

Clase 9: Aprendizaje de Representaciones y Aprendizaje Profundo

Marcelo Luis Errecalde^{1,2}

¹Universidad Nacional de San Luis, Argentina

²Universidad Nacional de la Patagonia Austral, Argentina
e-mails: merreca@unsl.edu.ar, merrecalde@gmail.com



Curso: Minería de Datos
Universidad Nacional de San Luis - Año 2018

Resumen

1 Introducción

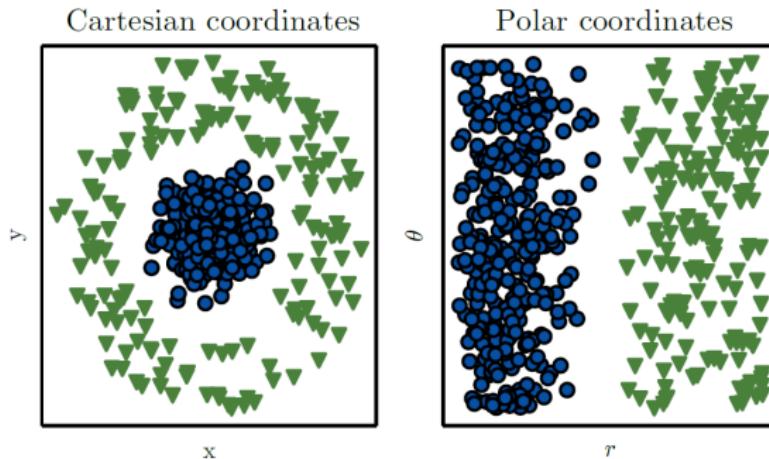
2 Aprendizaje profundo

3 Aprendizaje de representaciones en PLN (Word embeddings)

- Representación distribucional de términos (BoC)
- Embeddings aprendidos de la predicción

Representación de los datos

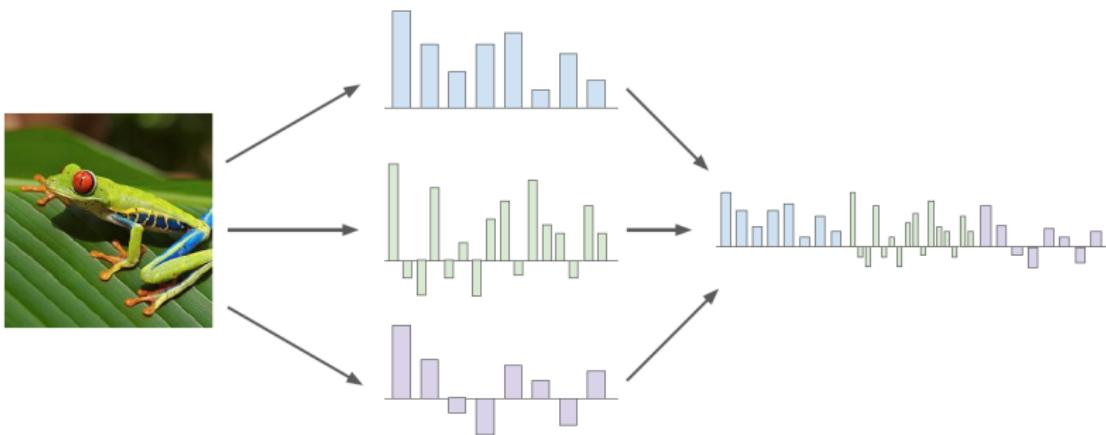
- El desempeño de los algoritmos de aprendizaje depende fuertemente de la **representación** de los datos que le son dados.
- Ejemplo: usar un clasificador lineal con datos representados en sistemas de coordenadas distintos:



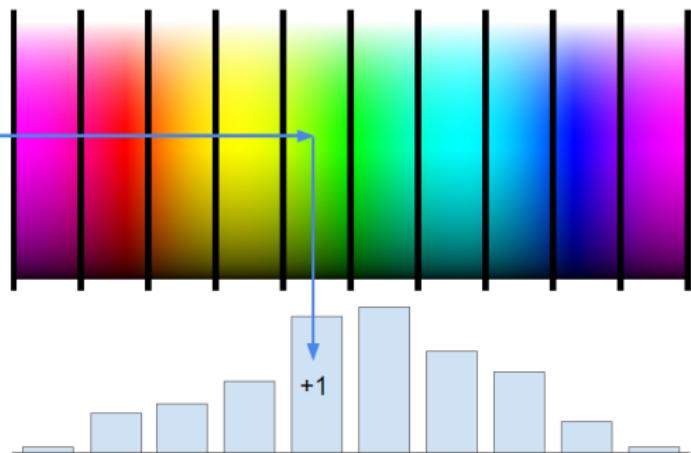
Representación de los datos

- Cada pieza de información incluida en la representación de un objeto es una **característica (feature)**
- Los algoritmos de aprendizaje aprenden modelos que correlacionan las **características** representadas con los **distintos resultados**
- Así, la **elección de las características** a usarse es un punto **crucial**
- Veamos algunos ejemplos con **imágenes** y **textos**

Características de una imagen



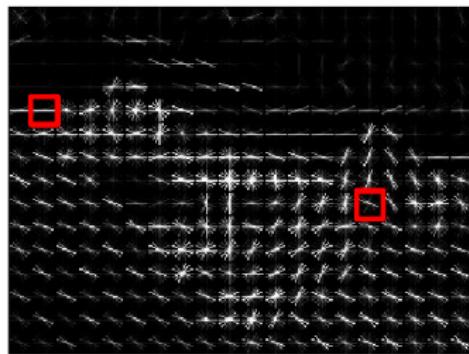
Ejemplo: histograma de colores



Ejemplo: histograma de gradientes orientados (HoG)



Divide image into 8x8 pixel regions
Within each region quantize edge direction into 9 bins



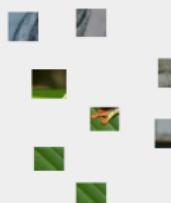
Example: 320x240 image gets divided into 40x30 bins; in each bin there are 9 numbers so feature vector has $30 \times 40 \times 9 = 10,800$ numbers

Ejemplo: Bag of (Visual) Words

Step 1: Build codebook



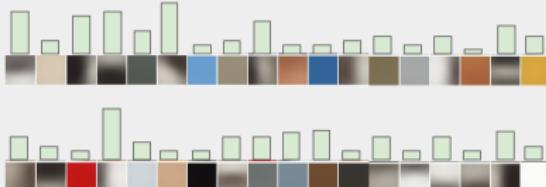
Extract random patches



Cluster patches to form "codebook" of "visual words"

