

# Clase 4: Reducción de la Dimensionalidad

Marcelo Errecalde<sup>1,2</sup>

<sup>1</sup>Universidad Nacional de San Luis, Argentina 

<sup>2</sup>Universidad Nacional de la Patagonia Austral, Argentina 



Curso: Minería de Textos  
Facultad de Informática - Universidad Nacional de La Plata  
23 al 27 de Septiembre de 2019

## Resumen

## 1 Reducción de la dimensionalidad

## 2 Selección de términos

### 3 Transformación del espacio de términos

- Indexado (análisis) de semántica latente

## 4 Otras variantes “densas”

- Embeddings aprendidos de la predicción

## Reducción de la dimensionalidad

- **Problema:** el conjunto de términos  $\mathcal{T}$  que ocurre al menos una vez en una colección de documentos puede ser **enorme**.
  - Esto implica que en una representación vectorial, debamos tratar con dimensiones de vectores en el orden de los **miles** de atributos (decenas o cientos).
  - **Solución:** convertir  $\mathcal{T}$  a otro espacio de términos  $\mathcal{T}'$ , tal que  $|\mathcal{T}'| \ll |\mathcal{T}|$

## Ventajas de reducir la dimensionalidad

- El conjunto de términos reducido permite ocupar mucho **menos memoria** para almacenar los vectores que representan los documentos.
  - El **tiempo** que insume entrenar un clasificador también es menor que con el conjunto de términos original.
  - En algunos casos, la reducción de términos permite eliminar palabras **redundantes** o que sólo introducen **ruido** en el aprendizaje del clasificador.
  - La eliminación de estas palabras puede resultar en un clasificador **más efectivo** (más exacto a la hora de clasificar) que el que se obtenía con el conjunto de términos orginal.

## Enfoques para reducir la dimensionalidad

## **Selección de términos**

Sólo se mantienen los términos más relevantes ( $\mathcal{T}' \subseteq \mathcal{T}$ ).

## **Transformación del espacio de términos**

$\mathcal{T}'$  no es un subconjunto de  $\mathcal{T}$ , e incluso pueden ser atributos de un tipo completa/ distinto.

## Selección de términos

- También llamada **reducción del espacio de términos**
  - Intenta **seleccionar**, del conjunto de términos  $\mathcal{T}$  original, el **subconjunto  $\mathcal{T}'$**  que tiene la más alta efectividad.
  - Principales métodos para la selección de términos
    - métodos de **envoltura (wrapper)**: utilizan el mismo método de aprendizaje que será usado en el clasificador
    - métodos de **filtrado**: se mantienen los términos que reciben la calificación más alta, de acuerdo a una función numérica que mide la **importancia** del término:
      - Ganancia de información
      - Información mutua
      - Chi-cuadrado ( $\chi^2$ )
      - Frecuencia de documento

## Principales funciones (supervisadas) para selección

Function	Denoted by	Mathematical form
DIA association factor	$z(t_k, c_i)$	$P(c_i   t_k)$
Information gain	$IG(t_k, c_i)$	$\sum_{c \in \{c_i, \bar{c}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t, c) \cdot \log \frac{P(t, c)}{P(t) \cdot P(c)}$
Mutual information	$MI(t_k, c_i)$	$\log \frac{P(t_k, c_i)}{P(t_k) \cdot P(c_i)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{ Tr  \cdot [P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)]^2}{P(t_k) \cdot P(\bar{t}_k) \cdot P(c_i) \cdot P(\bar{c}_i)}$
NGL coefficient	$NGL(t_k, c_i)$	$\frac{\sqrt{ Tr  \cdot [P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)]}}{\sqrt{P(t_k) \cdot P(\bar{t}_k) \cdot P(c_i) \cdot P(\bar{c}_i)}}$
Relevancy score	$RS(t_k, c_i)$	$\log \frac{P(t_k   c_i) + d}{P(\bar{t}_k   \bar{c}_i) + d}$
Odds ratio	$OR(t_k, c_i)$	$\frac{P(t_k   c_i) \cdot (1 - P(t_k   \bar{c}_i))}{(1 - P(t_k   c_i)) \cdot P(t_k   \bar{c}_i)}$
GSS coefficient	$GSS(t_k, c_i)$	$P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i)$

## Otras funciones (no supervisadas) para selección

Adecuadas para tareas no supervisadas como clustering

- **Frecuencia de documento**: sólo se mantienen los términos que ocurren en el mayor número de documentos.  
Variantes: se remueven los que ocurren en a lo más  $k$  documentos (o  $k$  veces) en el training set.
  - **Fuerza del término** (Term strength) (Ver [1] y [2])
  - Ranking basado en **entropía** (Entropy-based Ranking) (Ver [2])
  - **Contribución del término** (Term contribution):

$$TC(t) = \sum_{i,j \cap i \neq j} tf - idf(t, d_i) \times tf - idf(t, d_j)$$

## Transformación del espacio de términos

- También llamada **extracción** de términos o **reparametrización**
  - Intenta **sintetizar**, del conjunto de términos  $\mathcal{T}$  original, un **nuevo** conjunto  $\mathcal{T}'$  que tiene la más alta efectividad.
  - Se crean términos **sintéticos/artificiales**

## Métodos conocidos de extracción de términos

- **Indexado de semántica latente**: se obtiene un espacio dimensional reducido cuyas dimensiones se obtienen combinando las dimensiones originales utilizando patrones de **co-ocurrencia**. Las nuevas dimensiones no son intuitivamente **interpretables**.
  - **Clustering/agrupamiento** de términos: los nuevos términos (atributos) representan **grupos** de palabras relacionadas semánticamente.

