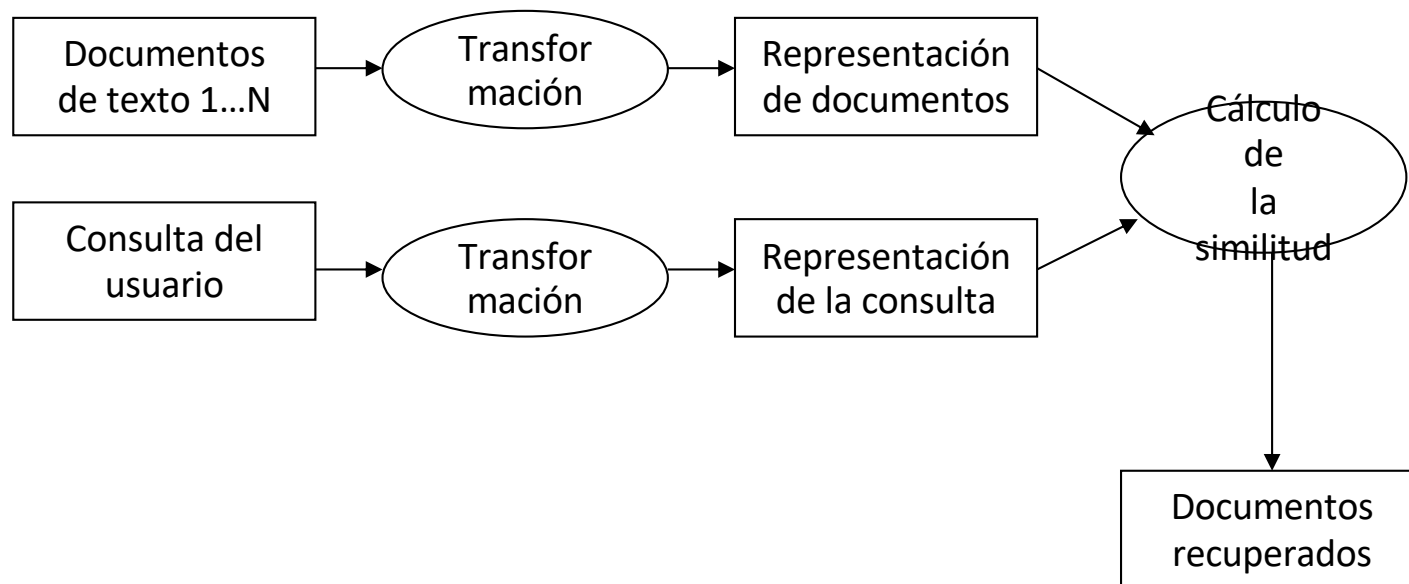
- El valor de α se puede fijar empíricamente con el fin de ajustar el rendimiento
 - También se puede hacer dependiente del “contexto”
 - Si existen muchos bigramas con la palabra w_1 entonces es mejor un valor alto
 - Si no existen muchos bigramas con la palabra w_1 es mejor un valor bajo

RECUPERACIÓN DE INFORMACIÓN

Recuperación de información

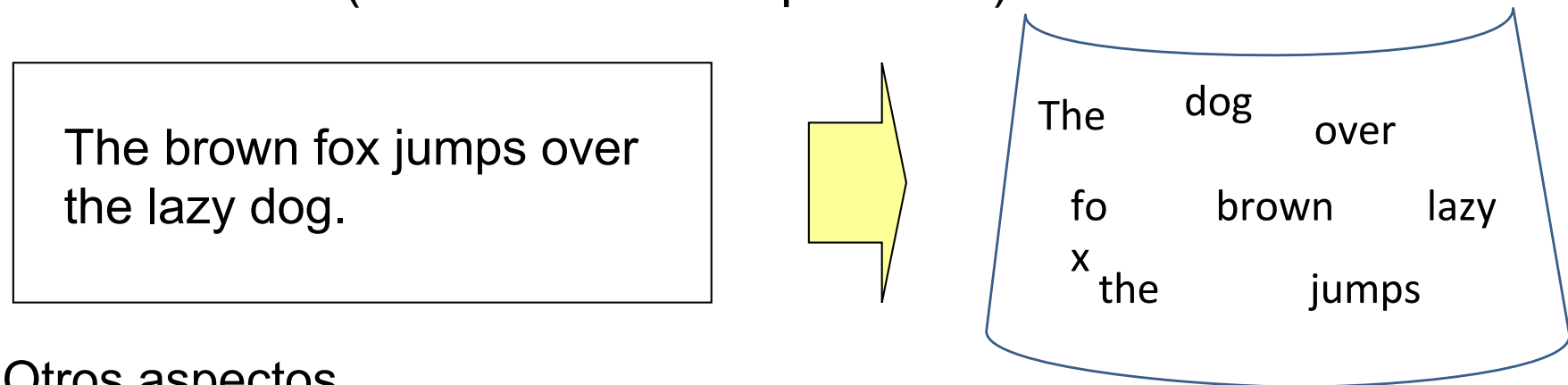
- La **recuperación de información** (*information retrieval*) consiste en encontrar los documentos relevantes asociados con una consulta
 - Es la tarea que hacen los motores de búsqueda que usamos comúnmente
- Aspectos a definir
 - Marco para modelar representaciones de documentos y consultas
 - Función de similitud entre la representación de un documento y una consulta
 - Genera un número real
 - Define un orden de los documentos con respecto a una consulta

Modelo de IR



Cómo pasar de texto a datos: bolsa de palabras (bag of words)

- Se representa un documento como el conjunto de palabras que contiene, independientemente de su orden o su categoría sintáctica
- WordTokenizer (delimitadores de palabras)



❑ Otros aspectos

❑ Normalización

❑ Minúsculas

❑ Extractores de raíces (**stemmer**)

❑ Eliminación de palabras vacías (**stop words**)

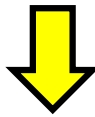
❑ Artículos, preposiciones, adverbios...