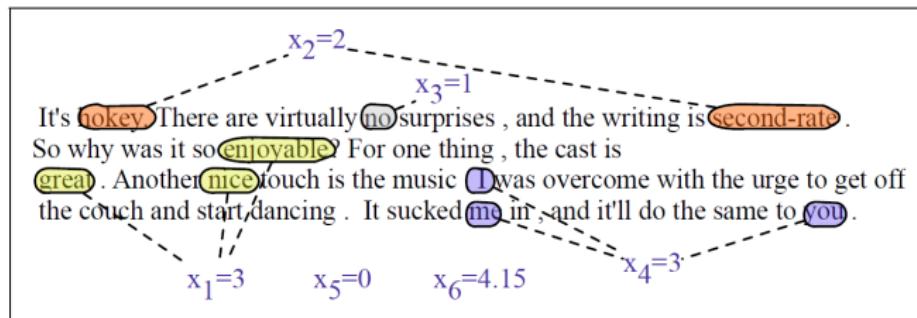


Step 2: Encode images



Características de un texto

“Mini” documento de test: extracto de crítica



Features del documento:

Var	Definition	Value
x_1	$\text{count}(\text{positive lexicon}) \in \text{doc}$	3
x_2	$\text{count}(\text{negative lexicon}) \in \text{doc}$	2
x_3	$\begin{cases} 1 & \text{if } \text{"no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	$\text{count}(1\text{st and 2nd pronouns} \in \text{doc})$	3
x_5	$\begin{cases} 1 & \text{if } \text{"!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	$\ln(64) = 4.15$	$\ln(64) = 4.15$

Ejemplo en Wikipedia

Information Processing and Management 54 (2018) 1169–1181



Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman



Quality flaw prediction in Spanish Wikipedia: A case of study with verifiability flaws

Edgardo Ferretti^{a,b}, Leticia Cagnina^{a*,a,b,c}, Viviana Paiz^a, Sebastián Delle Donne^a, Rodrigo Zacagnini^a, Marcelo Errecalde^{a,b}

^a Departamento de Informática, Universidad Nacional de San Luis (UNSL), Ejército de los Andes 950, San Luis, Argentina

^b Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (UNSL), Argentina

^c Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina



Ejemplo en Wikipedia

Table 1
Features that comprise the document model.

Feature	Description
<i>Content-based</i>	
Character count	Number of characters in the text (no spaces).
Word count	Number of words in the plain text.
Sentence count	Number of sentences in the plain text.
Word length	Average word length in characters.
Sentence length	Average sentence length in words.
Paragraph count	Number of paragraphs.
Paragraph length	Average paragraph length in sentences.
Longest word length	Length in characters of the longest word.
Longest sentence length	Number of words in the longest sentence.
Shortest sentence length	Number of words in the shortest sentence.
Long sentence rate	Percentage of long sentences. A long sentence is defined as containing at least 30 words.
Short sentence rate	Percentage of short sentences. A short sentence is defined as containing at most 15 words.
Longest subsection length	Length in words of the longest subsection.
Shortest subsection length	Length in words of the shortest subsection.
Subsections length	Total number of words in the article's subsections.
Average subsection length	Average number of words per subsection.
Longest subsubsection	Length in words of the longest subsubsection.
Shortest subsubsection	Length in words of the shortest subsubsection.
Subsubsections length	Total number of words in the article's subsubsections.
Average subsubsections	Average number of words per subsubsection.
<i>Structure-based</i>	
Section count	Number of sections.
Subsection count	Number of subsections.
Subsubsection count	Number of subsubsections.
Heading count	Number of sections, subsections and subsubsections.
Section nesting	Average number of subsections per section.
Subsection nesting	Average number of subsubsections per subsection.
Reference Sections Count	Number of reference sections, e.g. "References", "Footnotes", "Sources", "Bibliography".
Mandatory Sections Count	Number of mandatory sections, e.g. "See also".
Related page count	Number of related pages, e.g. "Further reading", "See also", etc.
Lead length	Number of words in the lead section (text before the first heading).
Lead rate	Percentage of words in the lead section.
Image count	Number of images.
Image rate	Ratio of image count to section count.
Link count	Every occurrence of a link (introduced with two open square brackets) in the unfiltered text.
Link rate	Percentage of links.
Table count	Number of tables.
Reference count	Number of all references using the <code><ref>...</ref></code> syntax.
Reference section rate	Ratio of reference count to the accumulated section, subsection and subsubsection count.
Reference word rate	Ratio of reference count to word count.
Unique reference count	Number of unique references using the <code><ref>...</ref></code> syntax.
Reference ratio	Ratio between the reference word rate of the article and the maximum reference word rate found in the dataset.
Templates-count	Number of (different) Wikipedia templates.

Ingeniería de características

- Los enfoques previos, donde las características suelen ser determinadas **a mano** (ingeniería de características) suelen ser muy efectivos si las características son **correctamente seleccionadas**
- El proceso usual, es que para las características diseñadas, se utiliza un proceso de **extracción de características** para representar los datos **y luego** el proceso de aprendizaje trabaja sobre esas representaciones:



Ingeniería de características

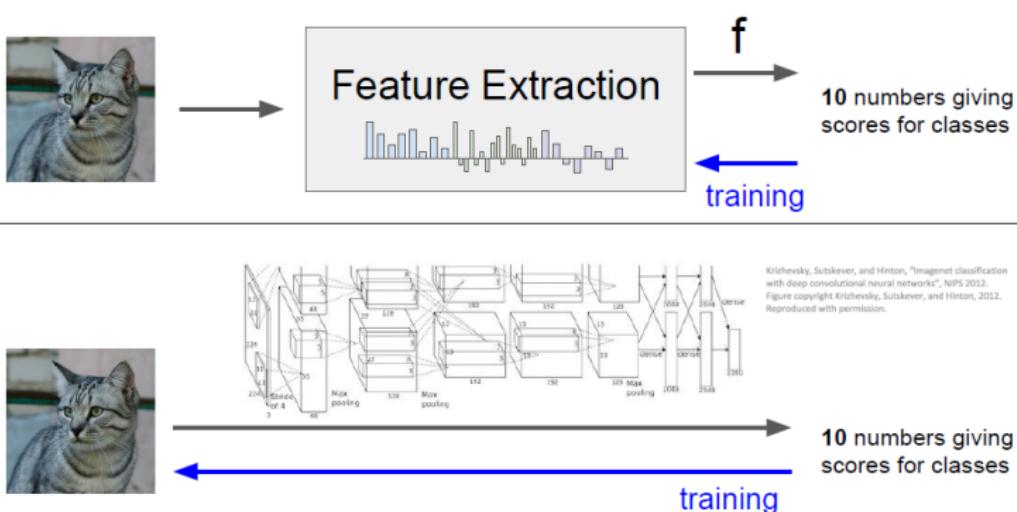
- Notar que en este caso el **proceso de aprendizaje** no tiene **ninguna influencia** en cómo las **características** son definidas.
- Observar que en **muchos casos** puede ser **muy dificultoso** saber **qué características** deberían ser extraídas (ejemplo: detección de autos en fotos).



