Vectores de palabras

Ingeniería de Características

Julio Waissman

Maestría en Ciencia de Datos

2022-2

¿Cómo utilizar texto para hacer modelos?

- Transformación de tokens:

$$\text{token} \to \mathbb{R}^M$$

- Transformación de documentos:

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

El método inocente: 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.

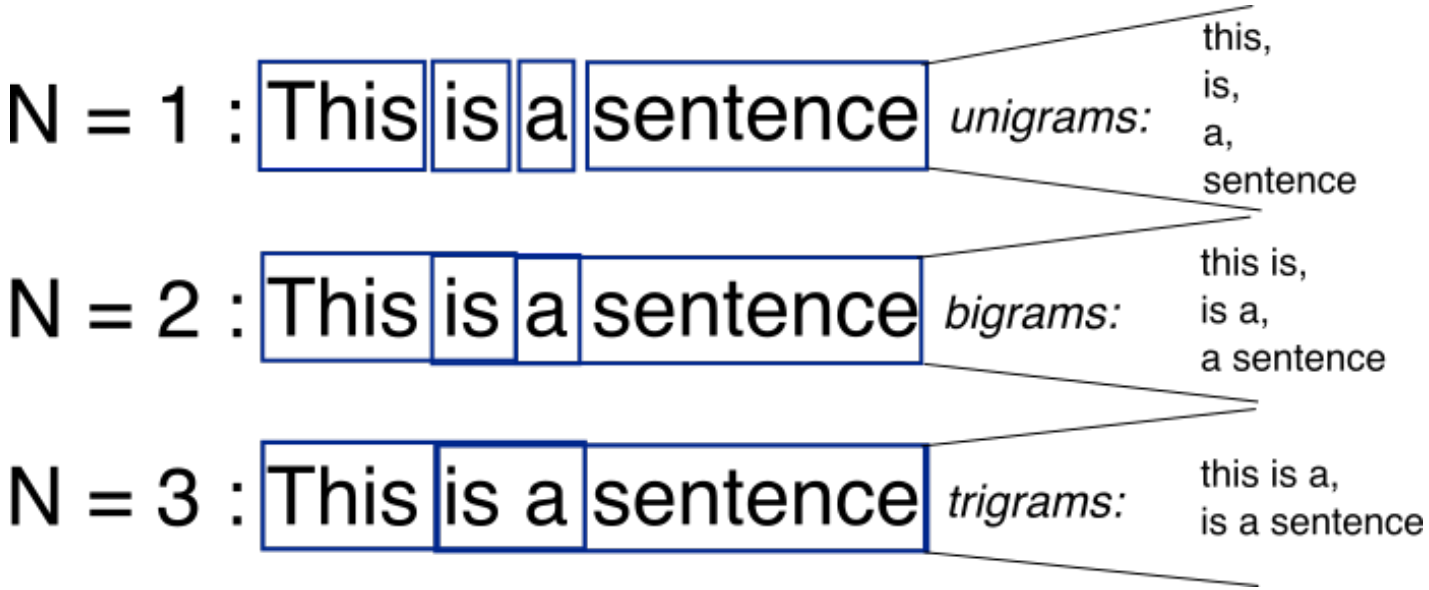
Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

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

El método semántico

- 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

Información mutua entre palabras

- 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 información mutua palabra a palabra (PMI, por *Pointwise Mutual Information*)

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

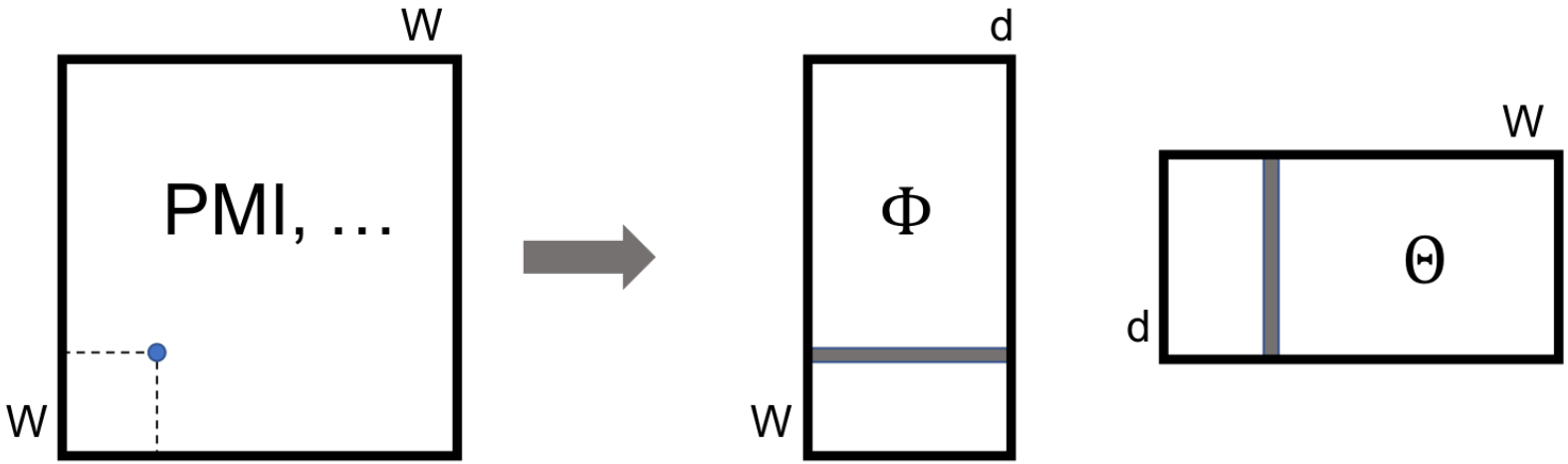
donde n es el número de palabras del corpus, n_u y n_v son el número de veces que aparecen las palabras u y v respectivamente.

- 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
- Reducción 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

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

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

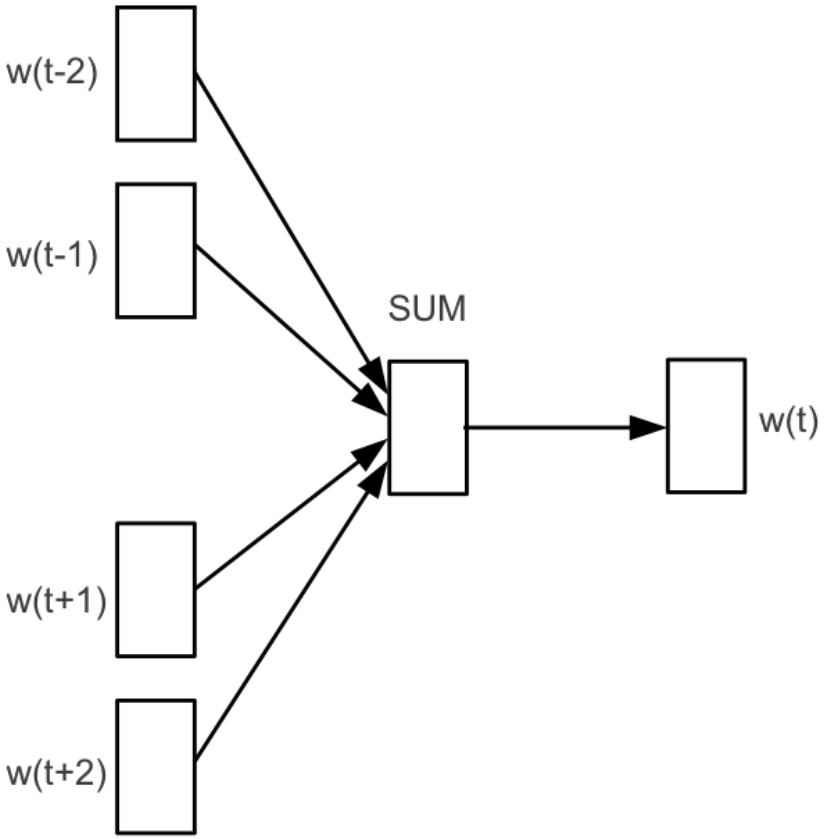
- Basados en la clasificación correcta de palabras en lugar de matrices de similaridad
- Se basa en el contexto sobre una ventana deslizante
- Dos estratégias básicas
  1. *Continous BOW (CBOW)*. Se basa en encontrar la probabilidad de una palabra, dado su contexto

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

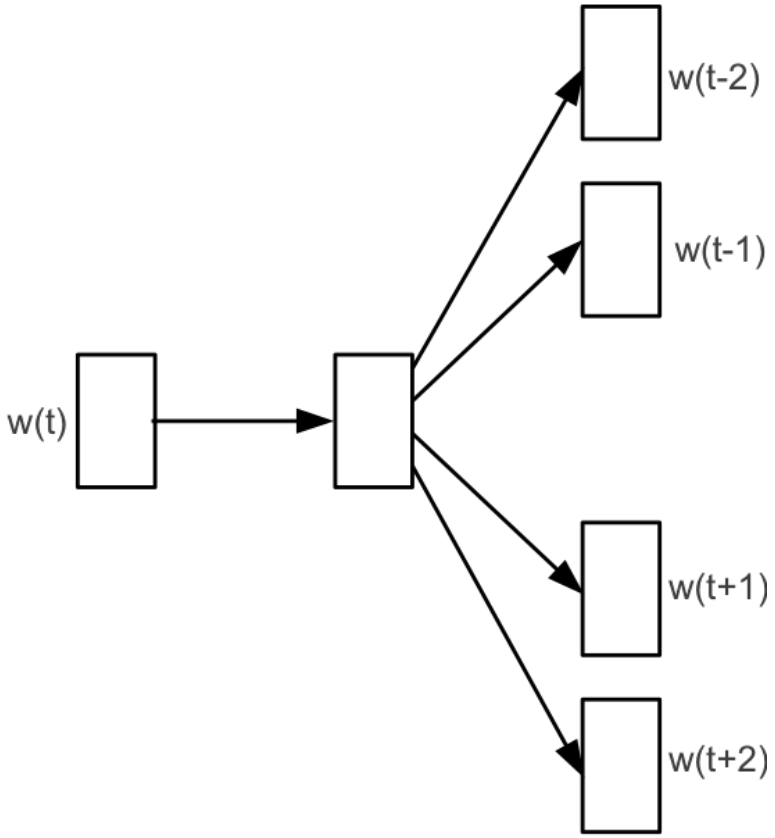
  2. *Skip-gram*. Se trata de encontrar el contexto de una palabra, si conociéramos la palabra

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

CBOW y Skip-gram

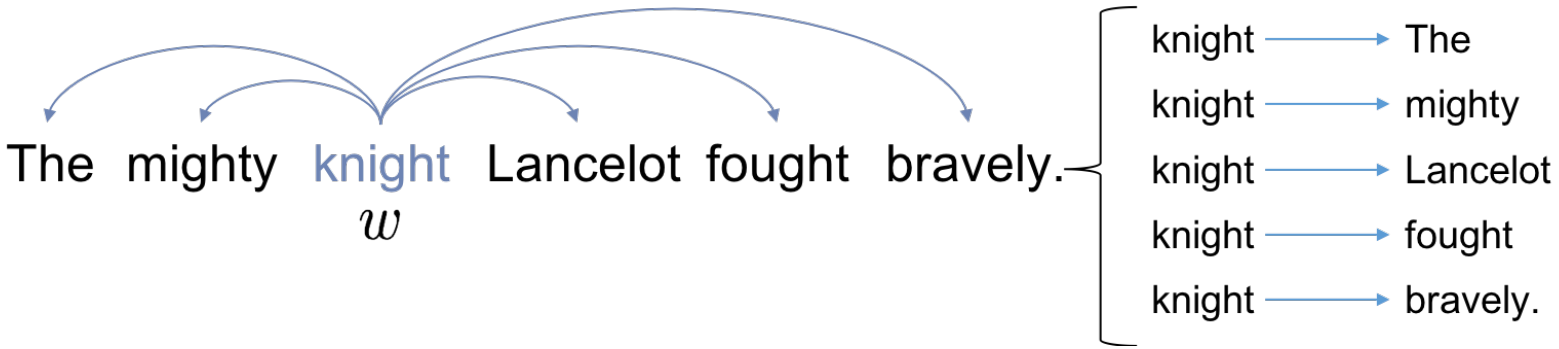


CBOW



Skip-gram

Skip-gram



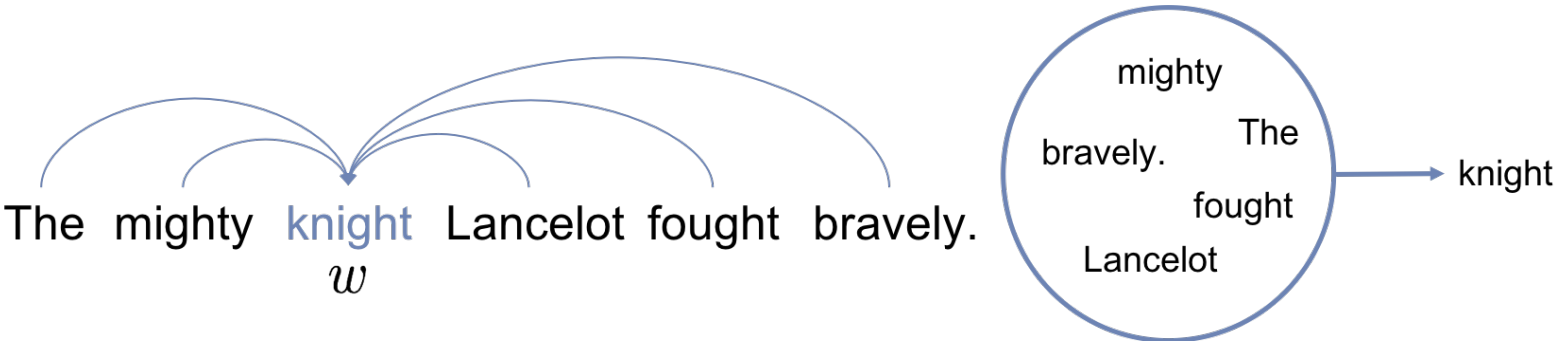
$$\Pr(c|w) = \frac{\exp(\phi_w^T \theta_c)}{\sum_{v \in V} \exp(\phi_w^T \theta_v)}, \quad \phi_w, \theta_v \in \mathbb{R}^d$$

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

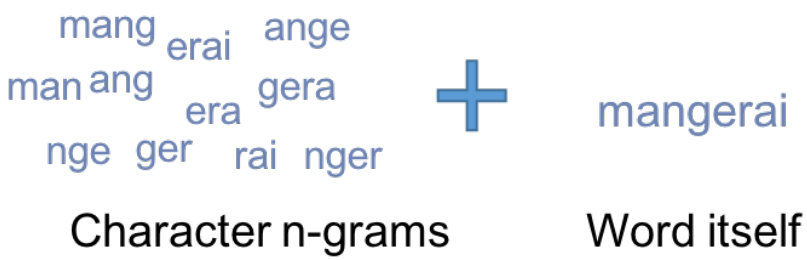


$$\Pr(c|w) = \frac{\exp(h_w^T \theta_w)}{\sum_{v \in V} \exp(h_w^T \theta_c)}, \quad h_w, \theta_v \in \mathbb{R}^d$$

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

- Existen métodos para entrenar sentencias.
- 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