Generación de las variables o atributos

1. Enumerar todas las palabras en todos los documentos
2. Eliminar duplicados y ordenarlas
3. Convertir cada palabra en un valor
4. Crear un vector cuyo valor *i*ésimo corresponde al término *i*ésimo

documento *i*

The brown fox jumps over the lazy dog.



abacus ...	brown	...	dog	...	fox	jump	lazy	over	zucchini
↓	↓		↓		↓	↓	↓	↓	↓
(0, 0, ..., 0,	1,	, 0, ..., 0,	1, 0, ..., 0,	1, 0, ..., 0,	1,	0, ..., 0,	1, 0, ..., 0,	1, 0, ..., 0,	1, 0, ..., 0)

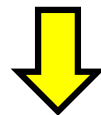
Son vectores dispersos donde la mayoría de los valores es 0

Valores en el vector de palabras

- El valor de un término en el vector de palabras puede
 - Ser binario para indicar la presencia o ausencia del término
 - Ser la frecuencia de aparición del término en el documento
 - Ser la frecuencia con TF/IDF según los documentos del corpus
 - Esto hace que la importancia de una palabra sea inversamente relativa a lo común que es en el documento
 - Lo veremos a continuación

documento i

The brown fox jumps over the lazy dog.



abacus ...	brown	...	dog	...	fox	jump	lazy	over	zucchini
↓	↓		↓		↓	↓	↓	↓	↓
(0, 0, ..., 0,	1,	, 0, ..., 0,	1, 0, ..., 0,	1, 0, ..., 0,	1,	0, ..., 0,	1, 0, ..., 0,	1, 0, ..., 0,	1, 0, ..., 0,

Modelos de IR

• Representación de documento y consulta

- Conjunto de palabras clave => términos índice
 - Vocabulario $T = \{t_1, \dots, t_M\}$, $|T| = M$
 - El índice a menudo se construye agregando las bolsas de palabras de los documentos, pero podría construirse usando un diccionario o combinando ambas aproximaciones
- Importancia de cada término índice => peso (weight)
 - Cuantifica la importancia para describir el contenido semántico del documento
 - Peso del término i en el documento j => $F(t_i, d_j) = w_{ij}$
 - $D = \{d_1, \dots, d_N\}$, $|D| = N$
 - Usar la frecuencia de aparición tiene problemas, porque puede haber palabras que son muy frecuentes en todos los documentos y por tanto son poco relevantes → Usaremos TF-IDF
 - Los pesos de todos los términos de un documento son independientes entre sí
 - Es una simplificación, porque realmente muchos términos estarán correlacionados
- Peso del término i en la consulta => $F(t_i, c) = w_i$

Peso asociado a cada término en el documento

TF-IDF *Term frequency – Inverse document frequency*

Asigna un peso a cada término t en un documento d

$$w_{t,d} = \underbrace{tf_{t,d}}_{\text{Frecuencia del término } t \text{ en el documento } d} \times \underbrace{\log\left(\frac{N}{df_t + 1}\right)}_{\text{Inversa de la frecuencia de aparición del término en los documentos (cuanto más frecuente, menos peso } w_{t,d} \text{).}}$$

Frecuencia del término t en el documento d

Inversa de la frecuencia de aparición del término en los documentos

(cuanto más frecuente, menos peso $w_{t,d}$).

N es el número total de documentos

df_t es el número de documentos del corpus

donde aparece el término t y se le suma 1

para evitar dividir por cero si no existe en ninguno


En el documento d , un término tendrá más peso si es muy frecuente en dicho documento y no aparece casi en el resto

Esto hace que términos que aparecen en todos los documentos no tengan mucho peso.

Documentos como vectores

- Cada documento se representa como un vector de valores $tf \times idf$, un componente por cada término
- Por tanto tenemos un espacio vectorial
 - Los términos son las dimensiones
 - Los documentos son puntos (vectores) en este espacio
- Una colección de documentos se puede representar como una matriz término-documento
 - M términos como columnas
 - N documentos como filas

Vectores TF-IDF

-  Cada documento es un vector de dimensión M y los pesos del vector son los pesos $w_{t,d}$

- Ejemplo: Obras de Shakespeare

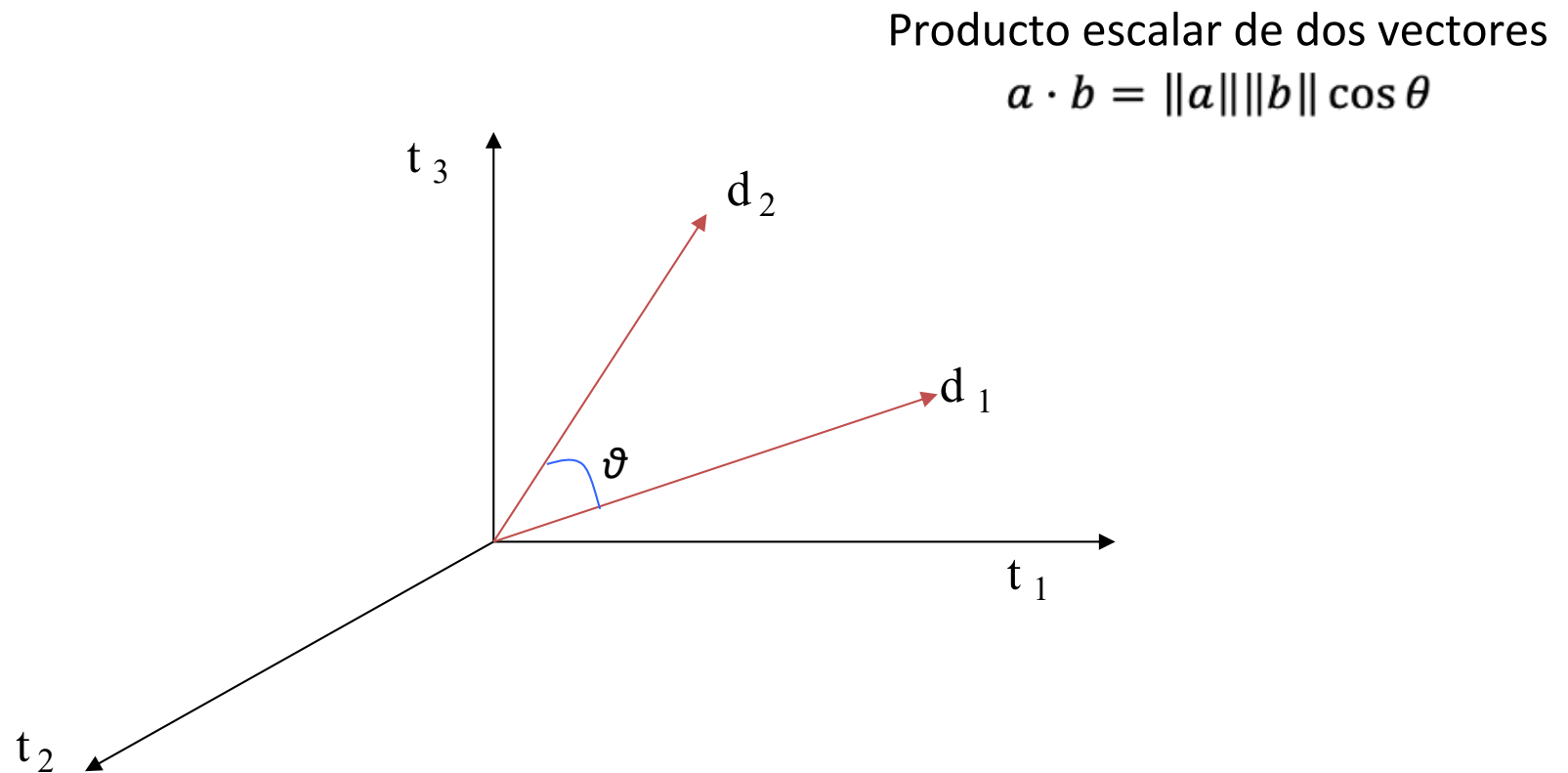
vector de términos

documentos

	Antony	Brutus	Caesar	Calpurnia	Cleopatra	mercy	worser
Antony and Cleopatra	13.1	3	2.3	0	17.7	0.5	1.2
Julius Caesar	11.4	8.3	2.3	11.2	0	0	0
The tempest	0	0	0	0	0	0.7	0.6
Hamlet	0	1	0.5	0	0	0.9	0.6
Othello	0	0	0.3	0	0	0.9	0.6
Macbeth	0	0	0.3	0	0	0.3	0