Aprendizaje de representaciones

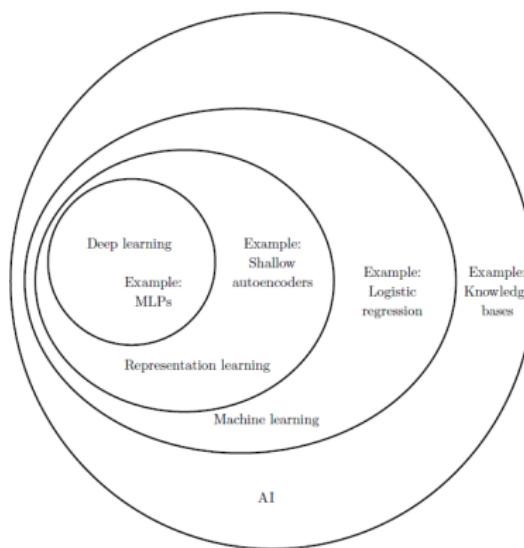
- **Alternativa:** usar el aprendizaje automático **no sólo** para descubrir la correspondencia entre la **representación** y la **salida** sino también la **representación en sí misma** ⇒ **aprendizaje de representaciones.**
- **Representaciones aprendidas** pueden obtener mucho mejor desempeño que **representaciones diseñadas a mano.**

Características diseñadas vs características aprendidas



Aprendizaje de representaciones y aprendizaje profundo

- El **aprendizaje profundo** es un tipo de **aprendizaje de representaciones**
- Pero es bueno considerar su relación con otros **conceptos relacionados**



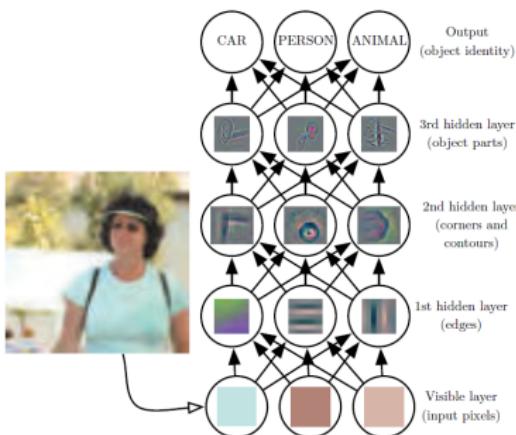
Aprendizaje de representaciones y aprendizaje profundo

En esta clase veremos ejemplos de **aprendizaje de representaciones**

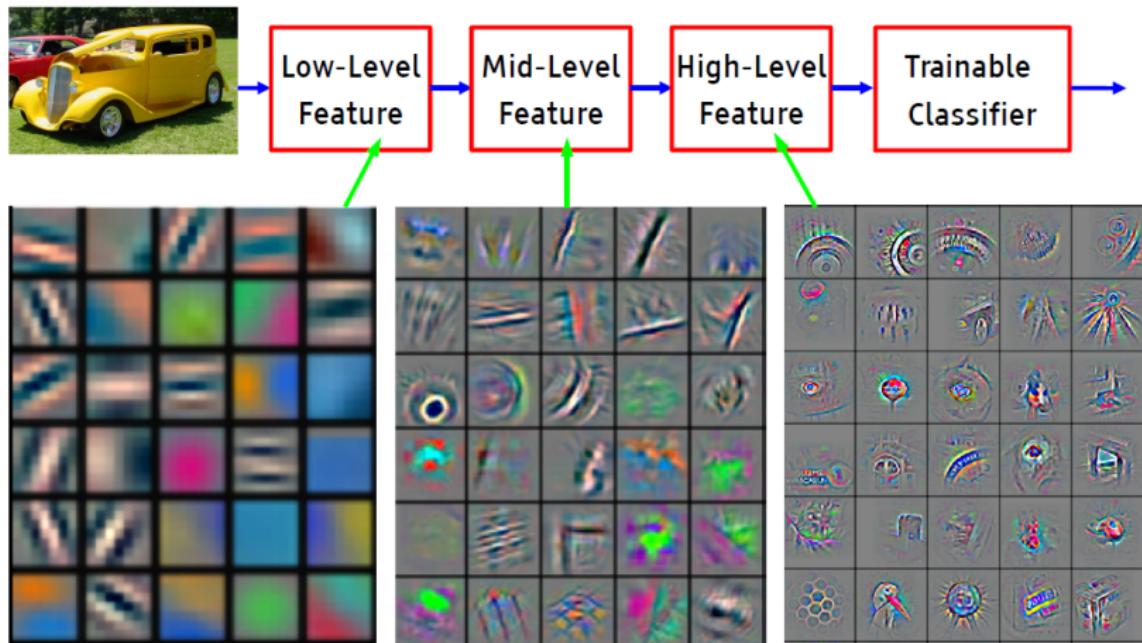
- Que son **profundos** en el número de capas (**redes convolucionales**)
- Que son **profundas** en el tiempo (**redes neuronales recurrentes**)
- Que aprenden **representaciones distribuidas** de las palabras (**word embeddings**)

Aprendizaje profundo

- El **aprendizaje profundo** (AP) permite que modelos computacionales compuestos de **múltiples capas** de procesamiento **aprendan representaciones** de datos con **múltiples niveles** de abstracción.
- El AP permite a la computadora construir **conceptos complejos** a partir de **conceptos más simples**



Aprendizaje profundo = Aprendizaje de representaciones jerárquicas (Ej: ConvNets)



Breve historia de las NN y el aprendizaje profundo

El área de NNs en general fue **referenciada** de diferentes maneras:

- **Cibernética**: 1940s-1960s
- **Conexionismo**: 1980s-1990s
- **Aprendizaje profundo**: 2006 a la fecha

Breve historia de las NN y el aprendizaje profundo

- **Primera ola** de las NN
 - Modelo de neurona de McCulloch y Pitts (1943)
 - **Perceptrón** (Rosenblatt, 1958). Primer modelo que podía aprender los pesos
 - El **adaptive linear element (ADALINE)** (Widroff y Hoff, 1960). Usa variante de SGD
- Limitaciones de modelos lineales (Minsky y Papert, 1969)
- **Segunda ola** de las NN
 - **Conexionismo** o procesamiento distribuido paralelo. (Rumelhart, 1986). Backpropagation.
 - Modelización de **secuencias** con NNs. Redes Long short-term memory (LSTM). 1997.
- Popularidad de otros modelos declinan la de NNs
- **Tercera ola** de las NN
 - Hinton (2006). Aprendizaje eficiente de Deep Belief Networks. Se populariza el término **deep learning**
 - Redes convolucionales, SotA en ImageNet (2012), Word2vec, Mikolov (2013)..... (muchos) otros

La Ambigüedad en el PLN

- El lenguaje natural se distingue de los lenguajes artificiales por su riqueza y flexibilidad.
 - Sin embargo, si bien es bueno para la comunicación humana introduce problemas como la ambigüedad.
 - En palabras de Sonia Vázquez:

“Lo que son ventajas para la comunicación humana se convierten en problemas a la hora de un tratamiento computacional, ya que implican conocimiento y procesos de razonamiento que son difíciles de formalizar.”
 - Entre esos problemas, la ambigüedad es uno de los principales en el PLN y se da a distintos niveles (palabras polisémicas, oraciones con ≠ interpretaciones), etc.