## Indexado (análisis) de semántica latente

## LSA - Idea:

Descomponer una matriz  $X$  de  $n$  documentos  $\times m$  términos:

	$t_1$	$t_2$	.....	$t_k$	.....	$t_m$
$d_1$						
$d_2$						
$\vdots$						
$d_j$						
$\vdots$						
$d_n$						

$$|D| \times |T|$$

## En 3 matrices

- $U$  de  $n$  documentos  $\times k$  conceptos
  - $S$  de  $k$  conceptos  $\times k$  conceptos
  - $V$  de  $m$  términos  $\times k$  conceptos

#### Indexado (análisis) de semántica latente

## LSA - Idea:

**Descomponer** una matriz  $X$  de  $n$  documentos  $\times m$  términos:

	$t_1$	$t_2$	.....	$t_k$	.....	$t_m$
$d_1$						
$d_2$						
$\vdots$						
$d_j$						
$\vdots$						
$d_n$						

$|D| \times |T|$

## En 3 matrices

	$d_1$	$d_2$	$\dots$	$d_n$	$ D  = k$
$c_1$					
$c_2$					
$\vdots$					
$c_i$					
$\vdots$					
$c_k$					

$$|D| \times K$$

$$\begin{matrix} & S \\ \begin{matrix} c_1 & c_2 & \cdots & c_n \end{matrix} \\ \begin{matrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{matrix} \end{matrix}$$

8 x 8

	V <sub>k</sub>		
	c <sub>1</sub>	c <sub>2</sub>	...
t <sub>1</sub>			
t <sub>2</sub>			
⋮			
t <sub>j</sub>			
⋮			
t <sub>n</sub>			

四

Indexado (análisis) de semántica latente

**tal que ...:**

	<b>X</b>					
	$t_1$	$t_2$	...	$t_i$	...	$t_m$
$d_1$						
$d_2$						
$\vdots$						
$d_j$						
$\vdots$						
$d_n$						

**=** $|D| \times |T|$ **U<sub>k</sub>**

	$c_1$	$c_2$	...	$c_k$
$d_1$				
$d_2$				
$\vdots$				
$d_j$				
$\vdots$				
$d_n$				

**X****S<sub>k</sub>**

	$c_1$	$c_2$	...	$c_k$
$c_1$				
$c_2$				
$\vdots$				
$c_k$				

 $k \times k$ **V<sup>T</sup><sub>k</sub>**

	$t_1$	$t_2$	...	$t_i$	...	$t_m$
$c_1$						
$c_2$						
$\vdots$						
$c_k$						

 $k \times |T|$  $|D| \times k$

## Indexado (análisis) de semántica latente

## **Indexado (análisis) de semántica latente**

Idea:

- Descomponer (factorizar) una matriz  $X$  de  $n$  documentos  $\times m$  términos en 3 matrices:
    - $U$  de  $n$  documentos  $\times k$  conceptos
    - $S$  de  $k$  conceptos  $\times k$  conceptos
    - $V$  de  $m$  términos  $\times k$  conceptos
  - tal que:

$$X \approx USV^T$$

## Indexado (análisis) de semántica latente

# Indexado (análisis) de semántica latente

Idea:

- $U \times S$  es el **embedding** de los **documentos** <—
- $V \times S$  es el **embedding** de los **términos**

 $U_k$ 

	$c_1$	$c_2$	.....	$c_k$
$d_1$				
$d_2$				
⋮				
$d_j$				
⋮				
$d_n$				

 $X$  $S_k$ 

	$c_1$	$c_2$	...	$c_k$
$c_1$				
$c_2$				
⋮				
$c_k$				

 $k \times k$  $|D| \times k$

# Indexado (análisis) de semántica latente

Idea:

- $U \times S$  es el embedding de los documentos
- $V \times S$  es el embedding de los términos <—

$V_k$

	$c_1$	$c_2$	.....	$c_k$
$t_1$				
$t_2$				
:				
$t_j$				
:				
$t_m$				

$X$

$S_k$

	$c_1$	$c_2$	.....	$c_k$
$c_1$				
$c_2$				
:				
$c_k$				

$k \times k$

$|T| \times k$

## Indexado (análisis) de semántica latente

## **Indexado (análisis) de semántica latente**

Idea:

- $U \times S$  es el embedding de los documentos
  - $P \times S$  es el embedding de los términos

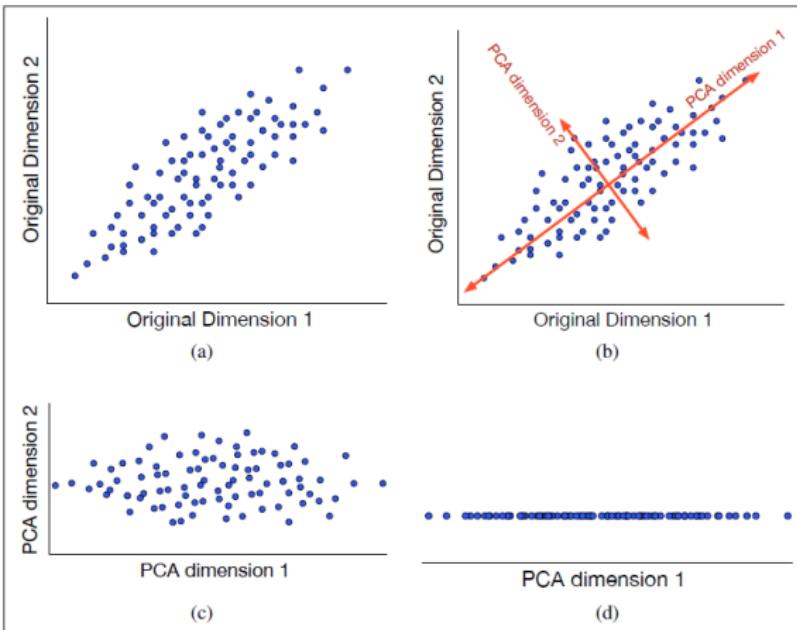
Un documento  $\vec{d}$  de **test  $m$ -dimensional** es fácilmente convertido a un vector  $k$ -dimensional haciendo  $\vec{d} \times V$

#### Indexado (análisis) de semántica latente

## **Indexado (análisis) de semántica latente**

## Consideraciones generales

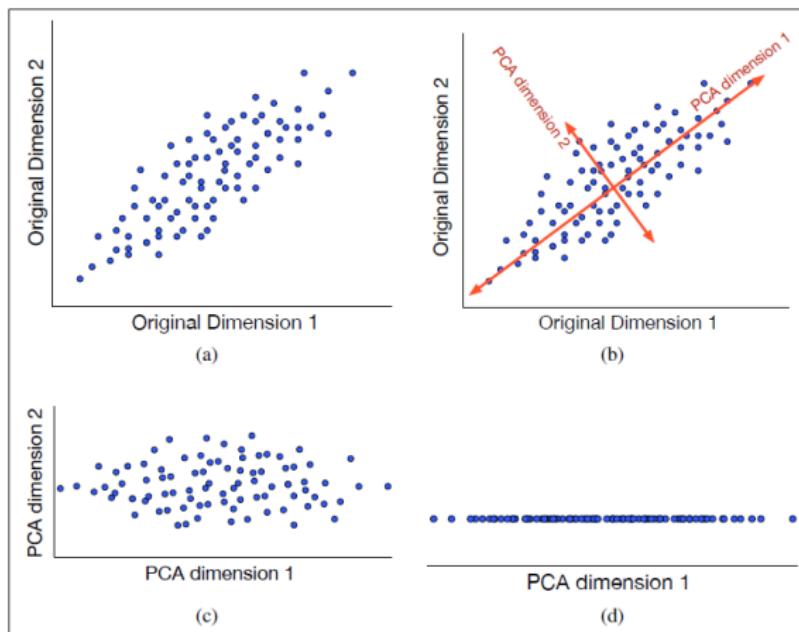
- Es la aplicación del método **singular value decomposition (SVD)** a datos textuales



# Indexado (análisis) de semántica latente

## Consideraciones generales

- SVD encuentra las dimensiones **más importantes** de un data set, aquellas en que los datos **más varían**



#### **Indexado (análisis) de semántica latente**

## **Indexado (análisis) de semántica latente**

## Consideraciones generales

- Surgido del área de IR con los nombres de Latent Semantic Indexing (LSI) o Latent Semantic Analysis (LSA).
  - SVD es parte de una familia de métodos para aproximar matrices con menos dimensiones como Principle Component Analysis (PCA), Factor Analysis, etc.
  - Para colecciones con varias decenas/centenas de miles de términos, valores de  $k$  entre 200 y 400 es a menudo suficiente.
  - Su principal dificultad es que la representación provista por SVD no es muy interpretable.

Indexado (análisis) de semántica latente

## **Visión término × documento (IR) de LSA**

$$\left[ \begin{array}{c} X \\ |V| \times c \end{array} \right] = \left[ \begin{array}{c} W \\ |V| \times m \end{array} \right] \left[ \begin{array}{ccccc} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_m \end{array} \right] \left[ \begin{array}{c} C \\ m \times m \\ m \times c \end{array} \right]$$

$$X = W_k \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix} \begin{matrix} C \\ k \times c \end{matrix}$$