Similitud entre vectores: coseno

- La similitud entre los vectores d_1 y d_2 es reflejada por el coseno del ángulo que forman



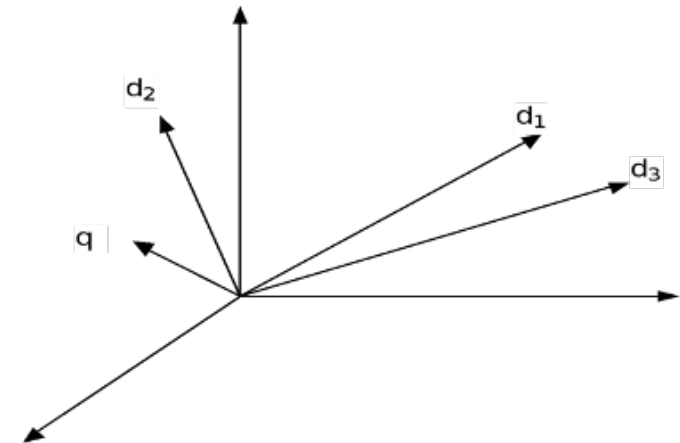
Medida de similitud del coseno

- La **similitud del coseno** se calcula como el coseno del ángulo que forman los dos vectores de términos
 - Si los vectores de términos son iguales, su ángulo es 0, y la similitud es la máxima, es decir, 1
 - El coseno se calcula a partir de la fórmula del producto escalar entre dos vectores
 - $a \cdot b = \|a\| \|b\| \cos \theta$
 - El producto escalar es la suma del producto de los vectores componente a componente

$$\text{coseno}(d_j, d_k) = \frac{d_j \cdot d_k}{|d_j| |d_k|} = \frac{\sum_{i=1}^M w_{i,j} \cdot w_{i,k}}{\sqrt{\sum_{i=1}^M w_{i,j}^2} \sqrt{\sum_{i=1}^M w_{i,k}^2}}$$

IR con la similitud del coseno

Dada una consulta q y un conjunto de documentos d_i , el buscador devuelve la lista de documentos ordenada de mayor a menor según la similitud del coseno.



	algorithm	architecture	computer	logic	program
Documento1	2.23	5.34	2.45	0.00	0.00
Documento2	3.50	0.00	0.00	3.20	1.51
Documento3	0.00	4.76	3.23	0.00	2.31
Consulta	0.00	0.00	0.00	1.06	0.74

similitudes entre documentos d_i y consulta q :

$$\text{sim}(d_1, q) = 0.0000$$

$$\text{sim}(d_2, q) = 0.7009$$

$$\text{sim}(d_3, q) = 0.2133$$

Midiendo el rendimiento de un sistema de IR

- Si tenemos etiquetados los documentos relevantes para una consulta podemos medir la precisión y la exhaustividad

- ¿Todos los documentos que recupera son relevantes? → Precisión
- ¿Recupera todos los documentos relevantes? → Exhaustividad

- Estas métricas son las mismas que se usan para clasificación

- Exhaustividad (recall) o Tasa de Verdaderos Positivos: $TVP = \frac{VP}{VP+FN}$

- Precisión o Valor Predictivo Positivo: $VPP = \frac{VP}{VP+FP}$

- Medida F1: $F1 = 2 \cdot \frac{VPP \cdot TVP}{VPP+TVP} = \frac{2 \cdot VP}{2 \cdot VP + FP + FN}$
 - Es la media armónica de Precisión y Exhaustividad

		VD	
		Documento relevante	
		1	0
Documento recuperado	1	VP	FP
	0	FN	VN

- En los buscadores típicamente se evalúa la calidad de un ranking, es decir, que los documentos más relevantes se devuelvan en las primeras posiciones

- Para ello se mide la precisión en cada posición de la lista, esto equivale a relacionar la precisión y la exhaustividad
 - Porque cada posición de la lista equivale un % de exhaustividad
 - Si la lista tiene 100 elementos, cada elemento supone un 1% de la exhaustividad
- Con esto se puede pintar una curva y como resumen de ella se puede calcular la media de la precisión para todo el rango de exhaustividad (es decir, el área bajo la curva o la integral)

Ventajas e inconvenientes de la bolsa de palabras

Al representar texto mediante vectores de palabras estamos dejando de trabajar con texto propiamente dicho, esto es un arma de doble filo.

- **Ventajas**
 - Simplificación del problema que en muchos casos funciona
 - Permite efectuar consultas de manera sencilla
 - Permite trabajar textos con técnicas estadísticas y de aprendizaje automático
- **Inconvenientes**
 - No maneja la ambigüedad y la variabilidad léxica
 - Considera diferentes dos palabras sinónimas (casa y hogar)
 - Reconoce como iguales palabras polisémicas (ratón)
 - Los vectores resultantes son dispersos (muchos ceros) lo cual es un problema para muchas técnicas de análisis de datos
 - Requiere trabajar con corpus específicos si trabajamos en dominios especializados (con léxico propio), p.ej. medicina

Existen soluciones para muchos de estos problemas pero no las veremos aquí.

WORD-EMBEDDINGS

Word embeddings

- Las word-embeddings son representaciones de palabras en espacios de varias dimensiones que permiten superar algunas de las limitaciones de las representaciones de palabras que hemos visto, por ejemplo:
 - Reflejando similitudes semánticas entre palabras
 - Plasmando relaciones complejas entre palabras
- En este caso cada palabra del diccionario es representada mediante un vector de varias dimensiones (decenas o centenas) que permiten capturar su relación con otras palabras
- Existen distintas técnicas para estimar word-embeddings a partir de un corpus de documentos y fijando el número de dimensiones que queremos usar
 - El resultado es una matriz de tantas columnas como palabras tenga el diccionario y tantas filas como dimensiones hayamos determinado

Representando un diccionario de palabras

La manera naïve de representar las palabras de un diccionario o corpus es un vector binario (one-hot encoding)

- Tantas dimensiones como palabras haya en el diccionario

$V = [a, aaron, \dots, zulu, <UNK>]$ ~10.000 palabras

- Todos valores a cero, menos el que representa la palabra que toma el valor 1
 - Es una representación dispersa

Man	Woman	King	Queen	Apple	Orange
(5391)	(9853)	(4914)	(7157)	(456)	(6257)

$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$
-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------

El número indica la posición de la palabra en el vector binario que representa todas las palabras del diccionario

¡No hay nada en esta representación que modele las relaciones existentes entre las palabras!

La similitud del coseno es cero entre todos ellos

Ejemplo tomado DeepLearning.ai

Representando palabras mediante un vector de características

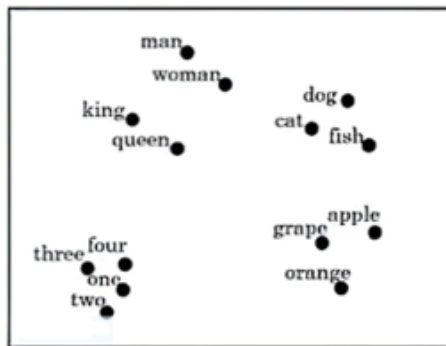
- En lugar de la representación binaria, es mejor usar un conjunto de dimensiones que representen “conceptos” sobre los que situar las palabras del diccionario

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royalness	0.01	0.02	0.93	0.95	-0.01	0.00
Edad	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.05	0.01	0.02	0.01	0.95	0.97
...

- En esta representación, el número de dimensiones es mucho menor que el número de palabras y los vectores de palabras resultantes son densos (o no tan dispersos) porque toma valores diferentes de cero para muchas de sus dimensiones.
 - Aunque para otras tomará valores muy cercanos a cero

Palabras en un espacio n-dimensional

- Si mostramos las palabras en un espacio **denso** donde las dimensiones tienen un **significado**, la posición de las palabras en dicho espacio nos indicará mucho sobre ellas. Por ejemplo
 - Las palabras “*red*”, “*blue*”, “*yellow*”, etc deberían estar relativamente cercanas, ya que suelen jugar un papel parecido en las frases
 - La palabra “*tomatoe*” debería estar más cerca de la palabra “*red*” que, por ejemplo, de “*blue*” ya que están asociadas con más frecuencia
 - Sería posible inferir que “*man*” es a “*woman*”, lo que “*king*” a “*queen*” si el espacio n-dimensional recoge las relaciones de género entre palabras
- Como ese espacio tiene muchas dimensiones, no se puede visualizar tal cual
 - Se podría mostrar usando técnicas de “reducción de la dimensionalidad” como análisis de componentes principales o, más sofisticadas, como t-SNE
 - Pero en la representación 2D resultante se habrán perdido matices importantes



Ejemplo tomado DeepLearning.ai

Analogías sobre word-embeddings

“Man” es a “Woman” como “King” es a...

e_i es el vector columna i-esimo de la matriz word-embedding

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royalness	0.01	0.02	0.93	0.95	-0.01	0.00
Edad	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.05	0.01	0.02	0.01	0.95	0.97
...

“Man” es a “Woman” $\sim e_{5391} - e_{9853} \approx [-2 \ 0 \ 0 \ 0 \ \dots]$

“King” es a “Queen” $\sim e_{4914} - e_{7157} \approx [-2 \ 0 \ 0 \ 0 \ \dots]$

Analogías sobre word-embeddings

“Man” es a “Woman” como “King” es a...

Formulación del problema

$$e_{man} - e_{woman} \approx e_{king} - e_w$$

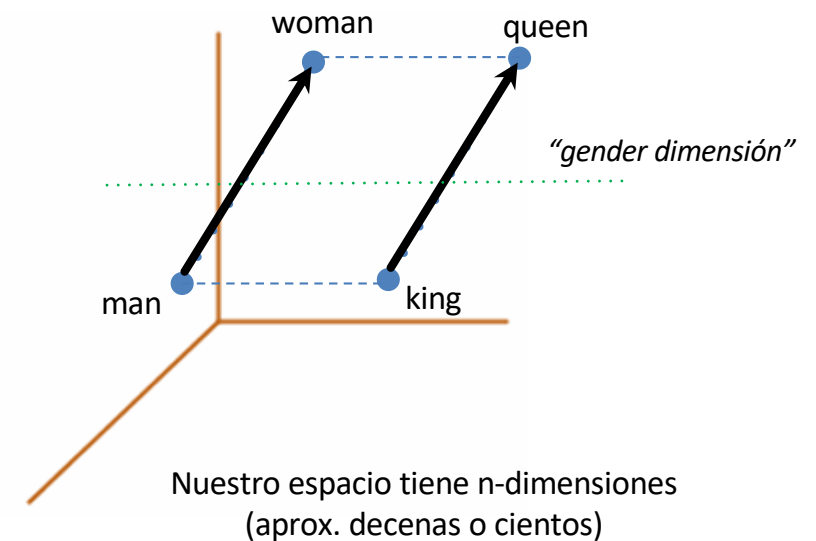
Encontrar la palabra w tal que

$$\arg \max_w (sim(e_w, e_{king} - e_{man} + e_{woman}))$$

Donde $sim(\cdot, \cdot)$ es una medida de semejanza adecuada como la similitud del coseno

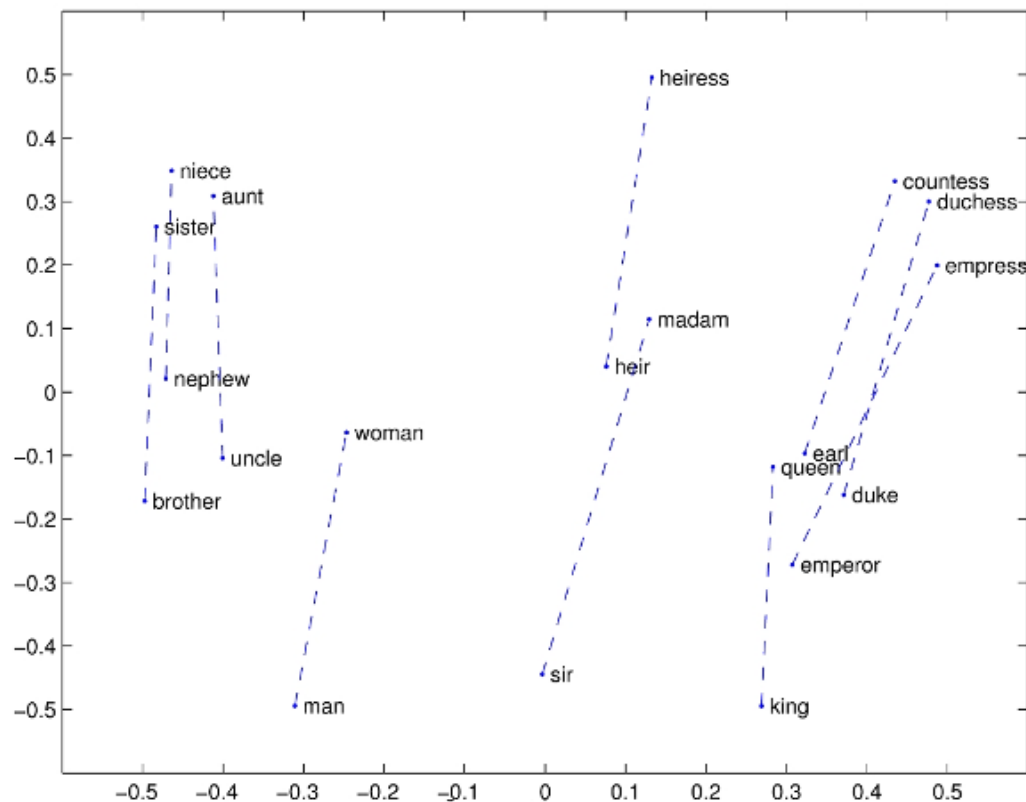
e_i es el vector columna i -ésimo de la matriz word-embedding

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royalness	0.01	0.02	0.93	0.95	-0.01	0.00
Edad	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.05	0.01	0.02	0.01	0.95	0.97
...

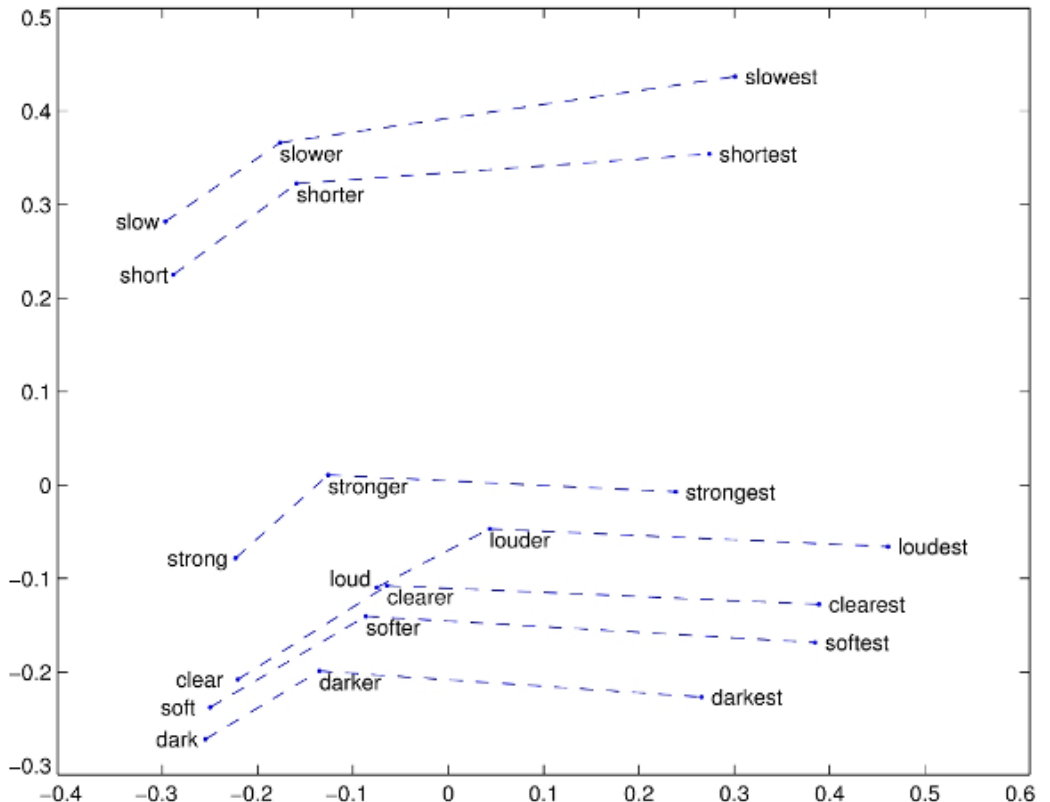


Palabras en un espacio n-dimensional

La diferencia de vectores muestra lo que las diferencia y esta diferencia puede contener conceptos interesantes



Relación de género



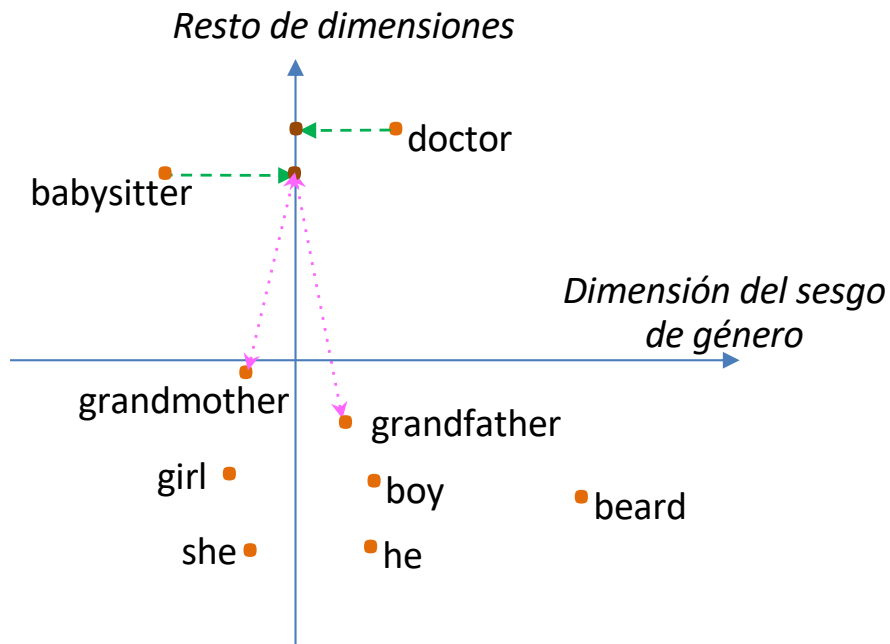
Relación de comparativo y superlativo

Ejemplo tomado de la web de GloVe de U. de Stanford

Sesgos en los word-embeddings

- Como todo sistema que aprenda de datos, los word-embeddings pueden capturar los sesgos de los textos que se usaron para entrenarlos
 - En particular, aprenden asociaciones estereotípicas de género, raza, edad, etc y combinaciones de las mismas.
 - Esto es un problema porque generarán sistemas que reproducirán esos sesgos: reforzándolos y pudiendo generar además resultados incorrectos
- Al usar sistemas de PLN con sesgos podemos encontrarnos con problemas como, por ejemplo, que:
 - En una analogía diga “Hombre es a programador, como mujer a ama de casa”
 - Se asocie la profesión de medicina al género masculino y enfermería al femenino
- Existen mecanismos para eliminar los sesgos

Eliminando sesgos en los word-embeddings



1. Identificar la dirección del sesgo

Promediando la diferencia entre los pares de palabras donde dicho concepto sí representa diferencias oportunas: p. ej. $e_{girl} - e_{boy}$, $e_{he} - e_{she}$, ...

2. Neutralizar el sesgo

En las palabras donde hay sesgo, eliminarlo (ver flechas verde)

3. Ecualizar pares

Reducir sesgos con respecto a la proximidad o lejanía de otras palabras que no deberían reflejar el concepto.

Por ejemplo, “babysitter” debería estar igual de cerca de “grandmother” que de “grandfather” (ver flechas rosas)

Hay que desplazar esas palabras para que estén a igual distancia de dicho concepto.

Cómo se calculan los Word-embeddings

- Los vectores se calculan a partir de un corpus de documentos usando distintas aproximaciones (típicamente redes neuronales profundas)
- Hay que fijar de antemano el número de dimensiones que tendrá el espacio en el que vamos a representar las palabras
 - Sin embargo, el “significado” de esas dimensiones no se puede elegir
- La técnica que calcula los word-embeddings usa el conjunto de datos para fijar el “significado” de las dimensiones y qué valor toman las palabras en cada uno de ellas
 - Aprende cómo distribuir las palabras y cómo configurar las dimensiones para representar bien los ejemplos del conjunto de datos
 - El espacio resultante reflejará **de forma conjunta** relaciones como el color o el género, etc
 - Eso no quiere decir que una dimensión sea explícitamente el género, otra el color, etc
 - De hecho, las dimensiones resultantes no son interpretables, aunque sería posible encontrar “combinaciones” de las dimensiones que representen conceptos inteligibles

Aproximaciones para calcular word-embeddings

- Word2Vec (Mikolov et al.2013)
 - Como datos, usa el modelo **skip-gram** que fija pares de palabras contexto-objetivo
 - Dada una palabra contexto elegida de forma aleatoria, genera varios pares contexto-objetivo eligiendo la palabra objetivo aleatoriamente en un rango de x palabras antes o después de la palabra contexto.
 - Para evitar que palabras vacías (the, my, etc) dominen a las palabras “con significado” se usa alguna heurística de selección
 - Para la estimación usa una aproximación de clasificación soft-max para predecir la palabra “objetivo” aprendiendo en el proceso la matriz con los word-embeddings
 - Utiliza estrategias eficientes para la estimación como softmax jerárquico o muestreo negativo.
- GloVe: Global vectors for word representation (Pennington et al.2013)
 - Como datos, modela el contexto-objetivo calculando el número de veces que la palabra objetivo sucede en la cercanía de la palabra contexto dada una definición de contexto (p.ej. la palabra inmediatamente anterior o en una ventana de 3 palabras antes o después)
 - Usa una aproximación no-supervisada para minimizando una función de error cuadrático encontrar una representación vectorial de las palabras óptima y con propiedades lineales
- Existen otras aproximaciones como [FastText](#) desarrollada por Facebook que además permite realizar clasificación de texto
- Más información:
 - <https://towardsdatascience.com/word-embeddings-exploration-explanation-and-exploitation-with-code-in-python-5dac99d5d795>
 - <https://towardsdatascience.com/word-representation-in-natural-language-processing-part-ii-1aee2094e08a>

Uso de los word-embeddings

- Los word-embeddings han sido son un área que tiene un notable desarrollo “en abierto” y que permite la reutilización al menos con fines no-comerciales
- Es posible **entrenar tus propios word-embeddings** a partir de un corpus de documentos usando software libre, por ejemplo,
 - [Gensim](#) es una librería de PLN que permite entrenar tus propios word-embeddings con word2vec y también tenemos disponible la [implementación de Google](#)
 - [GloVe](#) también tiene el código disponible en la web de la Universidad de Stanford y también en Gensim
- Sin embargo, esto requiere conocimiento y un corpus de documentos muy grandes. Para muchas tareas es mejor usar word-embeddings “pre-entrenados” de gran calidad, como por ejemplo:
 - Los word-embeddings de [word2vec generados por Google](#) y los de [GloVe de Stanford](#) para disitintas configuraciones, y [word2vec sobre Wikipedias en distintas lenguas](#)
- Dado un word-embedding pre-entrenado se puede:
 - Actualizarlo con nuestro corpus de documentos para así incorporar nuevas palabras y relaciones entre ellas
 - Incorporar nuestro “word-embedding” en una red neuronal o bien de manera estática o dinámica (dependiendo de si se permite a la red actualizar sus valores o no durante el entrenamiento)

CLASIFICACIÓN DE DOCUMENTOS

Clasificación de documentos

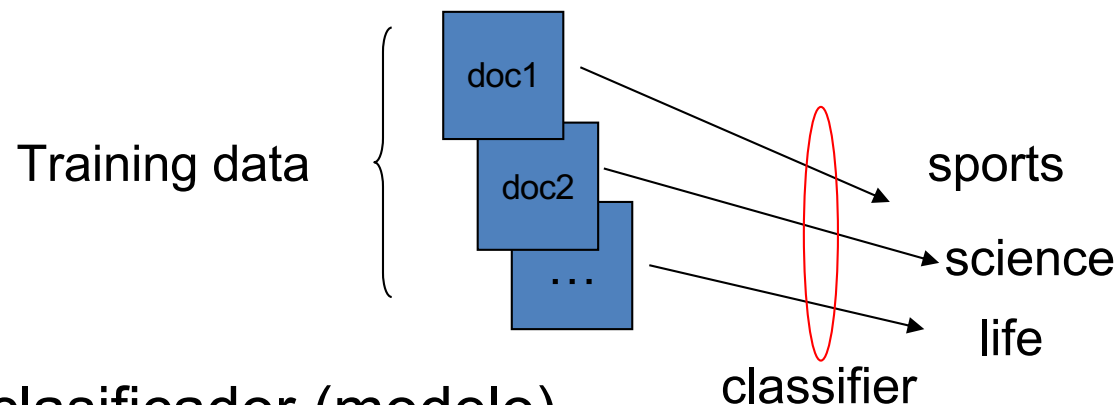
- Categorización de textos (Text categorization)
 - Consiste en clasificar documentos en categorías predefinidas.
 - Clasificar correo en spam o no-spam.
 - Asignar etiquetas temáticas a noticias
 - Enfoque supervisado
 - Aprendizaje de una función que asigne documentos a categorías
 - Requiere de un conjunto de documentos previamente etiquetado (datos de entrenamiento)

Clasificación de documentos con bolsa de palabras

- Para realizar estas tareas se trabaja, como hemos visto en aprendizaje automático, con una matriz de datos
 - Las filas son los documentos
 - Las columnas, es decir las variables son las palabras (o la raíz de las mismas) usando la representación de bolsa-de-palabras
 - En las celdas podemos tener
 - Valores binarios
 - Frecuencia de aparición (normalmente el valor relativo con respecto al número de palabras del documento)
 - TF/IDF

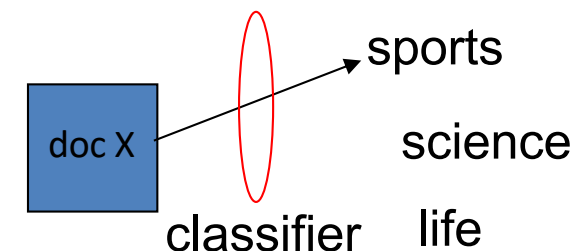
Categorización de texto

1. Construir un conjunto de entrenamiento clasificando cada documento
 - Si pueden pertenecer a varias clases es un problema de clasificación multi-etiqueta (multilabel)



1. Crear un clasificador (modelo)
 - Aplicando un algoritmo de aprendizaje automático a los datos de entrenamiento
 - Redes Neuronales, Support Vector Machine (SVM), Naïve Bayes (NB)

3. Clasificar nuevos documentos con el clasificador



Clasificadores para categorización de texto

- Los problemas de clasificación de texto sufren, a menudo, del problema de la maldición de la dimensionalidad
 - Gran cantidad de variables y, según el caso, incluso más que documentos
- Además, la matriz de datos resultante suele ser una matriz dispersa (con muchos valores a cero)
- Para evitarlo se deben usar métodos de clasificación que no se vean afectados por estas características del problema
- También se usan técnicas de reducción de la dimensionalidad (aprendizaje no supervisado) para reducir las variables a un número de factores
 - Estos factores a la postre constituyen los ejes “temáticos” o “conceptuales” del conjunto de documentos

Clasificación probabilística de documentos

- Para aproximarnos al problema vamos a suponer:
 - una variable de clase y con m categorías
 - el vector x_i de términos con d **variables binarias** que representan la presencia o no de un término i en el documento
 - Luego veremos cómo trabajar con la frecuencia del término
- Si quisiésemos aproximarnos al problema de clasificación utilizando la tabla de la distribución conjunta de las probabilidades, necesitaríamos especificar $2^{d+m}-1$ probabilidades para todas las combinaciones de valores de x e y , es decir, $p(y \wedge x_1 \wedge \dots \wedge x_d)$
 - El teorema de Bayes puede ayudarnos a mitigar este problema

OJO: representaremos el vector de términos x_i en negrita
y cada término k de ese vector sin negrita $x_{k,i}$

Aproximación bayesiana al problema

- Según el teorema de Bayes, dado un vector de términos \mathbf{x}_i para el que queremos determinar su clase y_j , tenemos que calcular la probabilidad condicionada cada clase y_j y optar por la clase más probable

$$P(y = y_j | \mathbf{x} = \mathbf{x}_i) = \frac{P(y = y_j)P(\mathbf{x} = \mathbf{x}_i | y = y_j)}{P(\mathbf{x} = \mathbf{x}_i)}$$

- Las probabilidades de $P(\mathbf{x} = \mathbf{x}_i)$ podemos obtenerlas a partir de los datos observados en los que se ha observado el vector de términos \mathbf{x}_i , aunque veremos que no son estrictamente necesarias
- Las probabilidades a priori $P(y = y_j)$ también se obtienen de los datos
 - Si n_j es el número de documentos de la clase $y = y_j$, $P(y = y_j) = n_j/n$
- Las probabilidades condicionales $P(\mathbf{x} = \mathbf{x}_i | y = y_j)$ a determinar son 2^d y además debemos tener en cuenta la dependencia

$$\begin{aligned} P(x_1 \wedge \dots \wedge x_d | y) &= P(x_1 | y)P(x_2 \wedge \dots \wedge x_d | y \wedge x_1) = P(x_1 | y)P(x_2 | y \wedge x_1)P(x_3 \wedge \dots \wedge x_d | y \wedge x_1 \wedge x_2) = \dots \\ &= P(x_1 | y)P(x_2 | y \wedge x_1) \dots P(x_d | y \wedge x_1 \wedge \dots \wedge x_{d-1}) \end{aligned}$$

- La dependencia de las variables del vector \mathbf{x} , complica el problema

Ignorando la dependencia

- Si ignoramos las posibles dependencias entre las variables del vector x y asumimos que son independientes, es decir, que para todo $a \neq b$, tenemos que $P(x_a|y \wedge x_b) = P(x_a|y)$
 - Esto supone asumir que la co-ocurrencia de términos es totalmente independiente cosa que es radicalmente falsa

- Sin embargo, los cálculos se simplifican enormemente ya que

$$P(x_1 \wedge \dots \wedge x_d|y) = \prod_{k=1}^d P(x_k|y)$$

siendo x_k una variable binaria que indica la presencia del término k

- El clasificador **naïve Bayes** simplifica la realidad asumiendo que todas las variables utilizadas para clasificar son independientes
 - Esta hipótesis será en muchísimos casos incorrecta
 - De ahí que su nombre sea *naïve Bayes* (Bayes ingenuo) o maliciosamente *idiot Bayes* (Bayes idiota)
 - Sin embargo, pese a ello, el naïve bayes obtiene resultados tan buenos o mejores que otras técnicas de clasificación más sofisticadas

El clasificador Naïve Bayes

- Dado el vector binario de términos x_i para el que queremos determinar su clase (o categoría) y_j , tenemos que determinar

$$P(y = y_j | x = x_i) = \frac{P(y = y_j) \prod_{k=1}^d P(x_k = x_{ki} | y = y_j)}{P(x = x_i)}$$

- Para ello, no es necesario estimar $P(x = x_i)$, porque al estar en el denominador es un valor constante para todas las posibles clases y_j
- En ese caso estamos obteniendo una estimación de la verosimilitud de que x_i sea de clase y_j que es proporcional a la probabilidad

$$P(y = y_j | x = x_i) \propto P(y = y_j) \prod_{k=1}^d P(x_k = x_{ki} | y = y_j)$$

siendo $x_{ki} \in \{0,1\}$ un valor binario que indica la presencia o no del término x_k en el documento i

- Con Naive Bayes solo habría que especificar $md + m - 1$ probabilidades
 - Siendo m el número de clases y d el número de términos (variables binarias) del vector que define a los elementos

El clasificador Naïve Bayes

- $$P(y = y_j | \mathbf{x} = \mathbf{x}_i) \propto P(y = y_j) \prod_{k=1}^d P(x_k = x_{ki} | y = y_j)$$

- La probabilidad de observar cada clase y_i es la frecuencia relativa de casos observados en el conjunto de entrenamiento en la clase

$$P(y = y_j) = \text{frec}(y = y_j)$$

- La probabilidad de observar el término x_k en el documento i condicionada que el documento sea de la clase y_i se calcula mediante frecuencias relativas como sigue

$$P(x_k = x_{ki} | y = y_j) = \frac{\text{frec}(x_k = x_{ki} \wedge y = y_j)}{\text{frec}(y = y_j)}$$

- Esta estimación da el óptimo según la estimación por máxima verosimilitud
 - Sin embargo, presenta problemas en muestras pequeñas que hay que corregir con métodos de alisado (p.ej Laplace o interpolación línea)

Ejemplo de clasificación con naïve Bayes

- Supongamos que debemos entrenar un conjunto de 220 emails para elaborar un filtro de SPAM y observamos las siguientes frecuencias

$$P(x_k = x_{ki} | y = y_j) = \frac{\text{frec}(x_k = x_{ki} \wedge y = y_j)}{\text{frec}(y = y_j)}$$

Emails de cada tipo

	SPAM	No-SPAM
	20	200
	SPAM	No-SPAM
millonario	5	5
Viagra	5	4
Nigeria	2	10
estimado	15	120
examen	2	50
ejercicio	4	40
clase	2	25

Prob. cond. de presencia
de término

	SPAM	No-SPAM
millonario	0,25	0,025
Viagra	0,25	0,02
Nigeria	0,1	0,05
estimado	0,75	0,6
examen	0,1	0,25
ejercicio	0,2	0,2
clase	0,1	0,125

Prob. cond. de ausencia
de término

	SPAM	No-SPAM
millonario	0,75	0,975
Viagra	0,75	0,98
Nigeria	0,9	0,95
estimado	0,25	0,4
examen	0,9	0,75
ejercicio	0,8	0,8
clase	0,9	0,875

Basta con calcular una tabla porque la otra es el complementario de la columna

Ejemplo de clasificación con naïve Bayes

$$P(y = y_j | \mathbf{x} = \mathbf{x}_i) \propto P(y = y_j) \prod_{k=1}^d P(x_k = x_{ki} | y = y_j)$$

$$P(x_k = x_{ki} | y = y_j) = \frac{\text{frec}(x_k = x_{ki} \wedge y = y_j)}{\text{frec}(y = y_j)}$$

Emails de cada tipo

SPAM	No-SPAM
20	200

Prob. cond. de presencia
de término

	SPAM	No-SPAM
millonario	0,25	0,025
Viagra	0,25	0,02
Nigeria	0,1	0,05
estimado	0,75	0,6
examen	0,1	0,25
ejercicio	0,2	0,2
clase	0,1	0,125

Prob. cond. de ausencia
de término

	SPAM	No-SPAM
millonario	0,75	0,975
Viagra	0,75	0,98
Nigeria	0,9	0,95
estimado	0,25	0,4
examen	0,9	0,75
ejercicio	0,8	0,8
clase	0,9	0,875

Nuevo email a clasificar

$\mathbf{x}_N = \{\text{millonario, Nigeria, estimado, clase}\}$

$$P(y = \text{SPAM} | \mathbf{x} = \mathbf{x}_N) = \frac{20}{220} (0,25 \cdot 0,75 \cdot 0,1 \cdot 0,75 \cdot 0,9 \cdot 0,8 \cdot 0,1) = \mathbf{0,92 \cdot 10^{-4}}$$

$$P(y = \text{No - SPAM} | \mathbf{x} = \mathbf{x}_N) = \frac{200}{220} (0,025 \cdot 0,98 \cdot 0,05 \cdot 0,6 \cdot 0,75 \cdot 0,8 \cdot 0,125) = 0,501 \cdot 10^{-4}$$

Es más verosímil que el email recibido sea SPAM

Naive Bayes con variables cuantitativas

- Si x_k es una variable continua (por ejemplo, frecuencia o peso de la palabra) en lugar de binaria necesitamos una manera diferente para estimar $P(x_k = x_{ki} | y = y_j)$
- Normalmente se suele asumir que la variable x_k sigue una distribución normal cuya media y varianza dependen de y
 - OJO: este supuesto puede no ser muy acertado cuando hablamos de frecuencias de palabras en un corpus de documentos
- Durante el entrenamiento, para cada combinación de un atributo continuo x_k con un valor de clase $y = y_j$, se estimará su media μ_{kj} y su desviación típica σ_{kj} según los datos observados
- Durante la fase de clasificación se estimará la $P(x_k = x_{ki} | y = y_j)$ de un ejemplo concreto utilizando la función de distribución gaussiana de media μ_{kj} y desviación típica σ_{kj}

$$P(x_k = x_{ki} | y = y_j) = \frac{1}{\sigma_{kj} \sqrt{2\pi}} \exp\left(-\frac{(x_{ki} - \mu_{kj})^2}{2\sigma_{kj}^2}\right)$$

Clasificación con word-embeddings

- Podemos caracterizar un documento usando un word-embedding pre-entrenado
- La forma más directa consiste en representar cada palabra del documento en lugar de con una variable binaria con el vector d-dimensional que representa a dicha palabra en el word-embedding y por un vector de ceros si dicha palabra no aparece en el documento
 - El problema de esta aproximación es que el número de variables de entrada ahora es el número de palabras del diccionario multiplicado por el número de dimensiones
 - ¡El problema de la maldición de la dimensionalidad aumenta!
- Una forma más practicable de incorporar esa información consiste en resumir en un solo vector de d-dimensiones todas las palabras del documento
 - El resumen consiste en **caracterizar el documento como el vector medio de los vectores d-dimensionales** de las palabras que contiene dicho documento
 - También pueden caracterizarse mediante los vectores mínimo y máximo (ambos juntos)
 - El resumen puede multiplicar los vectores de palabras resultantes por su frecuencia o TF/IDF
 - Esta aproximación es también muy tosca y funcionará peor cuanto más sofisticada deba ser la clasificación y más largo sea el documento
- Normalmente, las word-embeddings se incorporan en redes neuronales de aprendizaje profundo que se entrenan con un corpus de documentos específicos para realizar tareas más sofisticadas (*transfer learning*)

Análisis de sentimiento

- Definición
 - Se trata de clasificar mensajes de redes sociales o de valoraciones de productos según tengan sentimientos positivos o negativos.
- Las aproximaciones más comunes se basan en los conceptos que ya sabemos
 - Transformación del texto en un vector de palabras y un filtrado para concentrarnos en palabras catalogadas como de sentimiento positivo (bueno, fenomenal, estupendo...) o negativas (malo, pésimo, fatal...)
 - Un enfoque muy sencillo consistiría en contar las palabras de sentimiento positivo (y negativo) dividir las entre el número de palabras totales para obtener un peso total de cada sentimiento en el texto.
 - Las palabras pueden pesar diferente según su intensidad, modificadores, negaciones, etc.
 - También se pueden usar un enfoque de clasificación haciendo a un clasificador aprender a partir de un corpus de opiniones positivas y negativas.

RECURSOS DE PLN

Más recursos de PLN

- Además de los enlaces ya vistos, hay muchos recursos de acceso abierto disponibles en la red
 - **NLTK**: librería de código abierto, escrita en Python, con herramientas avanzadas de PLN muy usada en investigación y docencia
 - <http://www.nltk.org/>
 - **SpaCy**: librería de código abierto, escrita en Python y Cython, con herramientas avanzadas de PLN para aplicación industrial y profesional
 - <https://spacy.io/>
 - **Lucene**: librería de código abierto para implementar motores de búsqueda (buscadores) escrita en Java (y portada a múltiples lenguajes)
 - <http://lucene.apache.org/>
 - **OpenNLP**: librería de código abierto, escrita en C++, con herramientas comunes de PLN basadas en aprendizaje automático
 - <http://opennlp.apache.org/>
 - **Freeling**: librería de código abierto, escrita en C++, con herramientas avanzadas de PLN, desarrollada por la UPC
 - <http://nlp.lsi.upc.edu/freeling/>