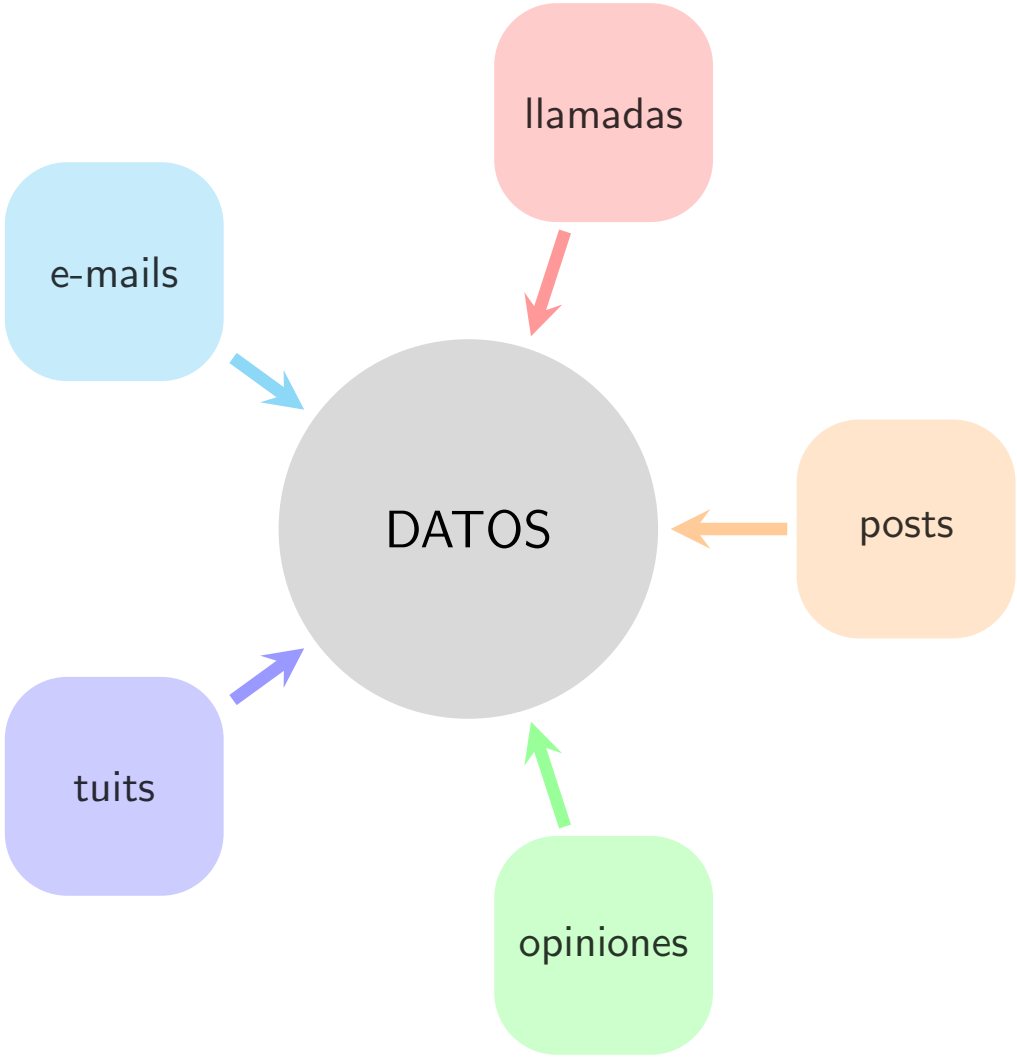
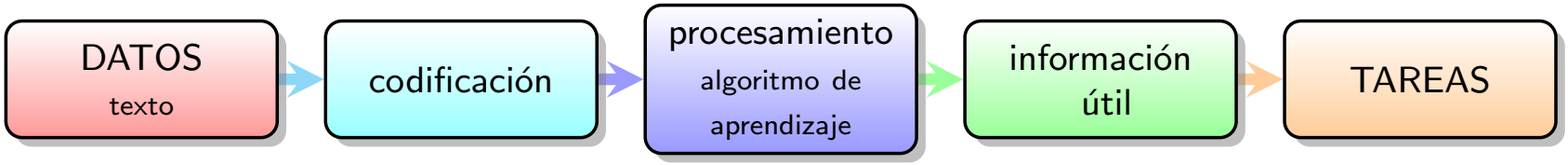
Vectores y el procesamiento de lenguaje natural

Olivia Gutú

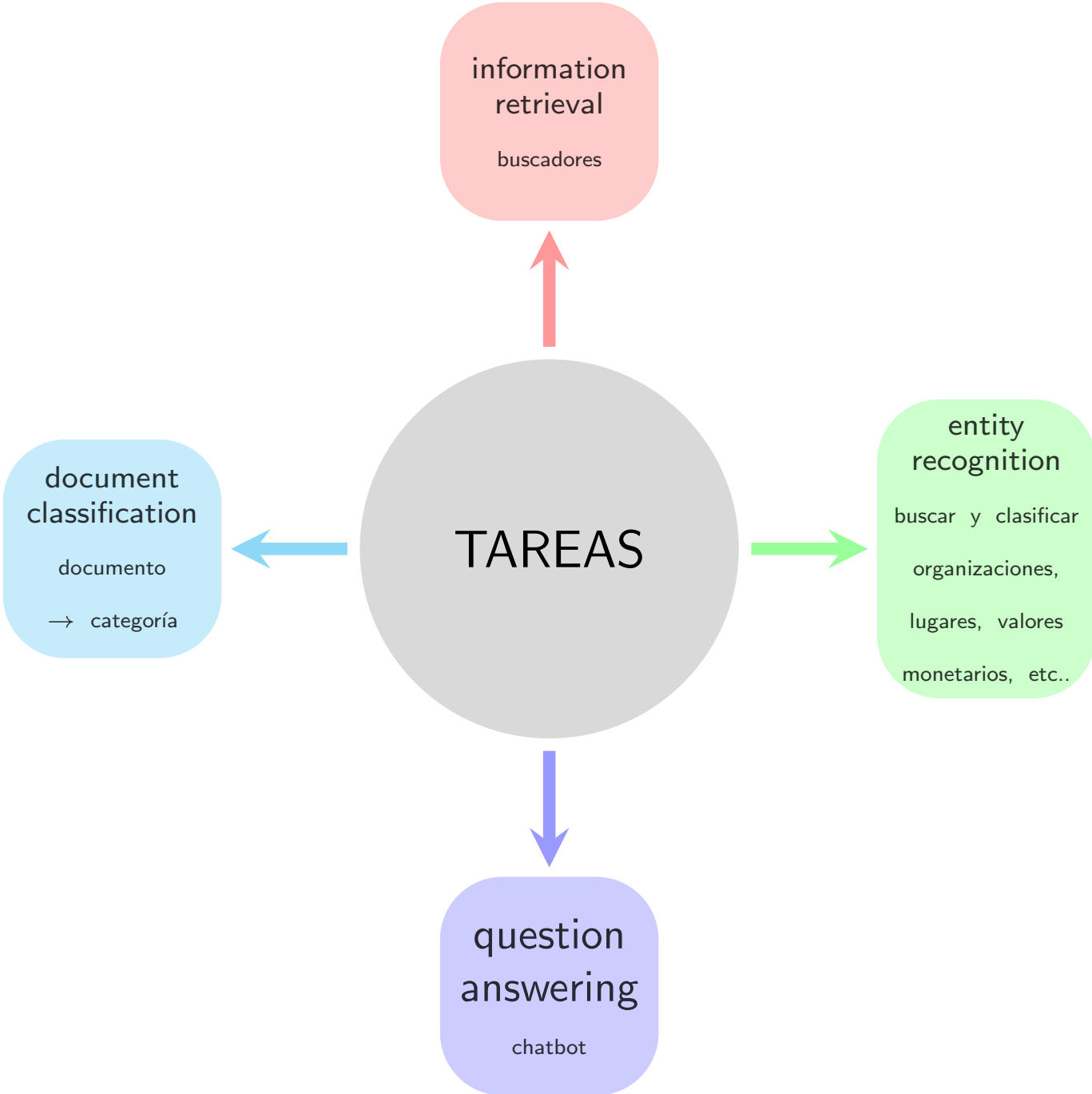
SIDM 2019

PLN


llamadas
e-mails
DATOS
posts
tuits
opiniones


DATOS
texto
codificación
procesamiento
algoritmo de
aprendizaje
información
útil
TAREAS

PLN


information retrieval
buscadores
document classification
documento
→ categoría
TAREAS
entity recognition
buscar y clasificar
organizaciones,
lugares, valores
monetarios, etc..
question answering
chatbot

Interger encoding

denominación de origen	**categorical value**
chairo	1
facha	3
fifí	2
globalista	4
nacionalista	5
rojo	6

- El problema con el **integer encoding** es que da más importancia al **categorical value** que a la categoría.

One-hot encoding

chairo	facha	fifí	globalista	nacionalista	rojo
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

Document-Term matrix

DATOS

D_1: y no hablemos de izquierdas y derechas

D_2: hablemos de globalistas y nacionalistas

D_3: izquierdas nacionalistas

D_4: derechas globalistas

D_5: no nacionalistas

D_6: hablemos de

	de	derechas	globalistas	hablemos	izquierdas	nacionalistas	no	y
D_1:	1	1	0	1	1	0	1	2
D_2:	1	0	1	1	0	1	0	1
D_3:	0	0	0	0	1	1	0	0
D_4:	0	1	1	0	0	0	0	0
D_5:	0	0	0	0	0	1	1	0
D_6:	1	0	0	1	0	0	0	0

TF-IDF matrix

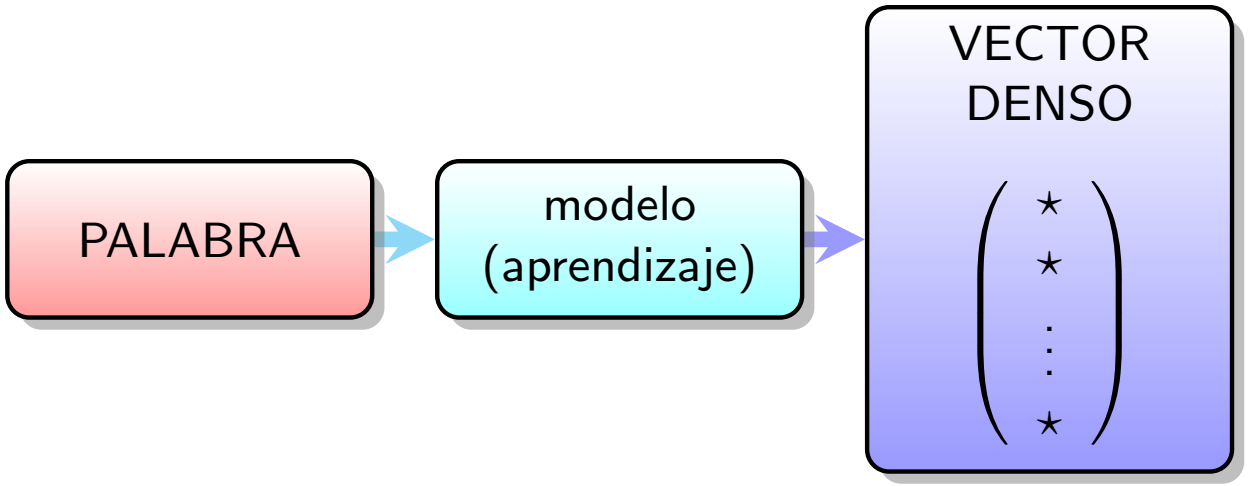
- Palabra muy frecuente en un documento $\rightarrow$ ponderación alta.
- Palabra aparece en muchos documentos $\rightarrow$ ponderación baja.

$$M_{ij} = \begin{array}{c}\text{frecuencia palabra } i \\ \text{en doc. } j\end{array} \times \frac{\text{núm. doc.}}{\text{núm. doc. con la palabra } i}$$

Algunos problemas con este tipo de matrices

- Demasiados ceros dado un gran corpus (matriz dispersa) $\rightarrow$ computacionalmente costoso
- No hay información contextual / semántica incorporada $\rightarrow$ no es adecuada para ciertas tareas
- No escala bien, cuando el número de categorías es demasiado grande $\rightarrow$ vectores de dimensión de decenas o cientos de miles

Word embedding



Word embedding

Word embedding

Modelos de lenguaje

CBOW:

«**A lo largo de un ——— soleado**»➠ día | trayecto | camino

Skip-gram:

día ➠ {es, soleado, hoy, de}

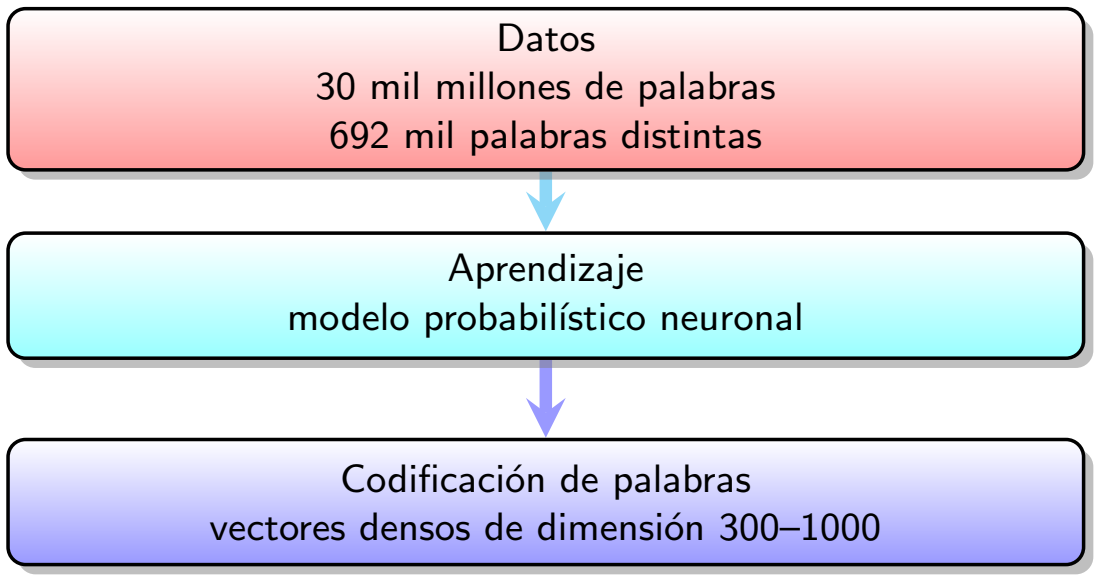
Skip-gram: filosofía lingüística

Hipótesis distribucional:

- "Words that are used and occur in the same contexts tend to surport similar meanings." Harris, Z. (1954).
- "You shall know a word by the company it keeps". Firth, J.R. (1957).

Skip-gram: idea general

Objetivo: Diseñar un algoritmo que a cada palabra w de un vocabulario V le asigne un vector denso $\mathbf{v}(w) \in \mathbb{R}^N$ de tal forma que las entradas del vector contengan información relevante sobre cómo se distribuye la palabra en el lenguaje.



Skip-gram: datos

- Texto: «pese a que Washington negó en su momento su participación en los derrocamientos de gobiernos, los documentos desclasificados años más tarde por sus mismas instituciones revelan lo contrario»
- Tamaño del contexto: por ejemplo ventana de ± 2 palabras

w	y	one-hot-encode(w)	one-hot-encode(y)
pese	a	$(1, 0, \cdots, 0, 0, 0)$	$(0, 1, \cdots, 0, 0, 0)$
pese	que	$(1, 0, \cdots, 0, 0, 0)$	$(0, 0, \cdots, 1, 0, 0)$
a	pese	$(0, 1, \cdots, 0, 0, 0)$	$(1, 0, \cdots, 0, 0, 0)$
a	que	$(0, 1, \cdots, 0, 0, 0)$	$(0, 0, \cdots, 1, 0, 0)$
a	Washington	$(0, 1, \cdots, 0, 0, 0)$	$(0, 0, \cdots, 0, 0, 1)$
que	pese	$(0, 0, \cdots, 1, 0, 0)$	$(1, 0, \cdots, 0, 0, 0)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Skip-gram: modelo

$$W = \begin{pmatrix} \omega_{11} & \omega_{12} & \cdots & \omega_{1|V|} \\ \omega_{21} & \omega_{22} & \cdots & \omega_{2|V|} \\ \vdots & & & \\ \omega_{N1} & \omega_{N2} & \cdots & \omega_{N|V|} \end{pmatrix}$$

- W manda $R^{|V|}$ a R^N
- W evaluada en x_i =one-hot-encode(w_i) es justo la i-ésima columna de W

Se considera:

$$\mathbf{v}_i := W x_i$$

Skip-gram: modelo

- Para cada palabra x_i, cada posición del vector $\mathbf{v}_i = (\omega_{1i}, \omega_{2i}, \ldots, \omega_{Ni})$ representa una característica la cual tiene mayor o menor importancia en relación a como se distribuye en el lenguaje.
- Para cada palabra x_j se considera un vector de pesos $\mathbf{v}'_j = (\omega'_{1j}, \omega'_{2j}, \ldots, \omega'_{Nj})$ para ponderar el valor de cada característica de x_i respecto a x_j:

$$\mathbf{v}'_j \cdot \mathbf{v}_i = \omega'_{1j}\omega_{1i} + \omega'_{2j}\omega_{2i} + \cdots + \omega'_{Nj}\omega_{Ni}$$

- Esta información la puedo guardar en una matriz:

$$W' = \begin{pmatrix} \omega'_{11} & \omega'_{12} & \cdots & \omega'_{1|V|} \\ \omega'_{21} & \omega'_{22} & \cdots & \omega'_{2|V|} \\ \vdots & & & \\ \omega'_{N1} & \omega'_{N2} & \cdots & \omega'_{N|V|} \end{pmatrix}$$

Skip-gram: modelo

- Hasta ahora:

$$x_i \in \mathbb{R}^{|V|} \xrightarrow{W} \mathbf{v_i} \in \mathbb{R}^N \xrightarrow{(W')^T} \begin{pmatrix} \mathbf{v}'_1 \cdot \mathbf{v}_i \\ \mathbf{v}'_2 \cdot \mathbf{v}_i \\ \vdots \\ \mathbf{v}'_{|V|} \cdot \mathbf{v}_i \end{pmatrix} \in \mathbb{R}^{|V|}$$

- Se considera probabilidad de que x_j esté en el contexto de x_i (modelo lineal generalizado, distribución multinomial):

$$p(x_j|x_i) = \frac{\exp \mathbf{v}'_j \cdot \mathbf{v}_i}{\sum_{k=1}^{|V|} \exp \mathbf{v}'_k \cdot \mathbf{v}_i}.$$

Skip-gram: los parámetros que mejor se ajustan a mis datos

one-hot-encode(w)	one-hot-encode(y)
$(1, 0, \cdots, 0, 0, 0)$	$(0, 1, \cdots, 0, 0, 0)$
$(1, 0, \cdots, 0, 0, 0)$	$(0, 0, \cdots, 1, 0, 0)$
$(0, 1, \cdots, 0, 0, 0)$	$(1, 0, \cdots, 0, 0, 0)$
$(0, 1, \cdots, 0, 0, 0)$	$(0, 0, \cdots, 1, 0, 0)$
$(0, 1, \cdots, 0, 0, 0)$	$(0, 0, \cdots, 0, 0, 1)$
$(0, 0, \cdots, 1, 0, 0)$	$(1, 0, \cdots, 0, 0, 0)$
$\vdots$	$\vdots$

DATOS iid

máxima verosimilitud

$$\arg \max_{W,W'} \prod_{\text{DATOS}} p(x_j|x_i)$$

$$\arg \min_{W,W'} -\sum_{\text{DATOS}} \log p(x_j|x_i)$$

Skip-gram: el fondo

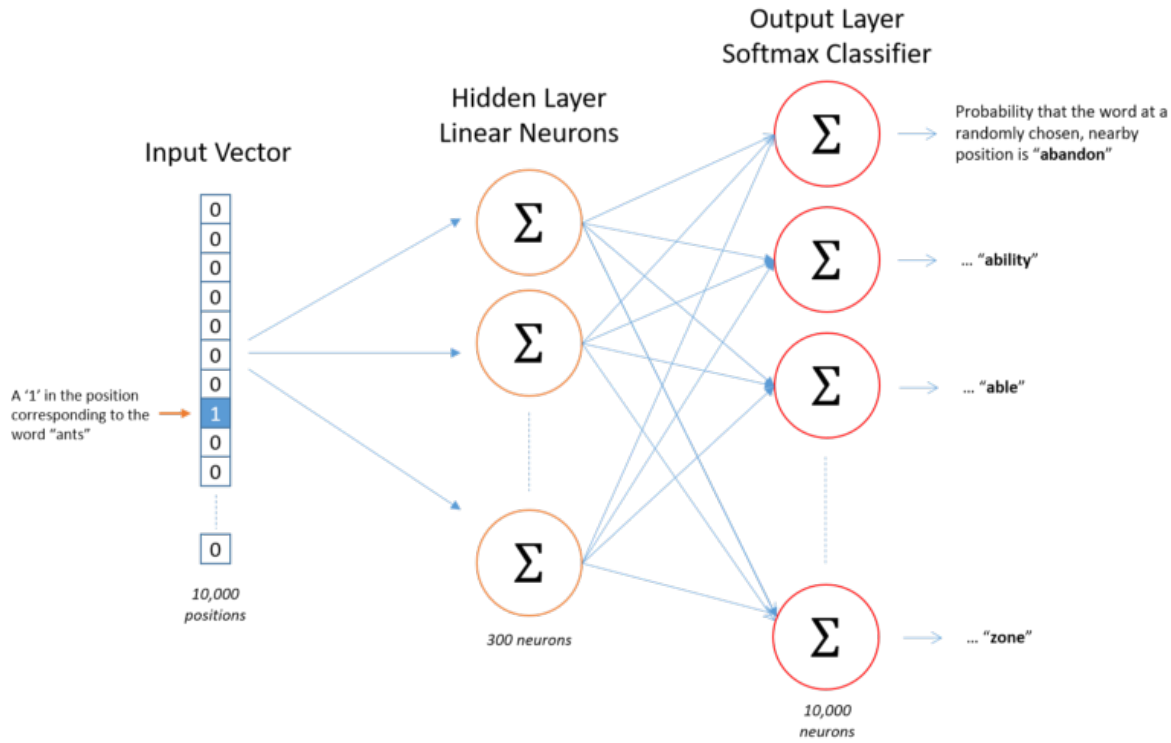
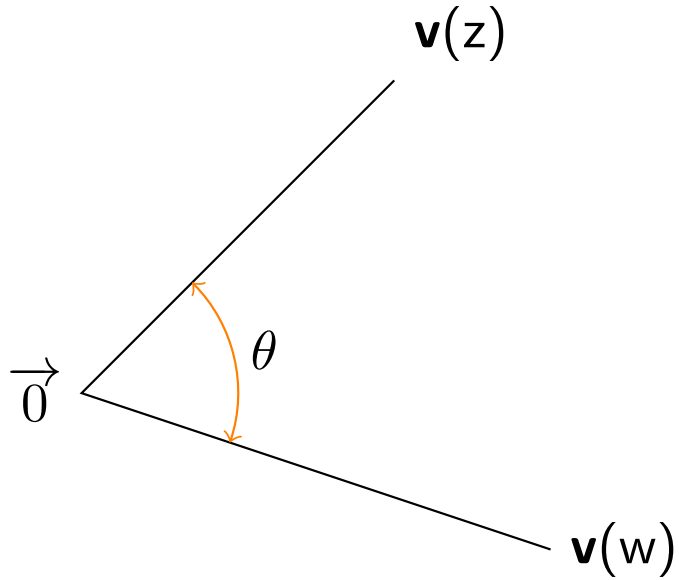


Figura tomada de: https://towardsdatascience.com

Similitud

$$\text{sim}(w, z) = \cos\theta = \frac{\mathbf{v}(w)\cdot\mathbf{v}(z)}{\|\mathbf{v}(w)\|\|\mathbf{v}(z)\|}$$



Similitud

model.similarity('mujer', 'hombre')
0.87441004776040909

model.similarity('gato', 'perro')
0.8850999746465289

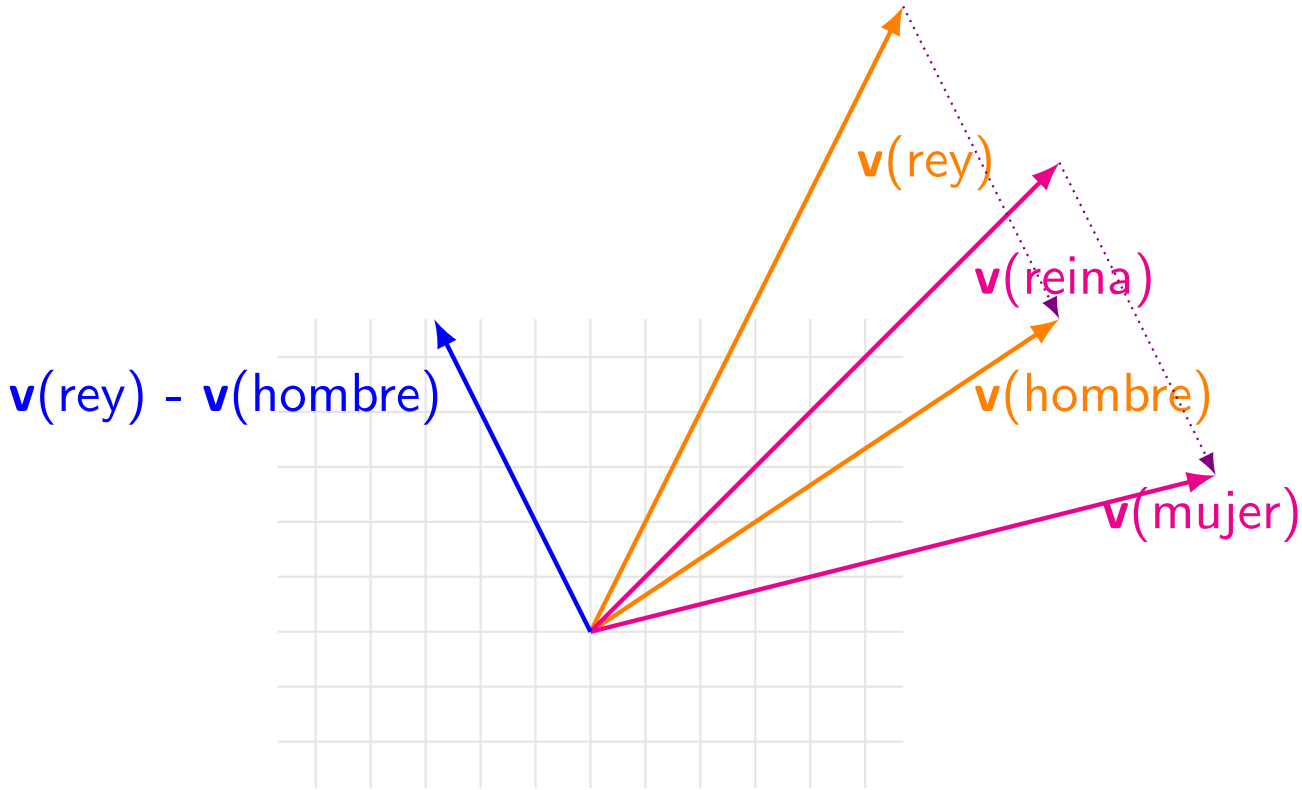
model.similarity('cdmx', 'tren')
0.25187103395420385

model.similarity('rey', 'principe)
0.82048178119674831

model.similarity('rey', 'reina')
0.8018559473097695

Relaciones

$$\mathbf{v}(\text{rey}) - \mathbf{v}(\text{hombre}) + \mathbf{v}(\text{mujer}) \rightarrow \mathbf{v}(\text{reina})$$



Relaciones

model.most_similar(positive=['low', 'lower'], negative=['high'], topn=8)

[('higher', 0.7889082431793213),
('funnier', 0.6868950128555298),
('greater', 0.6742820739746094),
('More', 0.671948432922363 3),
('bigger', 0.665863573551178),
('cheaper', 0.6584792137145996),
('dumber', 0.6579981446266174),
('quicker', 0.6447793245315552)]

Relaciones

model.most_similar(positive=['Paris', 'France'], negative=['Rome'], topn=8)

[('Italy', 0.784386575222015 4),
('Germany', 0.780021190643310 5),
('England', 0.773652017116546 6),
('Japan', 0.765874743461608 9),
('America', 0.762668728828430 2),
('Africa', 0.761730194091796 9),
('Europe', 0.758866906166076 7),
('London', 0.753485858440399 2)]

Relaciones

model.most_similar(positive=['father', 'girl'], negative=['mother'], topn=8)

[('boy', 0.9240204095840454),
('woman', 0.852312088012695 3),
('dog', 0.843636155128479),
('lady', 0.8371353149414062),
('man', 0.835669040679931 6),
('doctor', 0.8223854899406433),
('soldier', 0.787258327007293 7),
('kid', 0.7855633497238159)]

Relaciones

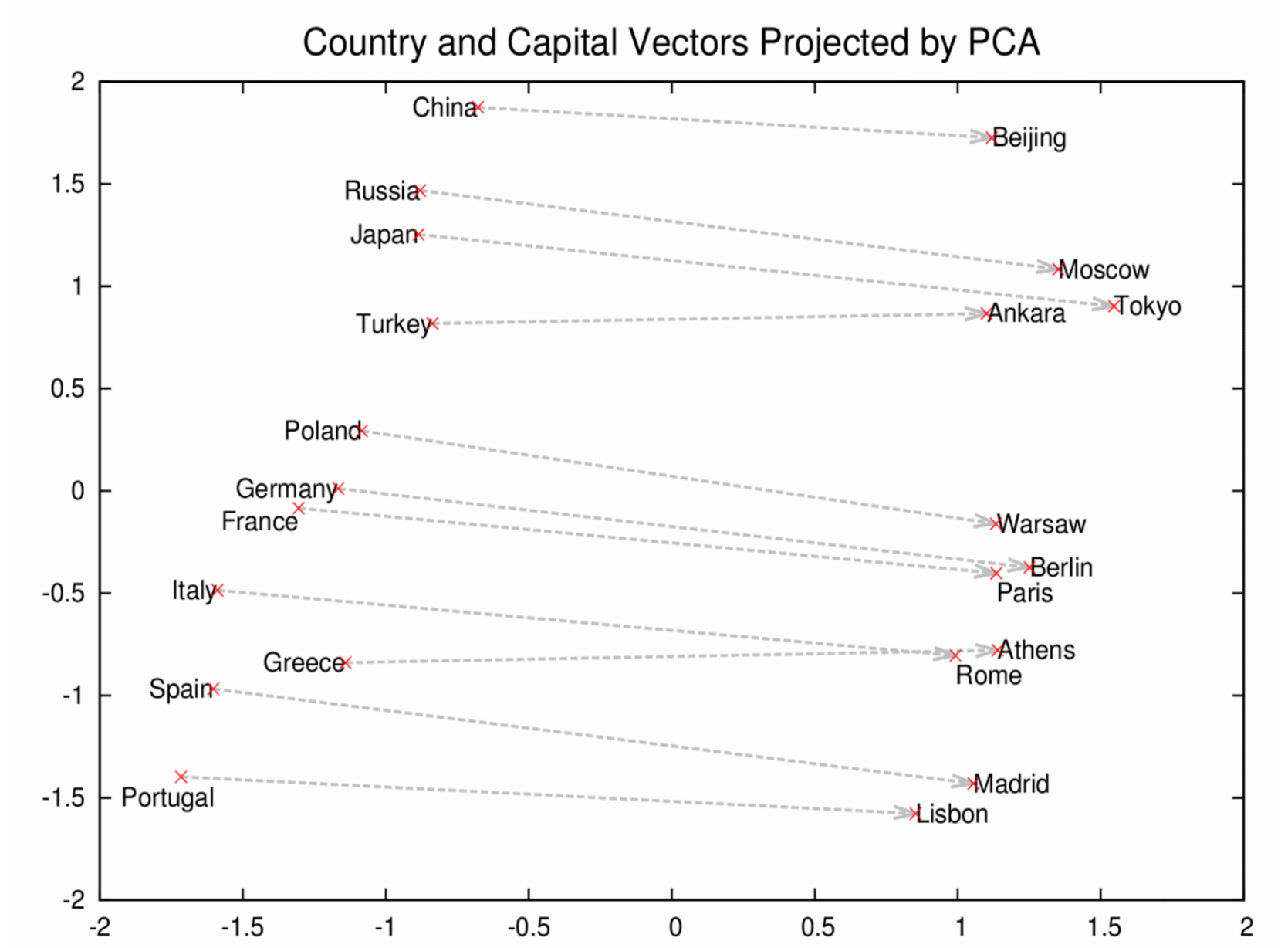
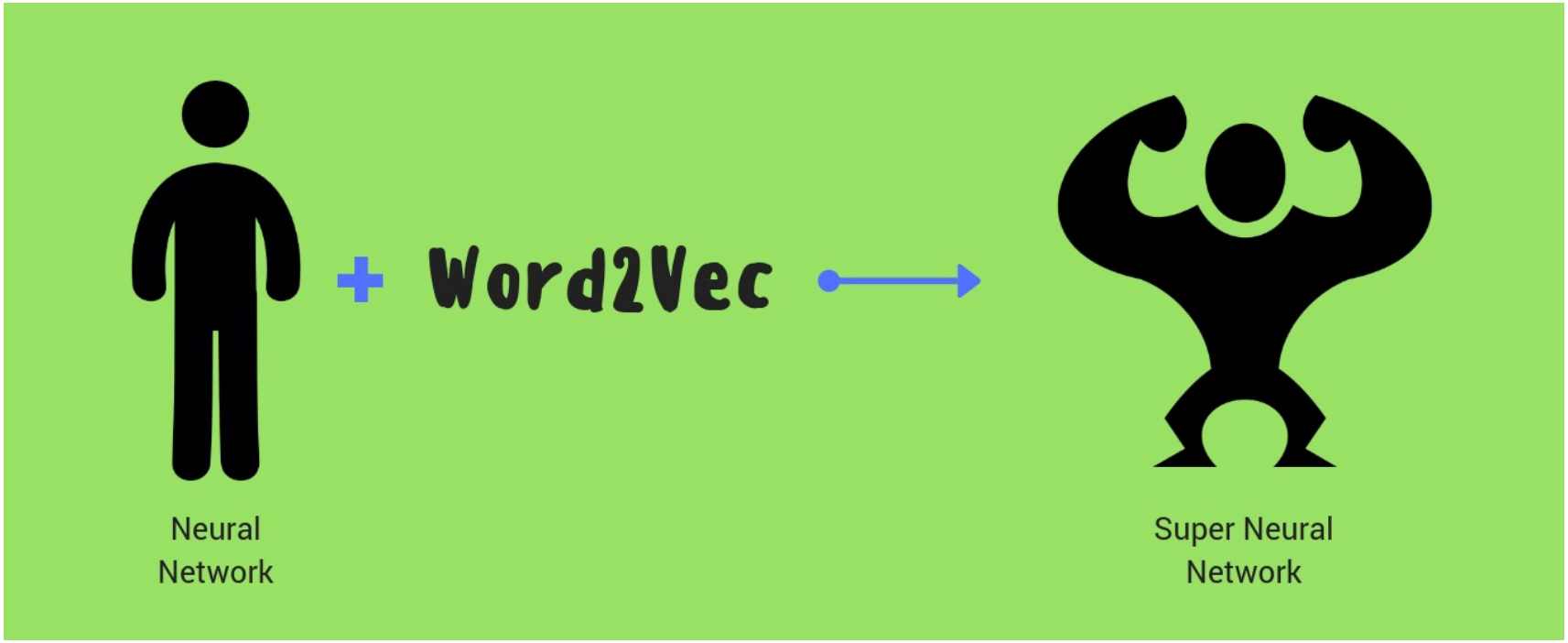


Figura tomada del artículo orgininal de Mikolov et. al (2014)

Conclusión

- Los vectores densos de palabras pueden contener información enorme en comparación con su tamaño.
- Pueden aprender tanto semántica como sintaxis.
- Son amigables para programar, ya que son todos vectores de números.
- La relación entre vectores se puede descubrir con solo álgebra lineal.

Fin



Word2Vec; the Steroids for Natural Language Processing

Figura tomada de: https://hackernoon.com