## Indexado (análisis) de semántica latente

## Ejemplo de LSA [3]

$$D = \begin{pmatrix} & \text{lion} & \text{tiger} & \text{cheetah} & \text{jaguar} & \text{porsche} & \text{ferrari} \\ \text{Document-1} & 2 & 2 & 1 & 2 & 0 & 0 \\ \text{Document-2} & 2 & 3 & 3 & 3 & 0 & 0 \\ \text{Document-3} & 1 & 1 & 1 & 1 & 0 & 0 \\ \text{Document-4} & 2 & 2 & 2 & 3 & 1 & 1 \\ \text{Document-5} & 0 & 0 & 0 & 1 & 1 & 1 \\ \text{Document-6} & 0 & 0 & 0 & 2 & 1 & 2 \end{pmatrix}$$

$$D \approx Q\Sigma P^T$$

$$\approx \begin{pmatrix} -0.41 & 0.17 \\ -0.65 & 0.31 \\ -0.23 & 0.13 \\ -0.56 & -0.20 \\ -0.10 & -0.46 \\ -0.19 & -0.78 \end{pmatrix} \begin{pmatrix} 8.4 & 0 \\ 0 & 3.3 \end{pmatrix} \begin{pmatrix} -0.41 & -0.49 & -0.44 & -0.61 & -0.10 & -0.12 \\ 0.21 & 0.31 & 0.26 & -0.37 & -0.44 & -0.68 \end{pmatrix}$$

$$= \begin{pmatrix} 1.55 & 1.87 & 1.67 & 1.91 & 0.10 & 0.04 \\ 2.46 & 2.98 & 2.66 & 2.95 & 0.10 & -0.03 \\ 0.89 & 1.08 & 0.96 & 1.04 & 0.01 & -0.04 \\ 1.81 & 2.11 & 1.91 & 3.14 & 0.77 & 1.03 \\ 0.02 & -0.05 & -0.02 & 1.06 & 0.74 & 1.11 \\ 0.10 & -0.02 & 0.04 & 1.89 & 1.28 & 1.92 \end{pmatrix}$$

## Otras variantes de vectores densos

tf-idf, TCOR y PPMI son representaciones dispersas (ralas)

Los **vectores** de estas representaciones son:

- **largos** (longitudes  $|\mathcal{T}| = 20000$  a  $50000$ )
  - **dispersos** (la mayoría de los elementos son  $0$ )

## Otras variantes de vectores densos

**LSA** es una alternativa que utiliza **vectores densos**, los cuales son:

- **cortos** (longitudes de 50 a 1000)
- **densos** (la mayoría de los elementos son  $\neq 0$ )

Pero existen otros enfoques con **vectores densos**:

- Otras variantes de **descomposición de matrices** (PLSA, etc)
- Vectores **aprendidos** de tareas de predicción (**word2vec**, **GloVe**, **fastText**, etc)

## Embeddings aprendidos de la predicción

## Word2vec



**word2vec** [Mikolov et. al]

Disponible en:

<https://code.google.com/archive/p/word2vec/>

## Embeddings aprendidos de la predicción

## Word2vec



**word2vec** [Mikolov et. al]

Disponible en:

<https://code.google.com/archive/p/word2vec/>

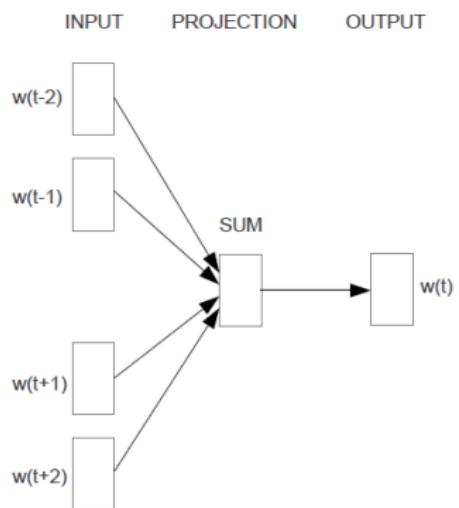
Embeddings aprendidos de la predicción

## Word2vec

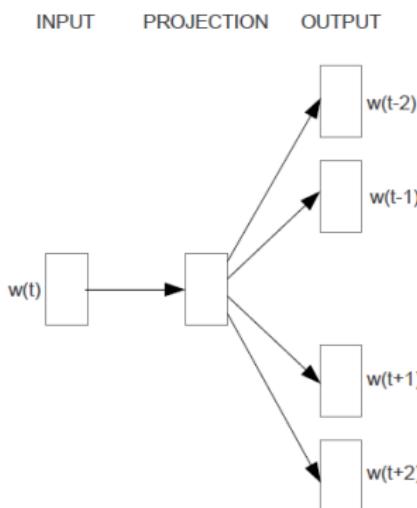
- El método **más popular** de **embedding** en la actualidad.
  - A diferencia de los enfoques **distribucionales**, la idea es **predecir** más que **contar**
  - Se **aprende** un **predictor** (**clasificador**), y como **efecto colateral** se obtienen los “**embeddings**” (**vectores**) que representan las palabras.
  - Enfoque bastante **eficiente** para aprender

## Embeddings aprendidos de la predicción

## Enfoques en Word2vec



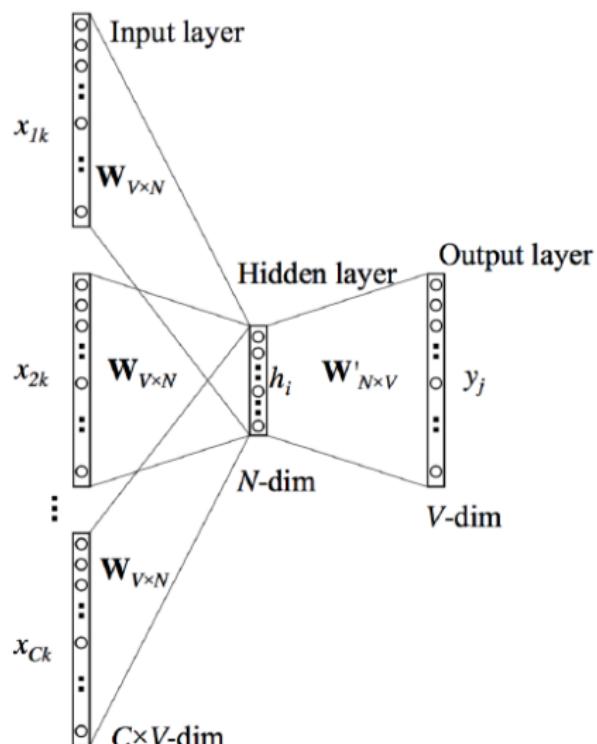
CBOW



## Skip-gram

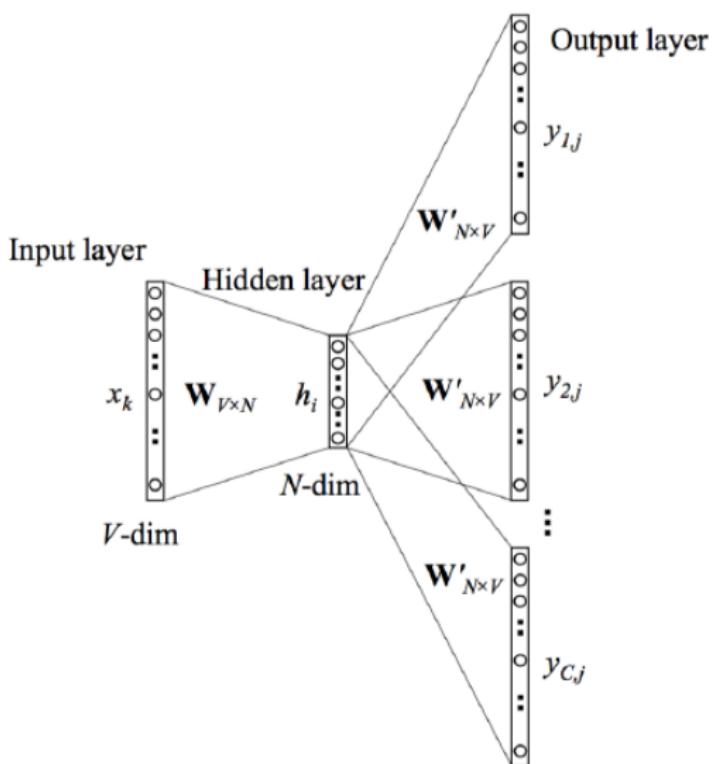
Embeddings aprendidos de la predicción

## Detalles de CBOW



Embeddings aprendidos de la predicción

## Detalles de Skip-gram



## Embeddings aprendidos de la predicción

## Word2vec

- En lugar de **contar** cuan a menudo cada palabra  $w$  ocurre **cerca** de “apricot”
  - **Entrenar** una clasificador sobre una tarea de clasificación binaria:
    - ¿es **probable** que  $w$  aparezca cerca de “apricot”?
  - En realidad **no nos interesa** demasiado esa tarea ....
    - ... pero tomaremos **los pesos** del clasificador aprendido como los **embeddings** de las **palabras**

## Word2vec

- Idea (**brillante**) tomada de **modelos** de lenguajes **neuronales**
  - usar el texto bajo análisis como datos (**supervisados**) implícitos
  - si una palabra  $w$  aparece “cerca” de la palabra “apricot”, se convierte en un caso **positivo** de su ocurrencia cerca de la misma.
- Esta idea **no requiere** datos etiquetados **manualmente**

Embeddings aprendidos de la predicción

## Word2vec: la variante de *Skip-Gram*

Una de las variantes provistas en el paquete **word2vec** es “*skip-gram with negative sampling*” (**SGNS**)

## Algoritmo Skip-Gram

- 1 Tomar la **palabra objetivo** y una **palabra de contexto vecina** como **ejemplos positivos**.
  - 2 Tomar al azar otras palabras en el corpus para obtener **muestras negativas**
  - 3 Usar **regresión logística** para entrenar un clasificador que distinga entre esos dos casos
  - 4 Usas los **pesos** aprendidos como **embeddings** de las palabras

Embeddings aprendidos de la predicción

## Entrenamiento de *Skip-Gram*

Sentencia de entrenamiento:

... lemon, a tablespoon of apricot jam a pinch ...  
[c1 c2 obj c3 c4]

Asumamos que las palabras de contexto son aquellas que están +/- 2 en la ventana de la palabra objetivo

Embeddings aprendidos de la predicción

## Objetivo de *Skip-Gram*

Dada una tupla  $(o, c)$  (**objetivo**,**contexto**)