Algunos tipos de *ambigüedad*

- Ambigüedad **léxica**: una **palabra** $\Rightarrow \neq$ **categorías gramaticales**. Ejemplo: “**para**”.
- Ambigüedad **sintáctica** (o **estructural**): una **oración** $\Rightarrow \neq$ **interpretaciones**. Ejemplo: “Juan vio a su hermana con unos prismáticos”.
- Ambigüedad **semántica**: incluye
 - Ambigüedad debido a **palabras polisémicas**: Ejemplo: “**banco**”
 - Ambigüedad **referencial**: Ejemplo: “El jamón está en el armario. **Sácalo. Ciérralo**”.

Representación de *bolsa de palabras* ("Bag of Words" (BoW))

Documentos

- ① "pintaron el banco de la plaza"
- ② "si paso la prueba, iremos paso a paso"
- ③ "no me banco ir al banco a cobrar cheques"

Representación BoW

D/T	a	al	banco	cheques	cobrar	de	el	ir	iremos	la	me	no	paso	pintaron	plaza	prueba	si
d1	0	0	1	0	0	1	1	0	0	1	0	0	0	1	1	0	0
d2	1	0	0	0	0	0	0	0	1	1	0	0	3	0	0	1	1
d3	1	1	2	1	1	0	0	1	0	0	1	1	0	0	0	0	0

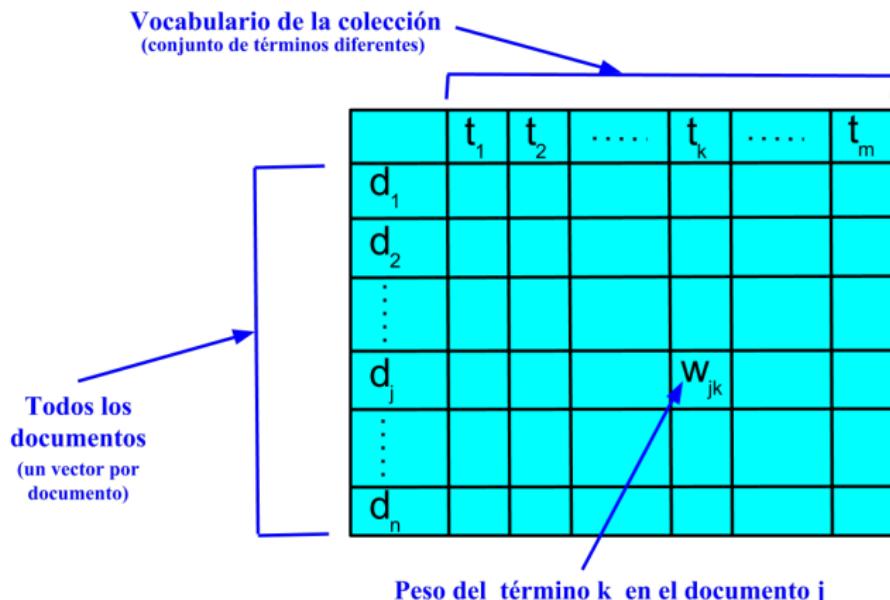
Representación BoW

- En BoW cada **documento** es representado como una **bolsa de palabras**
- Cada bolsa (multiset) es implementada como un **vector** con las **frecuencias** de ocurrencia en el documento de **cada palabra/término** del **vocabulario** de la colección de documentos.
- Así, las bolsas de todos los documentos de la colección constituyen una **matriz “documento-término”** X
- Con n **documentos** en la colección y m **términos** ocurriendo **en alguno** de esos documentos, tendremos una matriz X de dimensión $n \times m$

Representación BoW

- En IR, la **hipótesis de la bolsa de palabras** es que se puede estimar la relevancia de un documento a una consulta representando tanto los documentos como las consultas como **bolsa de palabras**.
- Esta hipótesis expresa la creencia que un vector fila en la matriz documento-término X captura (en alguna medida) un aspecto del **significado** del documento correspondiente: aquello de **lo que trata** el documento.
- Así, planteándolo en términos de la matriz documento-término X , si dos documentos tienen vectores **filas similares** en X , éstos tenderán a tener **significados similares**.

Representación vectorial de documentos: visión general



Consideraciones finales sobre BoW

Ideas subyacentes

- No se captura ningún tipo de información sobre el **orden** en que aparecen los términos/palabras
- BoW sólo mira a la **forma superficial** de las palabras, ignorando toda **información semántica** de las mismas

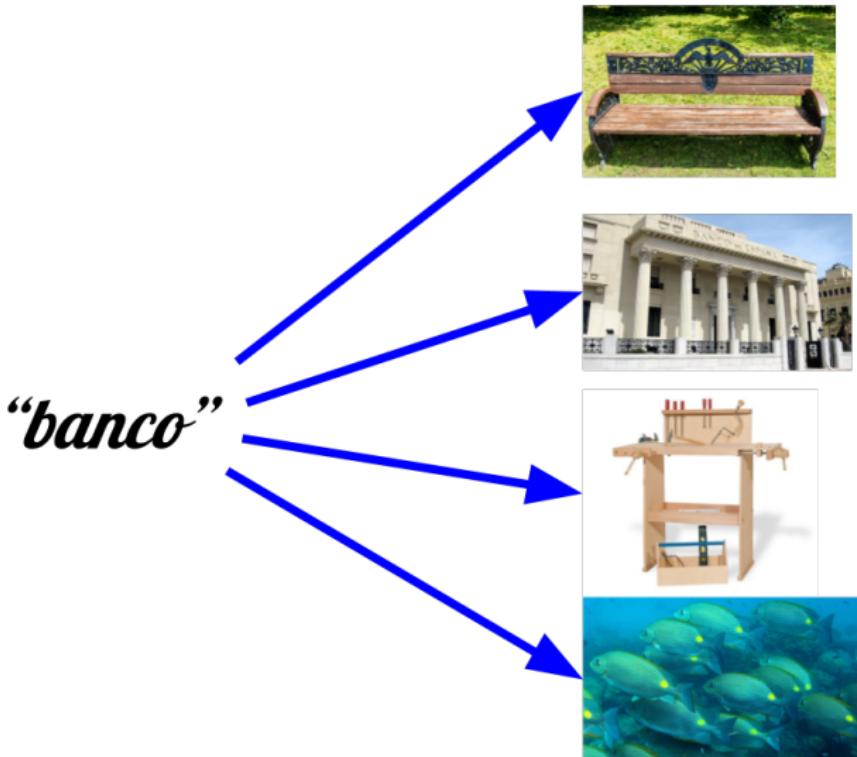
Ventajas

- Simplicidad.
- Eficiencia.

Desventajas

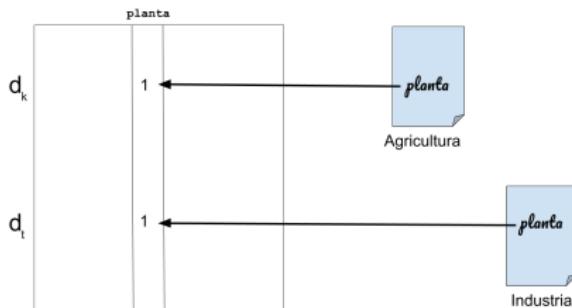
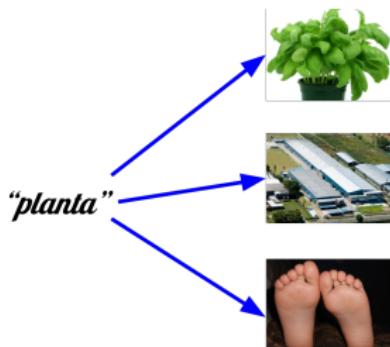
- BoW tiende a producir **representaciones muy dispersas** (“sparse”)
- Problemas “semánticos” con la **polisemia** y la **sinonimia**

Polisemia (y homonimia)

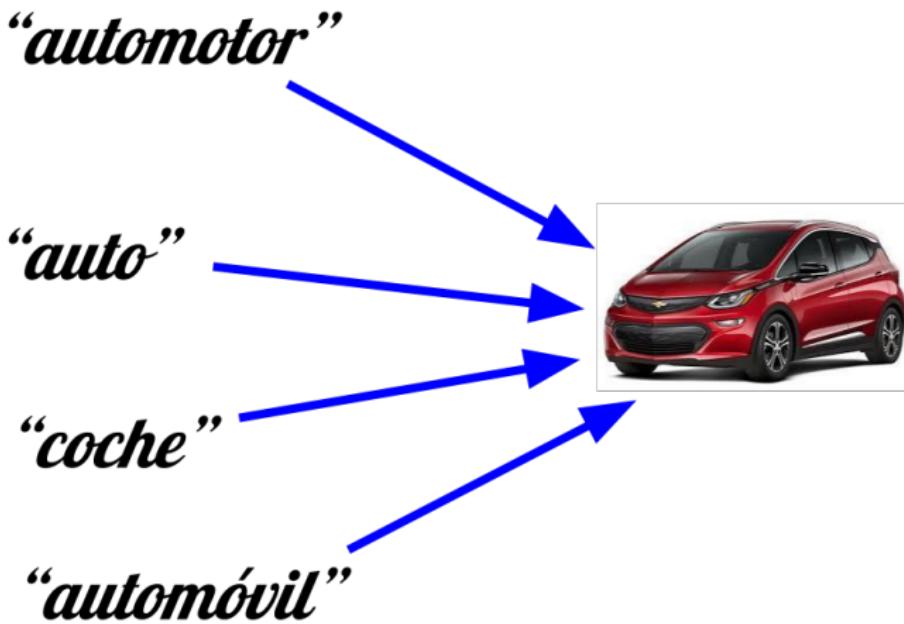


BoW y la polisemia

La **polisemia** introduce **ruido** en la **representación BoW**

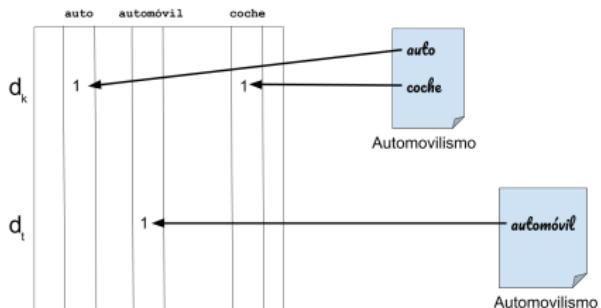
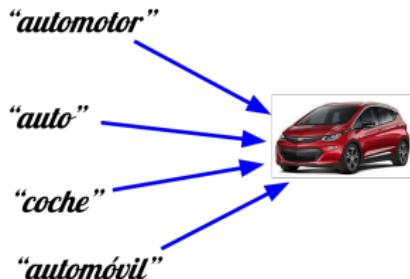


Sinonimia



BoW y la sinonimia

La **sinonimia divide** la evidencia en la **representación BoW**



Representación distribucional de términos (BoC)

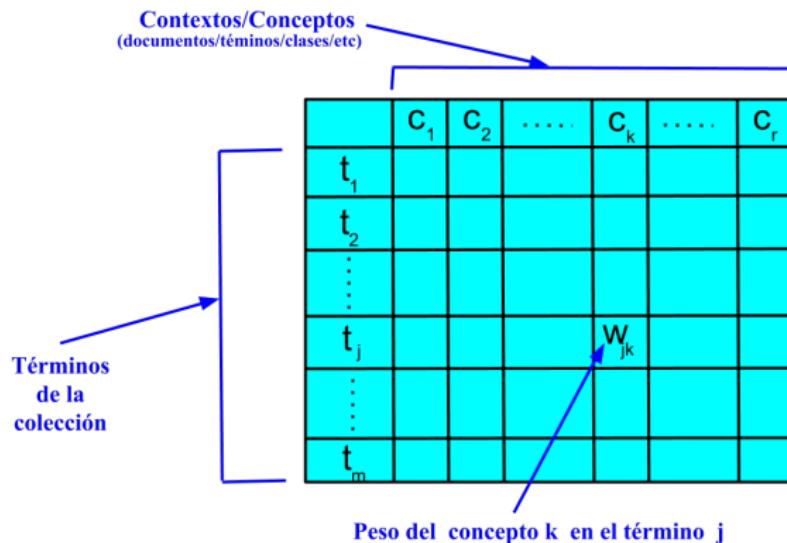
Representación distribucional de términos

- Las ideas surgen de las **dificultades** que se presentan para definir el **significado** de las palabras.
 - Circularidad de las definiciones de diccionario.
 - Dificultad para capturar otros tipos de **relaciones** entre palabras (más alla de las relaciones semanticas clásicas de WN) (**asociación** de palabras, **campos semánticos**, **significados afectivos/connotaciones**).
- Identificadas por filósofos como **Ludwig Wittgenstein**:
*... the **meaning** of a word is its **use** in the language*
- ... y lingüistas como **John R. Firth**:
*You shall **know** a word by the **company** it keeps!*

Representación distribucional de términos (BoC)

Representación distribucional de términos

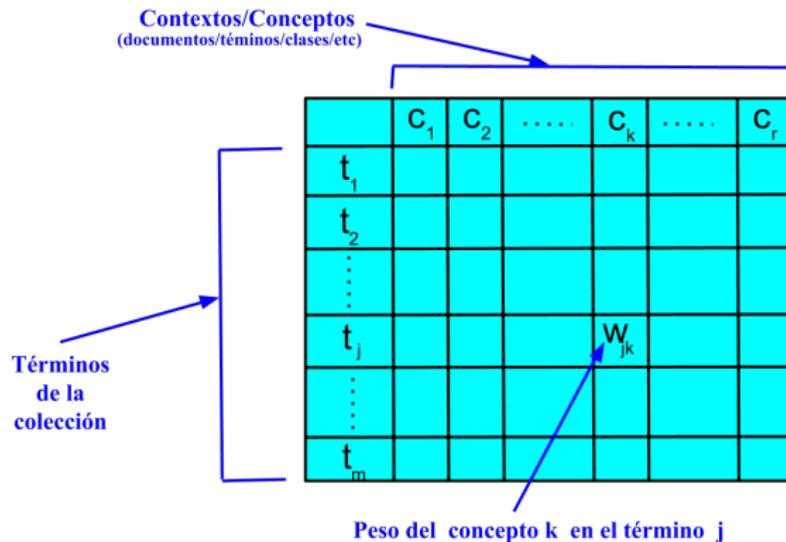
- Propuesto para abordar las **deficiencias** del modelo BoW
- El foco se pone en las **palabras** y los **contextos** en que éstas ocurren



Representación distribucional de términos (BoC)

Representación distribucional de términos

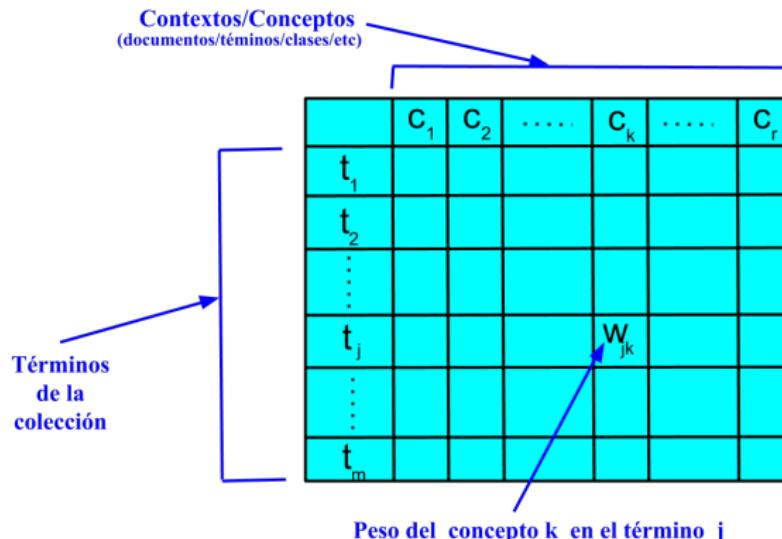
- Se cambia el foco en medir **similitud de palabras**
- **Hipótesis distribucional:** palabras que ocurren en contextos similares tienden a tener **significados similares.**



Representación distribucional de términos (BoC)

Representación distribucional de términos

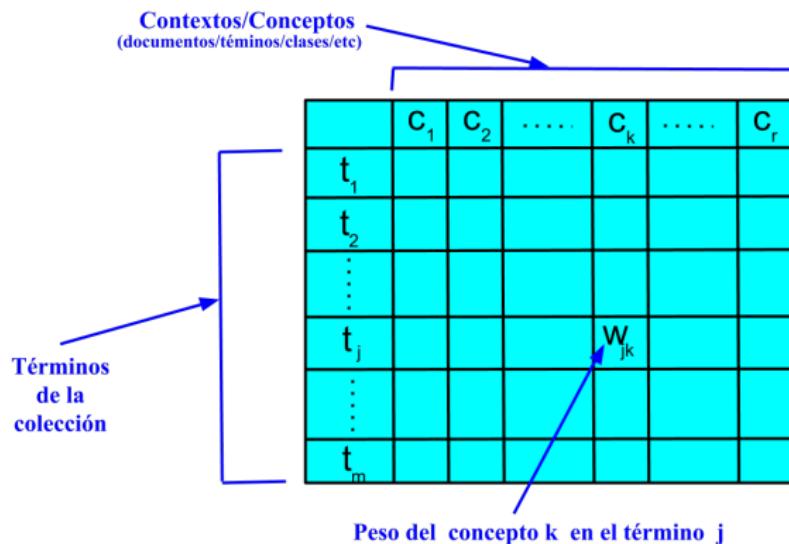
- Cada palabra es representada por un **vector**
- Cada elemento del vector se deriva de la **ocurrencia de la palabra en distintos contextos**: otras **palabras, frases, sentencias, párrafos, capítulos, documentos**, etc.



Representación distribucional de términos (BoC)

Representación distribucional de términos

- Cada **contexto** puede ser visualizado como un **concepto**
- Cada fila de la matriz palabra-concepto puede ser visualizada entonces como una **bolsa de conceptos (BOC)**



Algunas representaciones distribucionales conocidas

Existen distintas variantes de las representaciones distribucionales.

- Document occurrence representation (**DOR**)
- Term co-occurrence representation (**TCOR**)
- Concise semantic analysis (**CSA**)

Representación distribucional de términos (BoC)

Representaciones ralas

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**)

Representación distribucional de términos (BoC)

Variantes de vectores densos

El Análisis de Semántica Latente **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]



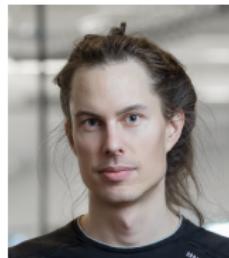
Disponible en:

<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/>

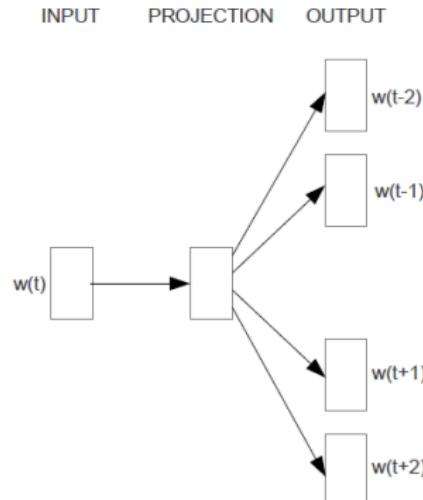
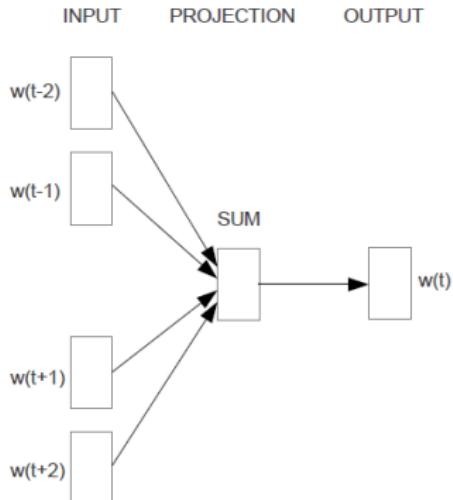
Embeddings aprendidos de la predicción

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

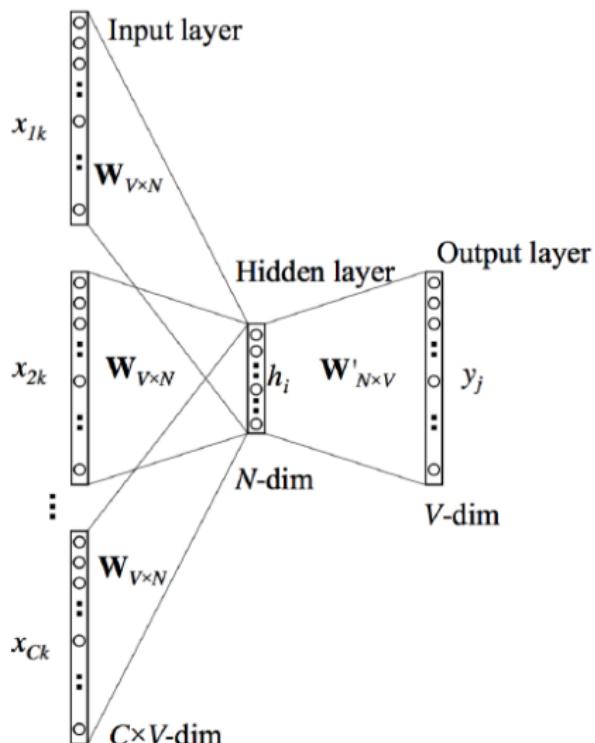
Embeddings aprendidos de la predicción

Enfoques en Word2vec



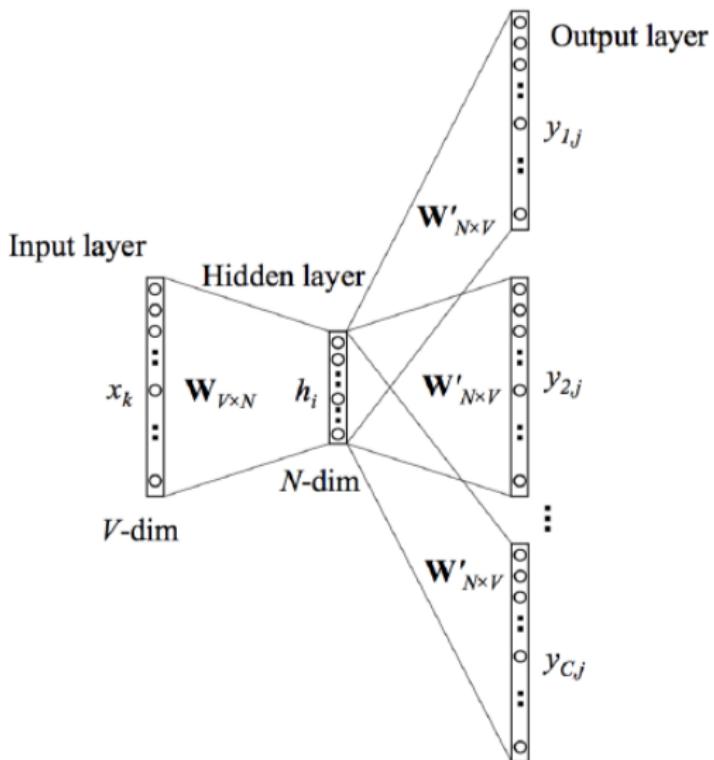
Embeddings aprendidos de la predicción

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**

Embeddings aprendidos de la predicción

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

¿Como 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

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

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

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.

Embeddings aprendidos de la predicción

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.

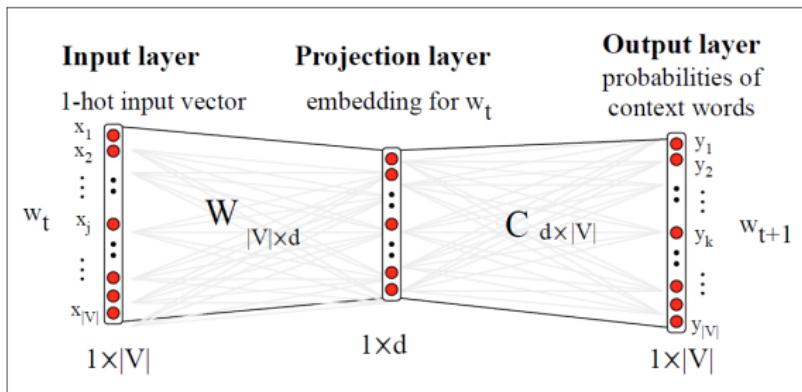
 $w_0 \ w_1$ w_j $w_{|V|}$

0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0
---	---	---	---	---	-----	---	---	---	---	---	---	---	---	---	---	---	-----	---	---	---	---

Embeddings aprendidos de la predicción

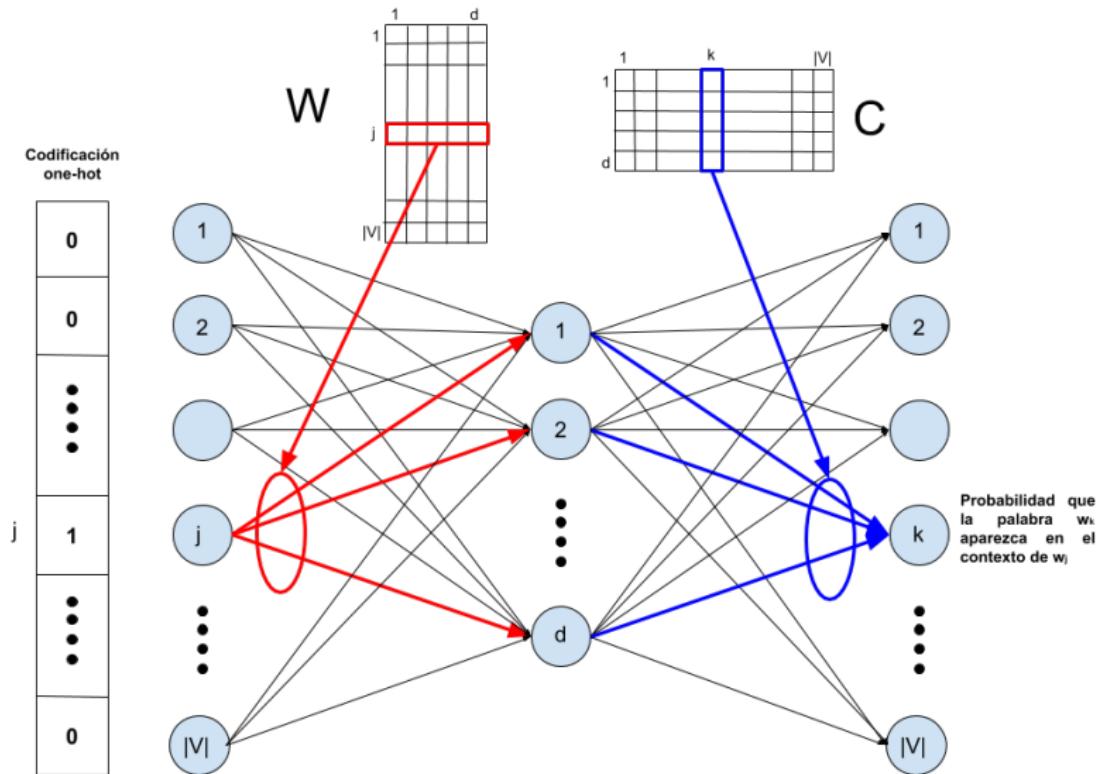
Esquema general de la arquitectura de aprendizaje

- 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.



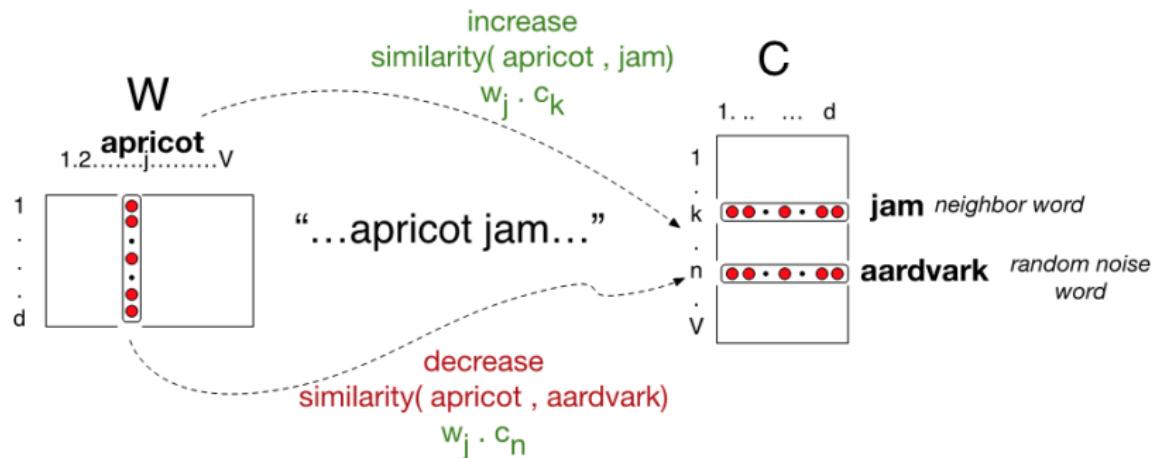
Embeddings aprendidos de la predicción

Cómo se relacionan W y C



Embeddings aprendidos de la predicción

Cómo se relacionan W y C



Embeddings aprendidos de la predicción

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

Embeddings aprendidos de la predicción

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*)

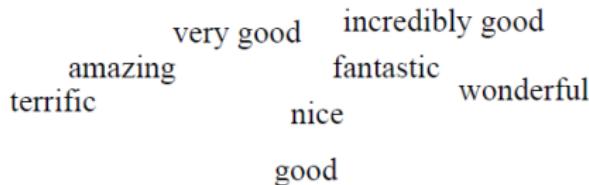
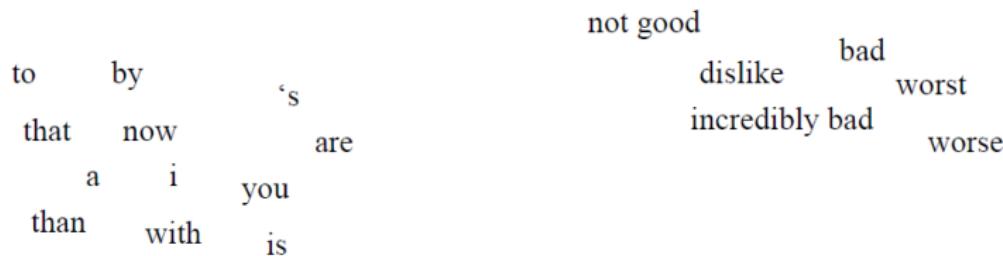
Correlación con ratings de similitud asignados por humanos

- 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

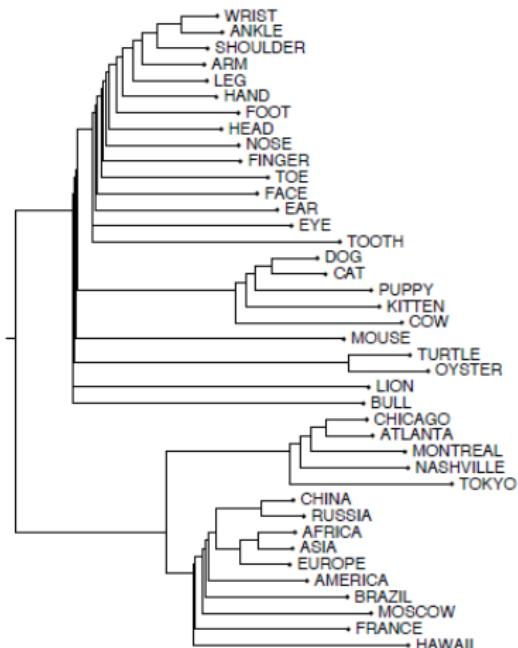
Visualización de embeddings

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:

