Pre-procesamiento de texto

Curso de ingeniería de características

Julio Waissman

Maestría en Ciencia de Datos

septiembre 2020

Algunas definiciones

- Un documento es una cadena de caracteres
  - Un tweet
  - Un libro
  - Un correo electrónico
- Un *corpus* es una colección de documentos (congruentes entre si)
- Los documentos pueden dividirse en párrafos, frases y/o palabras
  - Pueden estas anotados
  - Pueden estar asignados a una categoría
- Un vocabulario es un conjunto cerrado de palabras de dimensión V

Procesamiento básico de la información

- Normalmente la información en forma de texto en lenguaje natural viene en formatos complicados y trae muchos *artefactos* que influyen de manera muy importante en el resultado de una tarea de PLN.

- Se requiere manejo de información en diferentes formatos (i.e. `json` o `xml`).

- El acondicionamiento del texto es fundamental (capitalización, minúsculas, corrección, eliminación de argot . . .)

Expresiones regulares

1. Las expresiones regulares (*regex*) juegan un rol importante en PLN
2. El primer paso en una tarea de PLN (limpieza) seguido implica el uso de *regex* sofisticadas
3. Son difíciles de hacer y dificil de corregir pero son superpoderosas
4. Vamos a utilizar la primer semana de éste curso para aprender lo básico
5. Si quieres practicar, puedes consultar este tutorial de *regex* en *python*

Algunas expresiones regulares

Disyunciones

Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

Rangos

Pattern	Matches	
`[A-Z]`	An upper case letter	`Drenched Blossoms`
`[a-z]`	A lower case letter	`my beans were impatient`
`[0-9]`	A single digit	`Chapter 1: Down the Rabbit Hole`

Algunas expresiones regulares

Negaciones

Pattern	Matches	
[^A-Z]	Not an upper case letter	Oyfn pripetchik
[^Ss]	Neither 'S' nor 's'	I have no exquisite reason"
[^e^]	Neither e nor ^	Look here
a^b	The pattern a carat b	Look up a^b now

Disyunciones con |

Pattern	Matches
groundhog|woodchuck	
yours|mine	yours mine
a|b|c	= [abc]
[gG]roundhog|[Ww]oodchuck	

Algunas expresiones regulares

* + ? .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n

Inicio y fin de cadena

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	1 "Hello"
\.$	The end.
.$	The end? The end!

Vamos a ver la libreta

Veamos unos ejemplo no tan triviales

¿Que es un token?

- Una secuencia de caracteres con significado
- En español, la puntuación y los caracteres sirven para separar palabras.
- ¿Y los neologismos, los *hashtags*, las direcciones url, las fechas,. . . ?
- ¿La puntuación debe de integrarse en el token?

Token y tokenización

Token

Unidad útil de caracteres para el procesamiento semántico. Puede ser palabras, frases, simbolos, etc.

Tokenización

Proceso de separar un documento en tokens.

1. El método más popular en inglés es el Treebank tokenization
2. En español el proceso de tokenización es relativamente simple, si no se trata con documentos especializados
3. En idiomas como aleman, turco, japones o chino, la tokenización es un problema difícil

Veamos unos ejemplos

Regresamos a la libreta

Normalización de *tokens*

- Algunos métodos basados en reglas para normalizar tokens con mismo significado
  - Minúsculas en todas las palabras
  - Manejo de acrónimos (EUA, E.U.A., US, U.S., USA, U.S.A.)
  - Requiere de desarrollar muchas reglas particulares

- Encontrar el mismo token para diferentes formas con el mismo significado semántico (niños, niño, niña, niñas, . . .)
  - Basado en análisis morfológico (lematización)
  - Basado en reglas heurísticas (*stemming*)

- Palabras de paro (que no aportan valor semántico ¿será?)

Stemming

- Proceso de remover y remplazar sufijos de una palabra en varios pasos a fin de obtener la *raiz* de la palabra (llamada tallo, o *stem*)
- Método heurístico para ir recortando las palabras
- Típicamente falla en formas irregulares
- El método más típico para inglés es el Porter's stemming algorithm
- El método más típico en español es el Snowball spanish stemming algorithm

Lematización

- Se refiere al uso de un análisis lingüístico formal basado en un vocabulario cerrado y en reglas morfológicas.
- Difícil de hacer, requiere de mucho esfuerzo.
- Convierte el token a la forma base de la palabra, tal y como esté definido en un diccionario de referencia (lexema).
- Existen pocos lematizadores en español (freeling, spacy).

Problema de clasificación de documentos

- Una de las tareas más utilizadas con datos de texto es:
  - Análisis de polaridad
  - Determinación de tópicos
  - Detección de *spam*
  - Detección de autores
  - Identificación de idioma

- Se utilizan métodos de clasificación bien conocidos

- Transformación de la información:

$$\text{documento} \rightarrow (x_1, x_2, \ldots, x_{nd}) \rightarrow \mathbb{R}^M$$

donde M es el número de características que se extraen del documento

El método de la bolsa de palabras

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

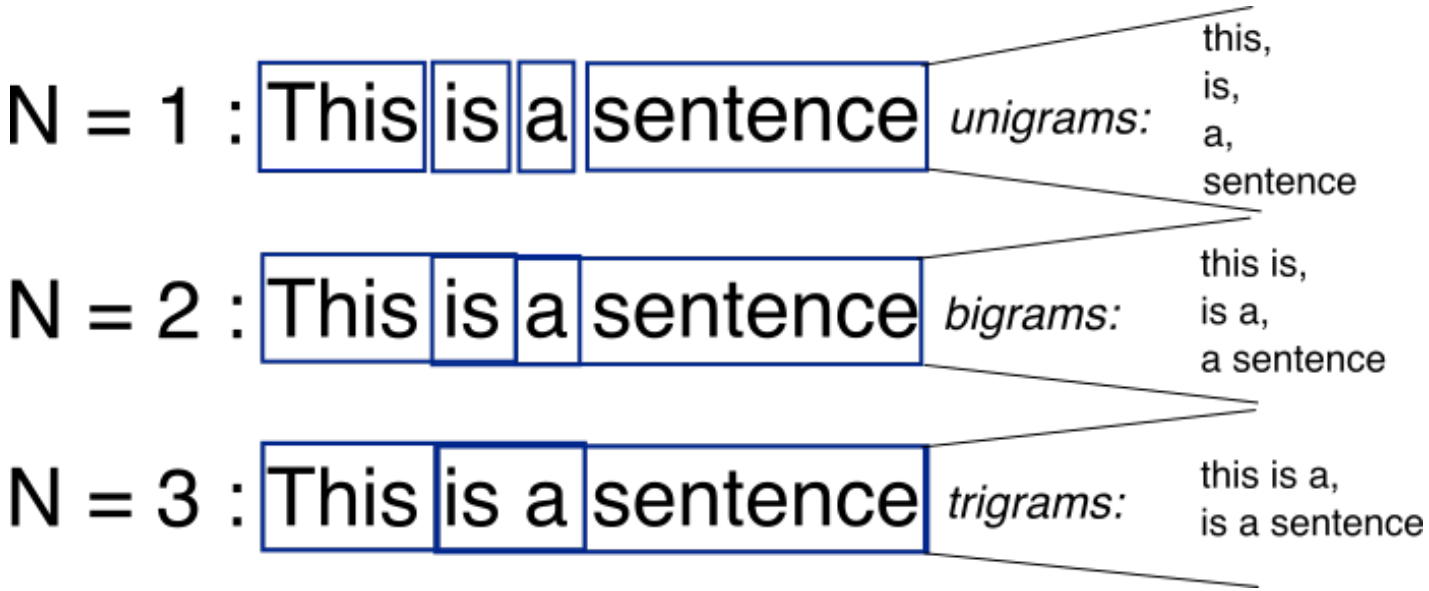
Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List

for
is
of
the
to

Manteniendo algo del orden de las palabras

Se pueden agregar pares de tokens, tripletas, . . .



Explosión de la dimensión de las características

Explosión de características

- Si las características son muchas, entonces se pueden eliminar las que son muy frecuentes
- Tambien se pueden eliminar las que aparecen muy poco
- Si las características que quedan son las de mediana frecuencia, mientras más aparezca el token en el documento más importancia tiene para éste (*frecuencia del término en el documento*)
- Pero mientras la palabra aparezca en más documentos diferentes (*frecuencia de documentos con el término*), menos representativo es el token respecto a un caso específico.

TF-IDF

TF (Term frequency)

$$tf(t, d) = n_{t,d}$$

donde $n_{t,d}$ es la frecuencia que aparece el término t en el documento d

IDF (Inverse Document frequency)

$$idf(t, D) = \log\left(\frac{1 + n_D}{1 + df(D, t)}\right) + 1$$

donde n_D es el número de documentos y $df(D, t)$ es el número de documentos en los que aparece t

TF-IDF

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

y se normalizan los valores para todos los documentos

Revisemos el ejemplo

Similaridad entre palabras

- Dos palabras (tokens) son más similares si comparten más características de significado
- Es una relación entre los sentidos de las palabras
- Es posible calcular la similaridad entre palabras con un thesaurus
- Funciona bien con sustantivos, pero no con adjetivos o verbos
- Es difícil y en muchas ocasiones el contexto es diferente

Se puede conocer una palabra por su compañía

Modelos distribuidos

- Estudiar las palabras por su contexto (en una ventana de tamaño fijo)
- Calcular las coocurrencias en el corpus de las palabras u y v dentro de la misma ventana n_{uv}
- Calcular la inforación mutua palabra a palabra (PMI, por *Pointwise Mutual Information*)

$$PMI = \log \frac{\Pr(u, v)}{\Pr(u)\Pr(v)} = \log \frac{n_{uv} n}{n_u n_v}$$

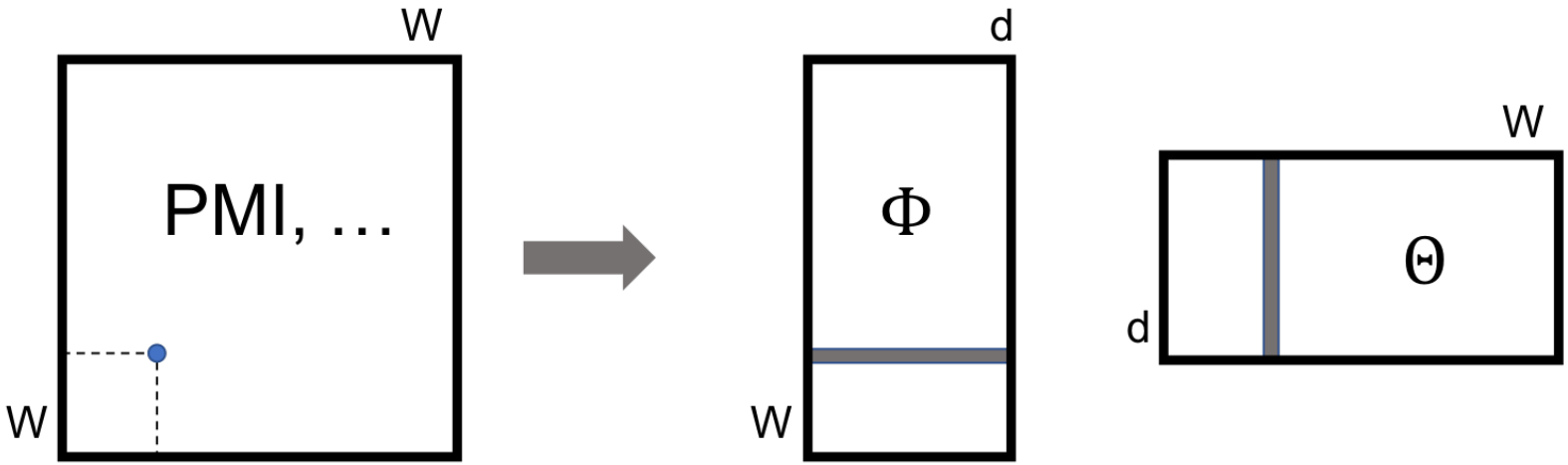
donde n es el número de palabras del corpus, n_u y n_v son el número de veces que aparecen las palabras u y v respectivamente.

- Usar solo la parte positiva (pPMI)

$$pPMI = \text{máx}(0, PMI)$$

Semantica distribuida por métodos lineales (SVD)

- Se basa en una matriz de similaridad entre palabras (PMI, cuentas, ...)
- Reduccion de la dimensionalidad por SVD
- Resultado: Una matriz Φ de dimensión $W \times d$ tal que el renglón i de la matriz es un vector de dimensión d que codifica la palabra w_i del vocabulario



Semantica distribuida por optimización (GloVe)

En lugar de estimar directamente la matriz Φ, se infiere del problema de optimización

$$\underset{\Phi,\Theta,b,b'}{\text{máx}} \sum_{u\in V}\sum_{v\in V} f(n_{uv})(\phi_u^T\theta_v + b_u + b'_v - \log n_{uv})^2$$

donde $f : [0,\infty] \to [0,1]$ es una función de saturación

- La función a optimizar es cuadrática y existen métodos bien desarrollados para solucionarlo.
- No deja de ser computacionalmente costoso (en tiempo y memoria)
- Solamente se basa en la matriz de coocurrencias
- Uno de los primeros métodos desarrollados mas allá de la descomposición matricial en valores singulares.

Métodos tipo word2vec

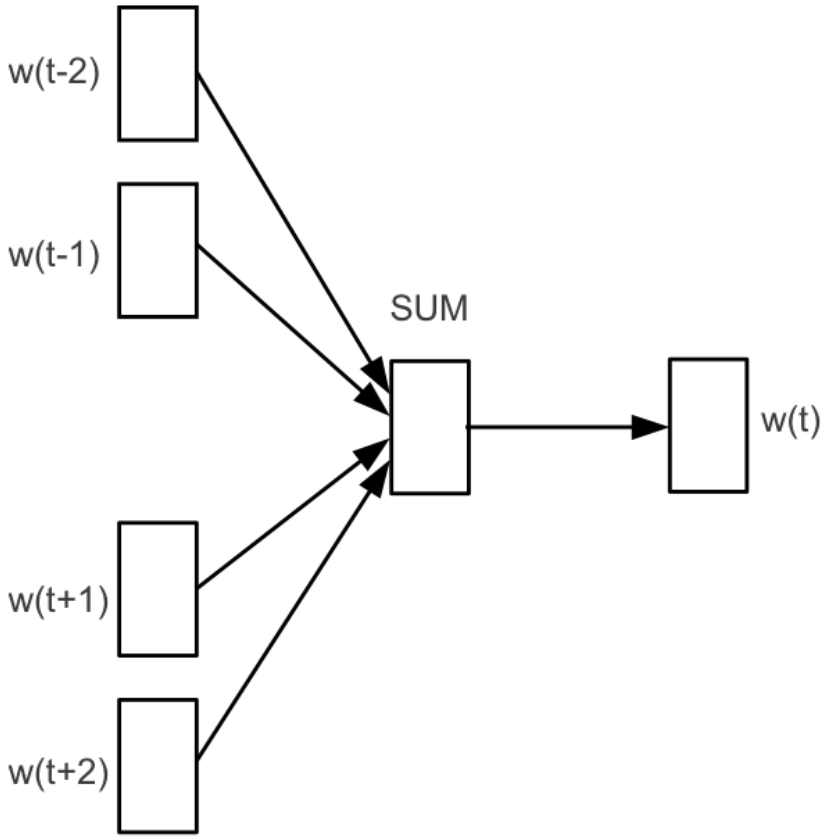
- Basados en la clasificación correcta de palabras en lugar de matrices de similaridad
- Se basa en el contexto sobre una ventana deslizante
- Dos estratégias básicas
  1. *Continous BOW (CBOW)*. Se basa en encontrar la probabilidad de una palabra, dado su contexto

$$\Pr(w_i | w_{i-n}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+n})$$

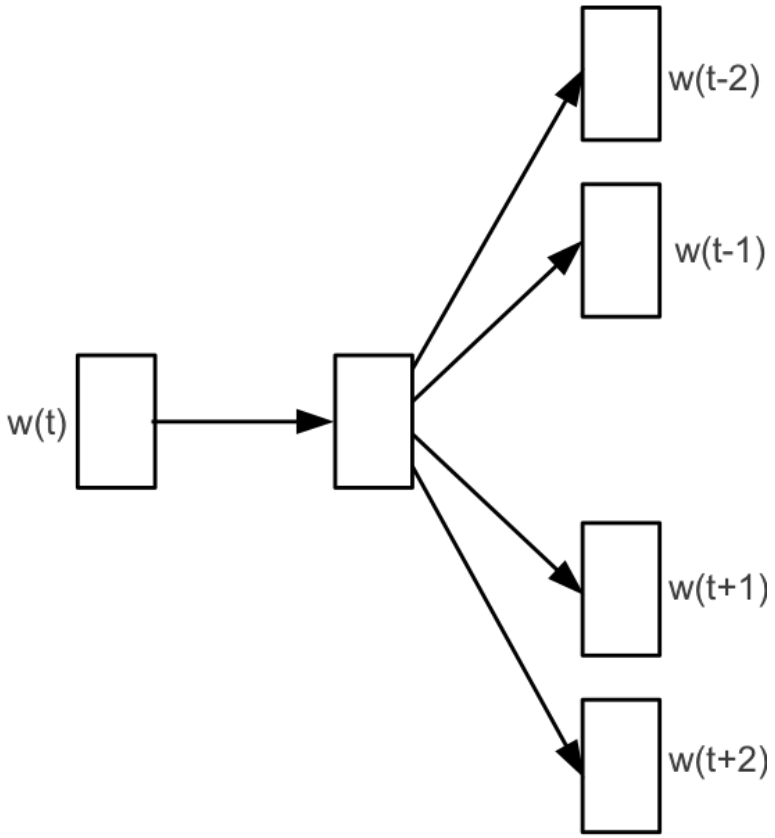
  2. *Skip-gram*. Se trata de encontrar el contexto de una palabra, si conociéramos la palabra

$$\Pr(w_{i-n}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+n} | w_i)$$

CBOW y Skip-gram

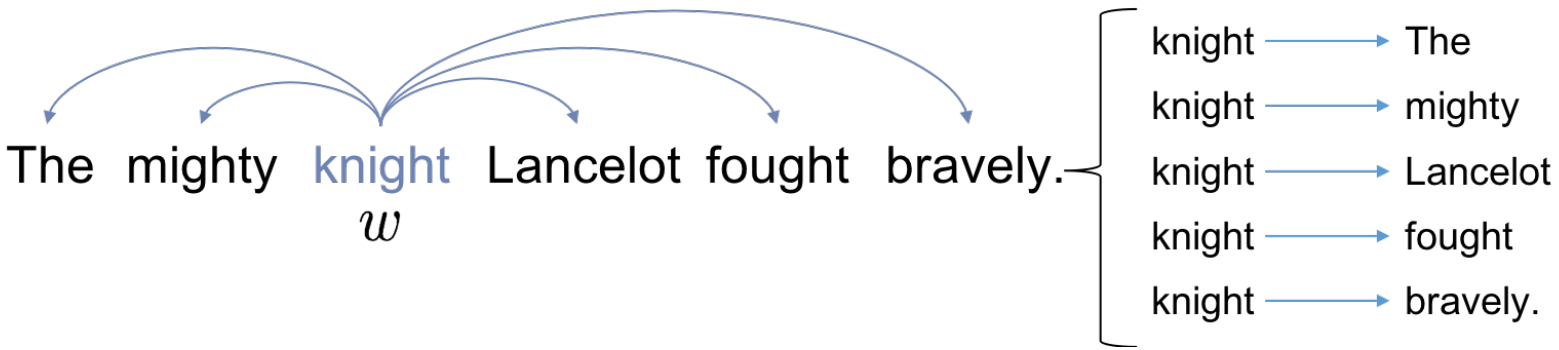


CBOW



Skip-gram

Skip-gram



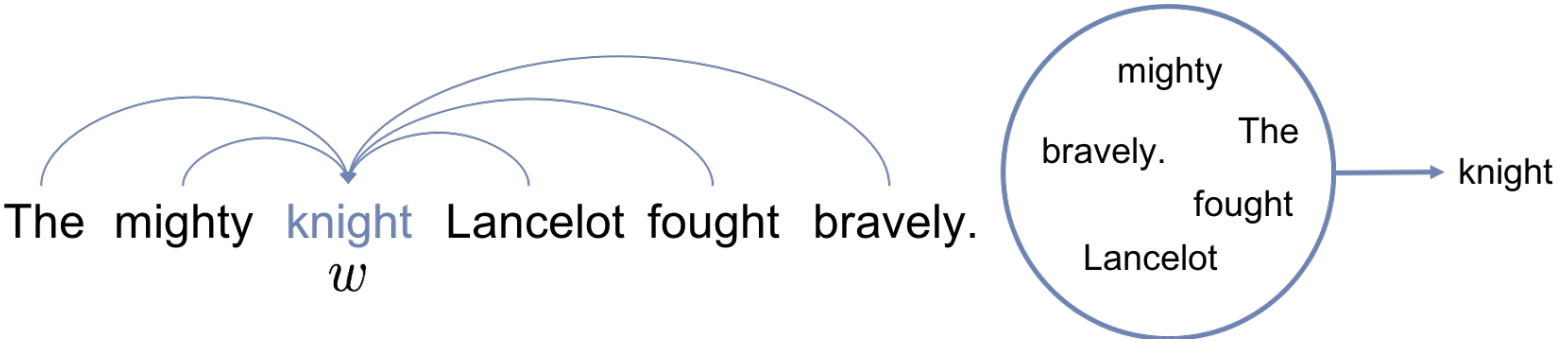
$$\Pr(c|w) = \frac{\exp(\phi_w^T \theta_c)}{\sum_{v \in V} \exp(\phi_w^T \theta_v)}, \quad \phi_w, \theta_v \in \mathbb{R}^d$$

Si tenemos un corpus $(w_1, w_2, \ldots, w_T)$ entonces Φ se obtiene de resolver

$$min_{\Phi,\Theta} - \sum_{t=1}^{T} \sum_{c \in \mathcal{C}_t} \log \frac{\exp(\phi_{w_t}^T \theta_c)}{\sum_{v \in V} \exp(\phi_{w_t}^T \theta_v)}$$

donde $\mathcal{C}_t$ son las palabras de contexto de w_t.

CBOW

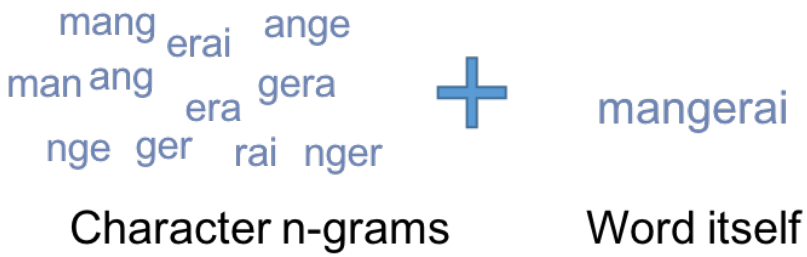


$$\Pr(c|w) = \frac{\exp(h_w^T \theta_w)}{\sum_{v \in V} \exp(h_w^T \theta_c)}, \quad h_w, \theta_v \in \mathbb{R}^d$$

donde

$$h_w = \sum_{c \in \mathcal{C}_w} \phi_c$$

FastText



$$\Pr(c|w) = \frac{\exp(h_w^T \theta_c)}{\sum_{v \in V} \exp(h_w^T \theta_v)}, \quad h_w, \theta_v \in \mathbb{R}^d$$

donde

$$h_w = \sum_{g \in w} x_g, \quad x_g \in \mathbb{R}^d$$

es la suma de n-gramas de caracteres

FastText permite estimar vectores para palabras OOV

Similaridad entre palabras

- Dado un par de palabras, encontrar su similaridad.
- La similaridad de las palabras no depende de la distancia entre sus vectores, si no de el angulo entre vectores.
- Similaridad coseno:

$$s(w, v) = \frac{\phi_w^T \phi_v}{\|\phi_w\|_2 \|\phi_v\|_2}$$

- Funciona bien para lenguajes morfológicamente ricos (como el español)

Analogías

- Sea a y a' dos palabras, tenemos la palabra b y queremos encontrar b' que tenga una relación con b análoga a la que tienen a y a', por ejemplo

  `hombre` es a `rey` lo que `mujer` es a ?

- En forma de vectores de palabras se podría expresar como:

$$\phi_a - \phi_{a'} \approx \phi_b - \phi_{b'},$$

  y por lo tanto,

$$\phi_{b'} \approx \phi_{a'} - \phi_a + \phi_b$$

- Tomando en cuenta la similaridad coseno

$$b' = \arg \max_{x \notin \{a, a', b\}} \cos(b - a + a', x)$$

- Funciona relativamente bien para analogías sintácticas

Codificación de sentencias o textos

- En la práctica, se prueba primero generar un vector por sentencia

Sea $s = (w_1, \ldots, w_T)$ una *sentencia* o *frase*, la cual está compuesta de una serie de palabras o tokens. Entonces, la sentencia se puede codificar como:

$$\phi_s = \frac{1}{T} \sum_{t=1}^{T} \phi_{w_t}$$

- Nada más se toman en cuenta los tokens que existen en el vocabulario
- Suele funcionar extrañamente bien

Veamos unos ejemplos