- (apricot, jam)
- (apricot, aardvark)

Retornar la probabilidad que  $c$  sea una palabra de contexto real

$$P(+|o, c)$$

$$P(-|o, c)$$

Embeddings aprendidos de la predicción

## ¿Cómo calcular la probabilidad $P(+|o, c)$ ?

### Intuitivamente

- Las palabras aparecerán **más probablemente** cerca de **palabras similares**
- Modelar **similitud** como **producto** (escalar) entre vectores!!
- $sim(o, c) \propto o \cdot c$

### Problema

- El producto entre vectores no es una probabilidad !!! (ni tampoco lo es el coseno)
- **Idea:** usar la **sigmoide** ( $\sigma(x) = \frac{1}{1+e^{-x}}$ ), que garantiza que  $\sigma(x)$  está entre 0 y 1

## Embeddings aprendidos de la predicción

## Convirtiendo producto entre vectores en probabilidad

$$P(+|o, c) = \frac{1}{1 + e^{-o \cdot c}}$$

$$P(-|o,c) = 1 - P(+|o,c)$$

$$= \frac{1}{1 + e^{-o \cdot c}}$$

Para todas las palabras de contexto (asumiendo que son independientes)

$$P(+|o, c_{1:k}) = \prod_{i=1}^k \frac{1}{1 + e^{-o \cdot c_i}}$$

$$\log P(+) | o, c_{1:k}) = \sum_{i=1}^k \log \frac{1}{1 + e^{-o \cdot c_i}}$$

Embeddings aprendidos de la predicción

## Entrenamiento de *Skip-Gram*

Sentencia de entrenamiento:

... lemon, a tablespoon of apricot jam a pinch ...  
[c1      c2      obj      c3 c4]

- Datos de entrenamiento: pares de entrada/salida centrados en **apricot**
- Asumimos una ventana de +/- 2 palabras
- **ejemplos positivos (+)**: (apricot,tablespoon), (apricot,of), etc
- **ejemplos negativos (-)**: (apricot,aardvark), (apricot,puddle), (apricot,where), (apricot,coaxial), (apricot,twelve), (apricot,hello), (apricot,dear), etc

## Embeddings aprendidos de la predicción

## Escenario

- Representemos las palabras como vectores de alguna longitud (digamos 300), inicializados en forma aleatoria.
  - Así, comenzamos con  $300 \times |V|$  parámetros aleatorios
  - Sobre el conjunto de entrenamiento completo, nos gustaría ajustar aquellos vectores de palabras tal que:
    - Maximizar la similitud de los pares **palabra objetivo, palabra de contexto** ( $o, c$ ) tomados de los datos positivos.
    - Minimizar la similitud de los pares ( $o, c$ ) tomados de los datos negativos.

## Aprendizaje del clasificador

- Proceso iterativo. Pesos iniciales aleatorios o en 0.
  - Luego, **ajustar los pesos** de las palabras de manera tal de:
    - hacer los pares positivos más probables
    - y los pares negativos más probables
  - sobre el conjunto de entrenamiento completo

Criterio Objetivo: maximizar ...

$$\sum_{(o,c) \in +} \log P(+ | (o, c)) + \sum_{(o,c) \in -} \log P(- | (o, c))$$

Maximizar la etiqueta + para los pares de los datos de entrenamiento positivos y los - para los pares de muestra de los datos negativos.

## Embeddings aprendidos de la predicción

## **Esquema general de la arquitectura de aprendizaje**

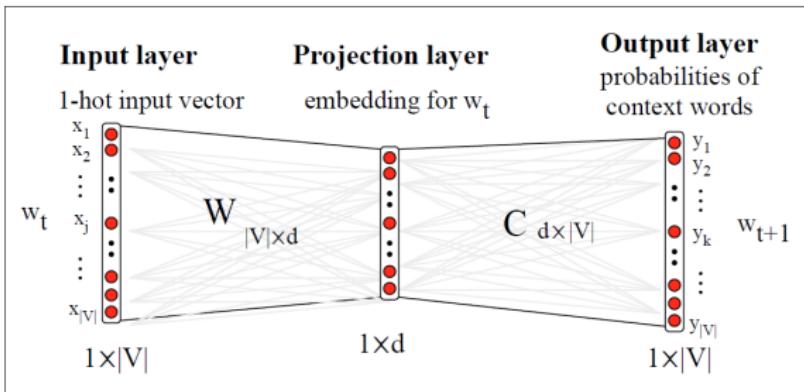
- Entrenamiento x **descenso del gradiente**.
  - Codificación de las palabras **one-hot**.
  - Aprende 2 matrices de embeddings separadas (**W** y **C**).
  - Se usa **W** (“tirando” **C**) o se combinan de alguna manera.

$$w_0 \ w_1 \ \dots \ w_j \ \dots \ w_{|V|}$$

Embeddings aprendidos de la predicción

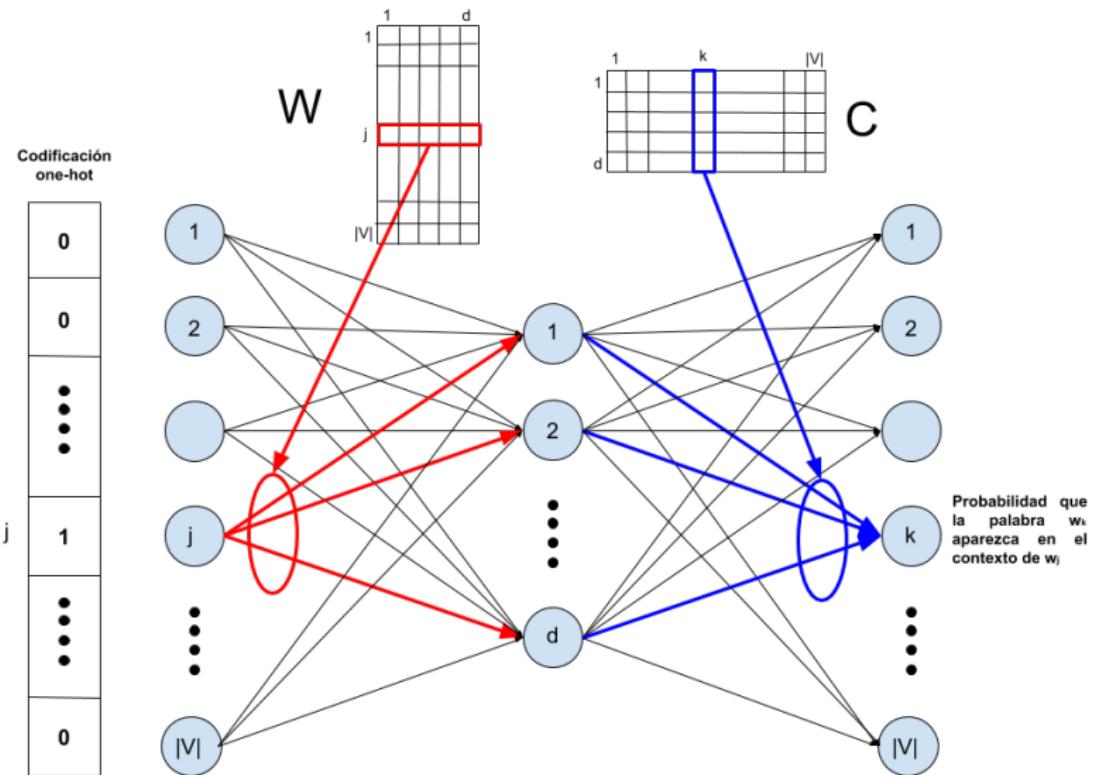
## Esquema general de la arquitectura de aprendizaje

- Entrenamiento x **descenso del gradiente**.
- Codificación de las palabras **one-hot**.
- Aprende 2 matrices de embeddings separadas (**W** y **C**).
- Se usa **W** ("tirando" **C**) o se combinan de alguna manera.



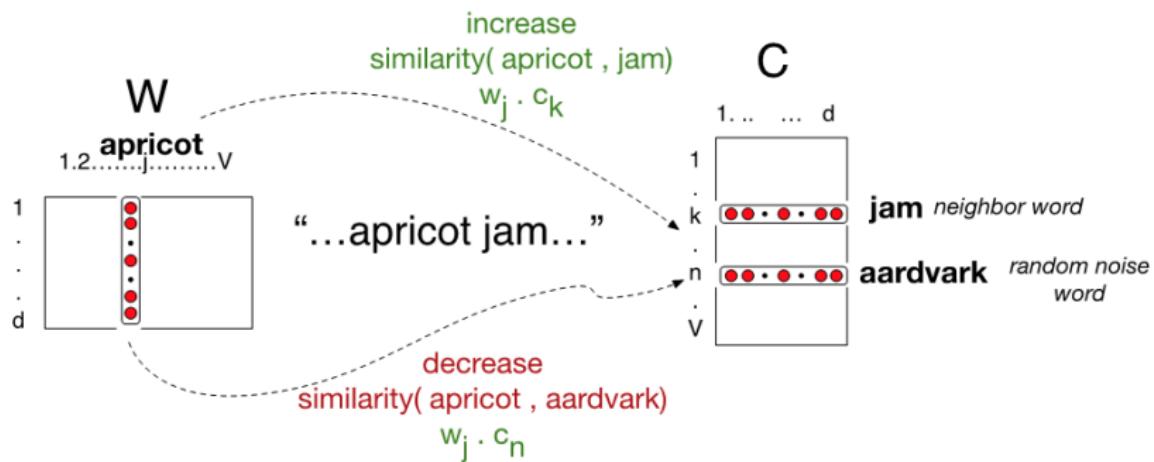
Embeddings aprendidos de la predicción

## Cómo se relacionan $W$ y $C$



Embeddings aprendidos de la predicción

## Cómo se relacionan $W$ y $C$



## Resumen

### Aprendizaje de embeddings word2vec (skip-gram)

- Comenzar con  $|V|$  vectores aleatorios de dimensión 300 como embeddings iniciales
- Usar el (sumamente básico) algoritmo de **regresión logística**
  - Tomar un corpus y usar los pares que co-ocurren (cercanamente) como ejemplos **positivos (+)**
  - Tomar los pares que no co-ocurren como ejemplos **negativos (-)**
  - **Entrenar** el clasificador para distinguir estos casos, ajustando lentamente los embeddings para mejorar el desempeño del clasificador
  - Descartar el clasificador y mantener los embeddings

## Evaluación de embeddings

### Evaluación extrínseca sobre *otras tareas*

Agregarlos como “features” en otras tareas de NLP (análisis de sentimiento, perfilado de autor, etc) y ver si esto mejora el desempeño sobre algun otro modelo

### Evaluación intrínseca (*similitud entre palabras*)

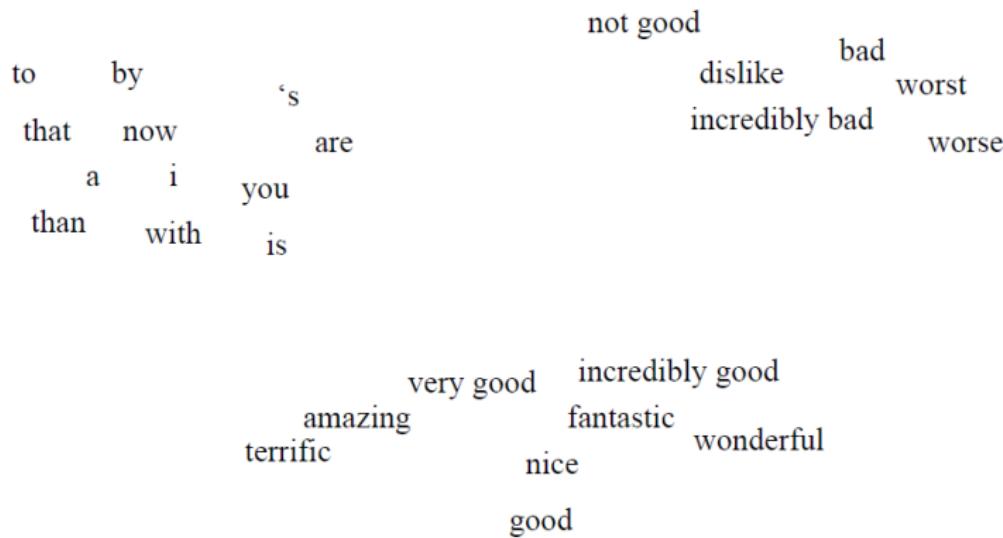
Correlación con ratings de similitud asignados por humanos

- WordSim-353 (Finkelstein et al., 2002)
- SimLex-999 (Hill et al., 2015)
- Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset:** Levied is closest in meaning to: imposed, believed, requested, correlated

Embeddings aprendidos de la predicción

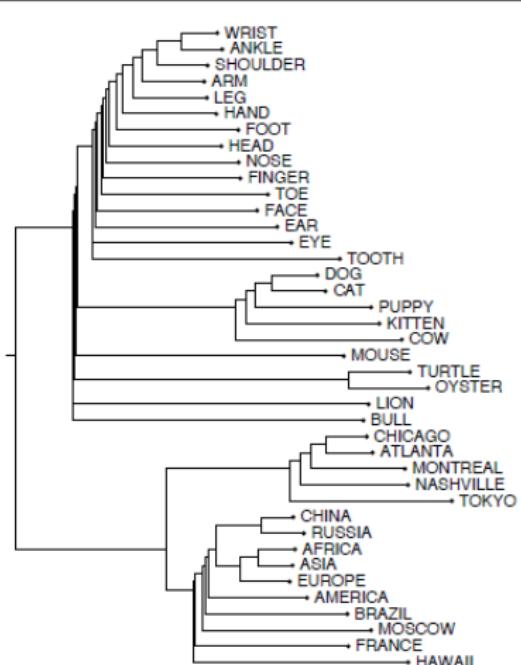
## Visualización de embeddings

Proyección bi-dimensional (**t-SNE**) - dimensión original 60 ([Li et al. (2015)])



Embeddings aprendidos de la predicción

## Visualización de embeddings (II)



Embeddings aprendidos de la predicción

## Propiedades (semánticas) de los embeddings

Tamaños de la ventana de contexto (**C**):

- **Cortos**: las palabras más similares a la palabra objetivo son **semánticamente similares** y con el **mismo POS**.
- **Largos**: las palabras más similares a la palabra objetivo están **relacionadas al tópico** pero no son similares.

Embeddings aprendidos de la predicción

## Propiedades (semánticas) de los embeddings

Habilidad para capturar significados relacionales!!!!:

- $\text{vector}(\text{'king'}) - \text{vector}(\text{'man'}) + \text{vector}(\text{'woman'}) \approx \text{vector}(\text{'queen'})$
  - $\text{vector}(\text{'Paris'}) - \text{vector}(\text{'France'}) + \text{vector}(\text{'Italy'}) \approx \text{vector}(\text{'Rome'})$

Embeddings aprendidos de la predicción

## Propiedades (semánticas) de los embeddings (II)

Cambios en el significado (histórico) de las palabras:

