

Clase 5: Enfoques Neuronales al Análisis de Textos (Parte A)

Enfoques Clásicos y Neuronales a la Minería de Texto

Marcelo Errecalde^{1,2}

¹Universidad Nacional de San Luis, Argentina 

²Universidad Nacional de la Patagonia Austral, Argentina 



Resumen

1 Word Embeddings

2 Modelos de Lenguaje y Redes Neuronales Recurrentes

- Modelos de Lenguajes
- Modelo de Lenguaje de n -gramas
- Modelo de Lenguaje Neuronal (con ventana fija)
- Redes Neuronales Recurrentes

Word Embeddings: repaso

Word Embeddings

También llamados **word vectors**, son representaciones de **palabras** que asignan un **vector numérico** a **cada palabra**. Son una forma de **representación distribuida** que usualmente captura distinta información sintáctica y semántica de las palabras.

Variantes de representaciones de palabras como vectores

- Codificación **one-hot (one-hot encoding)**
- Embeddings basados en técnicas de **conteo**
- Embeddings basados en **predicción**

One-hot encoding

- A cada palabra del **vocabulario** \mathcal{V} (palabras únicas) se le asigna una **posición** del vector

One-hot encoding

- A cada palabra del **vocabulario** \mathcal{V} (palabras únicas) se le asigna una **posición** del vector
 - La representación de una palabra es un vector de tamaño $|\mathcal{V}|$ con:
 - 1 todos los elementos en **0 (cero)** y
 - 2 un **1 (uno)** en la **posición** que corresponde a la **palabra**

One-hot encoding

Ejemplo: “The cat sat on the mat”

Vocabulario $\mathcal{V} = \{ \text{cat, mat, on, sat, the} \}$, $|\mathcal{V}| = 5$

One-hot encoding

	cat	mat	on	sat	the
the =>	0	0	0	0	1
cat =>	1	0	0	0	0
sat =>	0	0	0	1	0

One-hot encoding

- **Ventaja:** enfoque muy **sencillo** de implementar e intuitivo.

One-hot encoding

- **Ventaja:** enfoque muy **sencillo** de implementar e intuitivo.
- **Desventajas:**
 - ➊ **Alta dimensionalidad** de los vectores (en el término de las decenas/centenas de miles)

One-hot encoding

- **Ventaja:** enfoque muy **sencillo** de implementar e intuitivo.
- **Desventajas:**
 - 1 Alta dimensionalidad de los vectores (en el término de las decenas/centenas de miles)
 - 2 representación usualmente **muy dispersa** (casi todos los elementos son 0s)

One-hot encoding

- **Ventaja:** enfoque muy **sencillo** de implementar e intuitivo.
- **Desventajas:**
 - ① Alta **dimensionalidad** de los vectores (en el término de las decenas/centenas de miles)
 - ② representación usualmente **muy dispersa** (casi todos los elementos son 0s)
 - ③ No hay noción **natural de similitud**. Ejemplo:
motel = [0 0 0 0 0 0 0 0 0 1 0 0 0 0]
hotel = [0 0 0 0 0 0 1 0 0 0 0 0 0]
si bien similares, **no son cercanos** en la representación (en efecto son ortogonales)

Word embeddings

- Los word embeddings proveen una **representación densa**, de **baja dimensionalidad** en la cual las palabras similares tienen una **codificación similar**.

Word embeddings

- Los word embeddings proveen una **representación densa**, de **baja dimensionalidad** en la cual las palabras similares tienen una **codificación similar**.
- Un embedding es un vector denso de **valores reales**, siendo la longitud del vector un parámetro a especificar.

A 4-dimensional embedding

cat	=>	1.2	-0.1	4.3	3.2
mat	=>	0.4	2.5	-0.9	0.5
on	=>	2.1	0.3	0.1	0.4

...

...

Word embeddings

- Los valores de los embeddings corresponden a **parámetros entrenables** (pesos aprendidos por el modelo durante el entrenamiento).

Word embeddings

- Los valores de los embeddings corresponden a **parámetros entrenables** (pesos aprendidos por el modelo durante el entrenamiento).
- La dimensionalidad varía desde 8 (pequeños datasets) hasta 1024 dimensiones (grandes datasets).
- Embeddings de **dimensionalidad más alta** pueden capturar **relaciones más finas** entre palabras (toma más datos para aprender).
- Uno de los algoritmos más populares hasta la fecha para aprender embeddings es **Word2vec**

Representando palabras por su contexto

- **Semántica distribucional:** el **significado** de una **palabra** está dado por las **palabras** que aparecen fecuentemente **cercanas a ella.**

Representando palabras por su contexto

- **Semántica distribucional:** el **significado** de una **palabra** está dado por las **palabras** que aparecen **fecuentemente cercanas a ella**.
- Resumida en la frase: “**You shall know a word by the company it keeps**” (J. R. Firth, 1957)

Representando palabras por su contexto

- **Semántica distribucional:** el **significado** de una **palabra** está dado por las **palabras** que aparecen frecuentemente **cercanas a ella**.
- Resumida en la frase: “**You shall know a word by the company it keeps**” (J. R. Firth, 1957)
- Cuando una palabra w aparece en un texto, su **contexto** es el contexto de palabras que aparecen **cerca** (dentro de una ventana de tamaño fijo)

Representando palabras por su contexto

- **Semántica distribucional:** el **significado** de una **palabra** está dado por las **palabras** que aparecen **fecuentemente cercanas a ella**.
- Resumida en la frase: “**You shall know a word by the company it keeps**” (J. R. Firth, 1957)
- Cuando una palabra w aparece en un texto, su **contexto** es el contexto de palabras que aparecen **cerca** (dentro de una ventana de tamaño fijo)
- **Idea:** usar los distintos contextos de w para construir una representación de w

Representando palabras por su contexto

Ejemplo: desconocemos qué significa **ongchoi**, pero la vemos en **contextos** como:

“El **ongchoi** es **delicioso salteado con ajo.**”

“El **ongchoi** es excelente sobre el **arroz.**”

“... las hojas de **ongchoi** con **salsas saladas ...**”

Representando palabras por su contexto

Ejemplo: desconocemos qué significa **ongchoi**, pero la vemos en **contextos** como:

“El **ongchoi** es **delicioso salteado con ajo.**”

“El **ongchoi** es excelente sobre el **arroz.**”

“... las **hojas** de **ongchoi** con **salsas saladas ...**”

muchas de estas palabras de contexto ocurre en contextos como:

“... las **espinacas** **salteadas** con el **ajo** sobre el **arroz ...”**

“... los **tallos** y las **hojas** de **acelga** son **deliciosos ...”**

“... las **hojas** del **repollo** y otras verduras de hoja”

Representando palabras por su contexto

- Las palabras de contexto de **ongchoi** (**delicioso, salteado, ajo, arroz, hojas**, etc), son similares a las de **espinaca, acelga, repollo**.

Representando palabras por su contexto

- Las palabras de contexto de **ongchoi** (**delicioso, salteado, ajo, arroz, hojas**, etc), son similares a las de **espinaca, acelga, repollo**.
- Se podría decir entonces, que es probable que el **ongchoi** sea también algun tipo de *verdura de hoja*.

Representando palabras por su contexto

- Las palabras de contexto de **ongchoi** (**delicioso**, **salteado**, **ajo**, **arroz**, **hojas**, etc), son similares a las de **espinaca**, **acelga**, **repollo**.
- Se podría decir entonces, que es probable que el **ongchoi** sea también algun tipo de *verdura de hoja*.
- **Idea:** computacionalmente, contando las palabras del contexto de **ongchoi** y viendo que también ocurren en los contextos de **espinaca**, **acelga** y **repollo**, nos ayuda a identificar **similitudes** entre las mismas.

Representando palabras por su contexto

- Las palabras de contexto de **ongchoi** (**delicioso**, **salteado**, **ajo**, **arroz**, **hojas**, etc), son similares a las de **espinaca**, **acelga**, **repollo**.
- Se podría decir entonces, que es probable que el **ongchoi** sea también algun tipo de *verdura de hoja*.
- **Idea:** computacionalmente, contando las palabras del contexto de **ongchoi** y viendo que también ocurren en los contextos de **espinaca**, **acelga** y **repollo**, nos ayuda a identificar **similitudes** entre las mismas.
- Word2vec trabaja sobre una idea similar estimando la **probabilidad** que, dada una palabra, ocurran distintas palabras en su contexto.

Word vectors/embeddings

Se aplican las mismas ideas que vimos recién: construir un vector denso para cada palabra, elegido para que **sea similar** a los vectores de palabras que aparecen en contextos similares

Word2vec

word2vec [Mikolov et. al]



Disponible en:

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

Word2vec

word2vec [Mikolov et. al]



Disponible en:

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

Word2vec

- El método **más popular** de **embedding** en la actualidad.

Word2vec

- El método **más popular** de **embedding** en la actualidad.
- A diferencia de los enfoques **distribucionales**, la idea es **predecir** más que **contar**

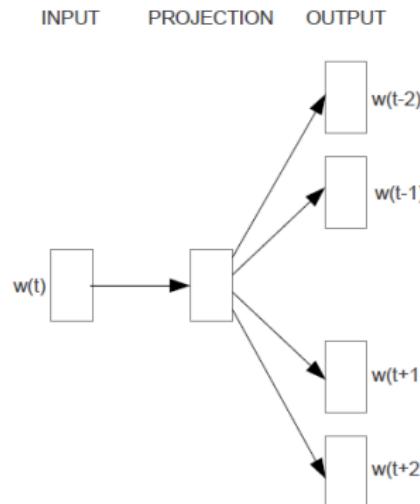
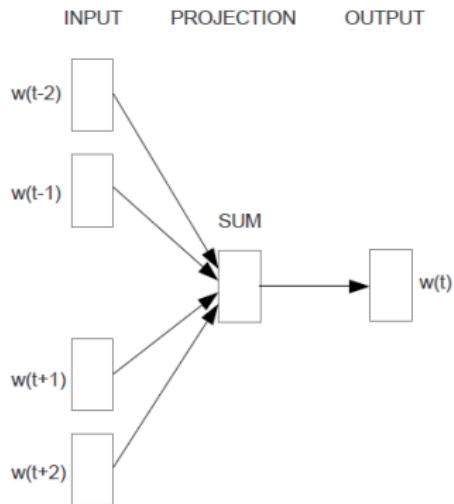
Word2vec

- 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.

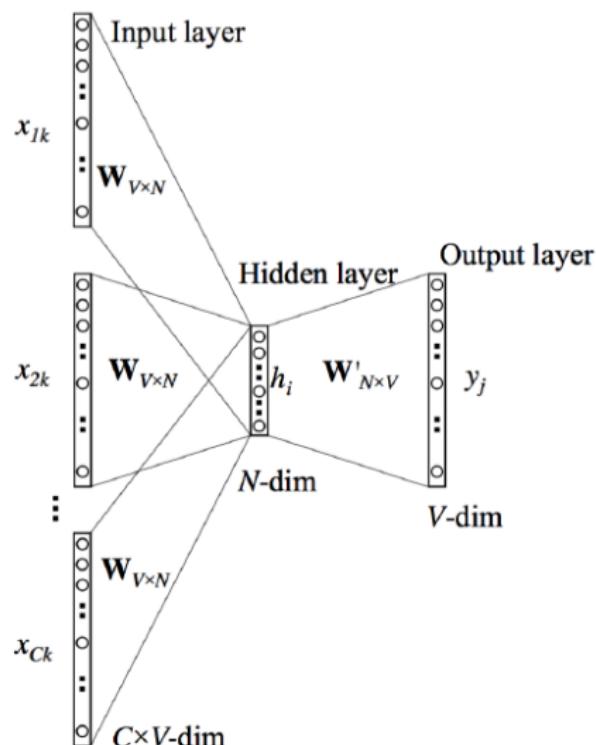
Word2vec

- 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

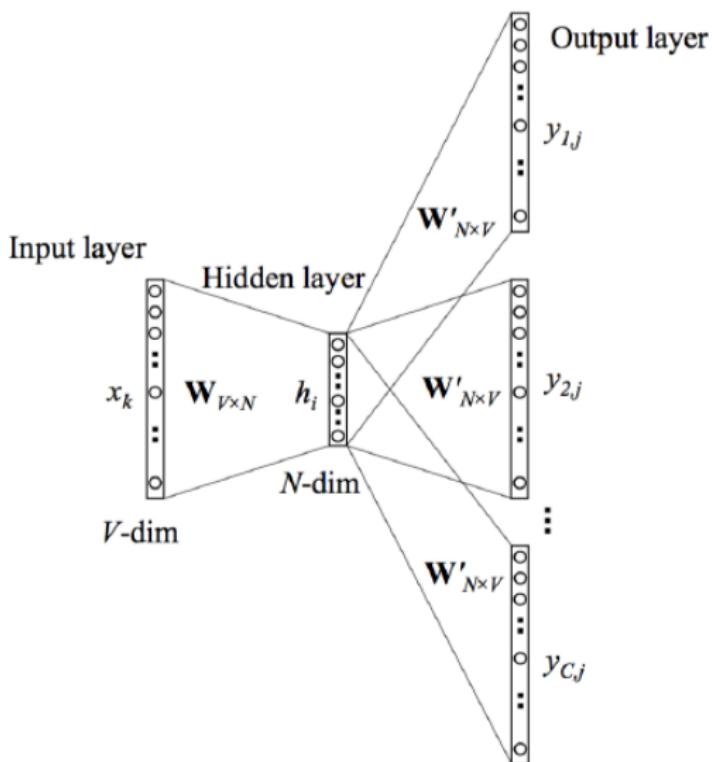
Enfoques en Word2vec



Detalles de CBOW



Detalles de Skip-gram



Word2vec

- En lugar de **contar** cuan a menudo cada palabra w ocurre **cerca** de “apricot”

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”?

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

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

Word2vec

- Idea (**brillante**) tomada de **modelos** de lenguajes neuronales
 - usar el texto bajo análisis como datos (**supervisados**) implícitos

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**

Entrenamiento de *Skip-Gram*

Sentencia de entrenamiento:

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

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

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

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

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

Entrenamiento de *Skip-Gram*

Sentencia de entrenamiento:

... lemon, a tablespoon of apricot jam a pinch ...

[c1 c2 obj c3 c4]

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

Esquema general de la arquitectura de aprendizaje

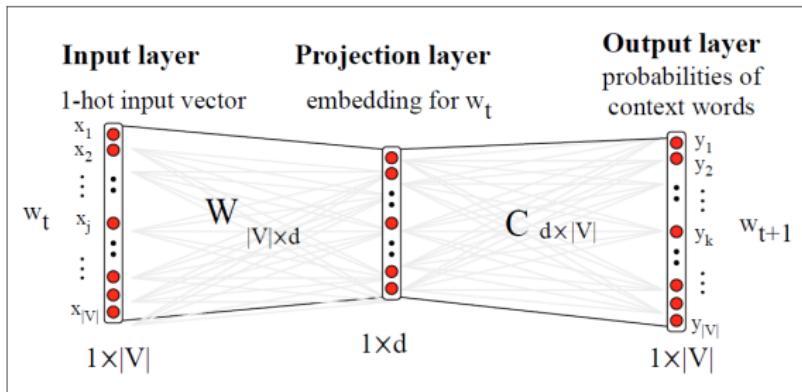
- Entrenamiento x **descenso del gradiente**.
- Codificación de las palabras **one-hot**.
- Aprende 2 matrices de embeddings separadas (**W** y **C**).
- Se usa **W** (“tirando” **C**) o se combinan de alguna manera.

One-hot encoding

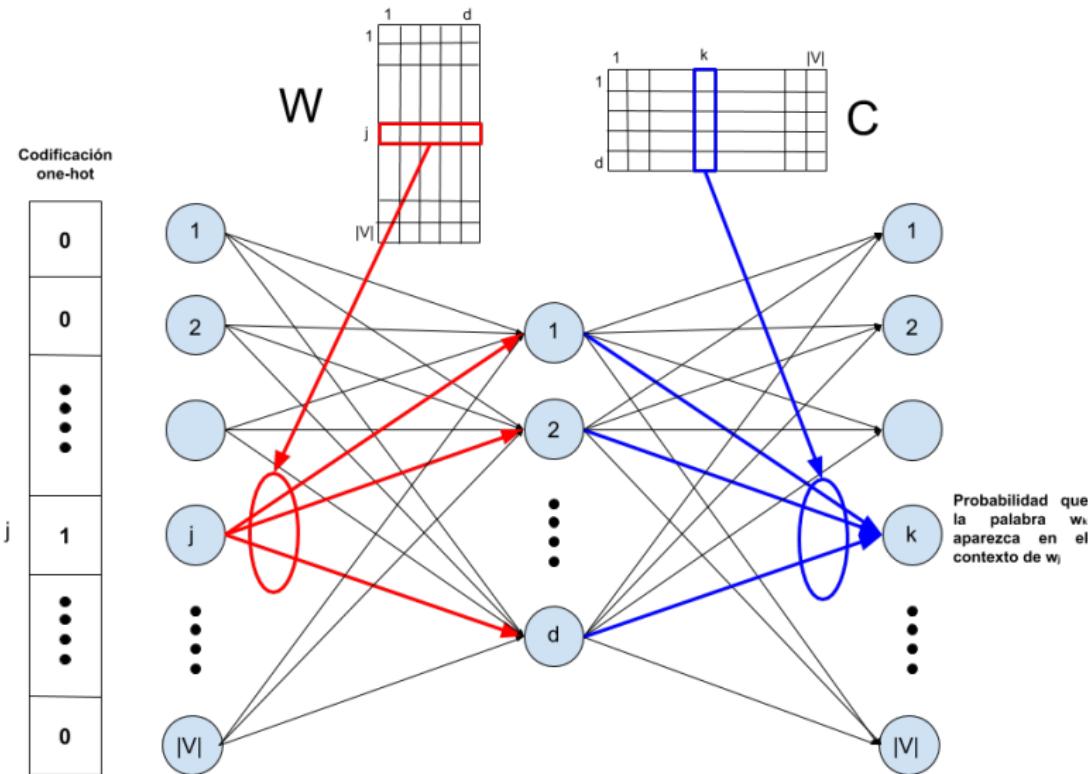
	ca†	mat	on	sat	the
the =>	0	0	0	0	1
cat =>	1	0	0	0	0
sat =>	0	0	0	1	0
...

Esquema general de la arquitectura de aprendizaje

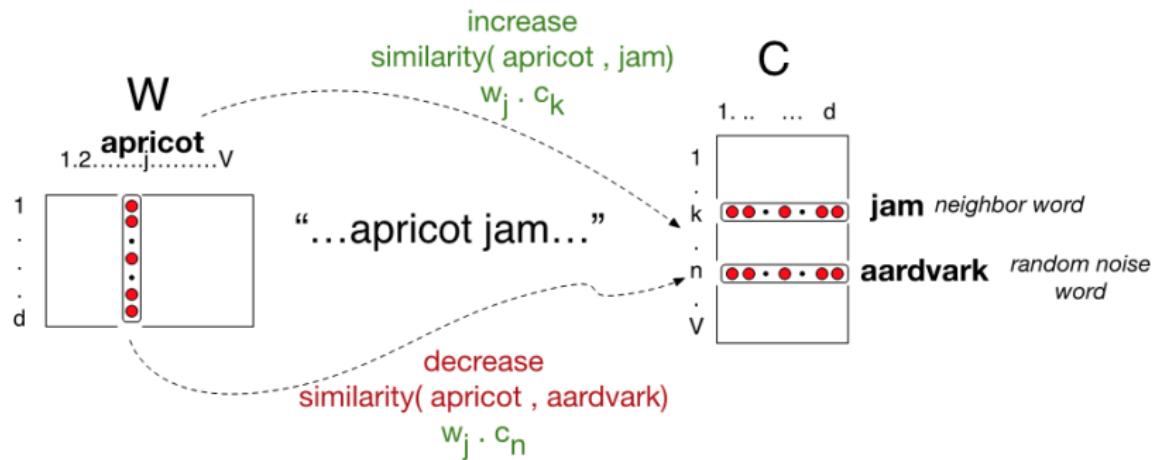
- Entrenamiento x **descenso del gradiente**.
- Codificación de las palabras **one-hot**.
- Aprende 2 matrices de embeddings separadas (**W** y **C**).
- Se usa **W** (“tirando” **C**) o se combinan de alguna manera.



Cómo se relacionan W y C



Cómo se relacionan W y C [Jurafsky]



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

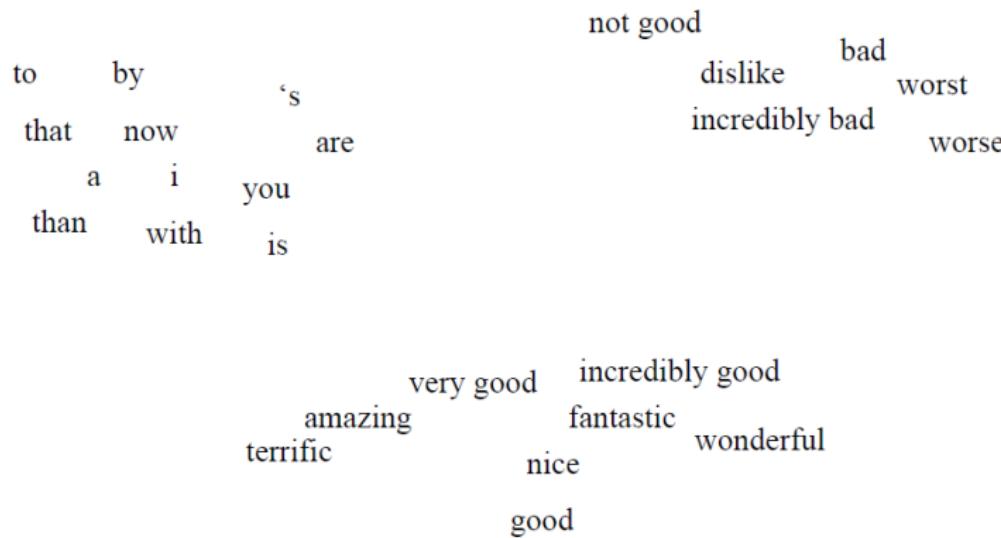
Evaluación intrínseca (*similitud* entre palabras)

Correlación con ratings de similitud asignados por humanos

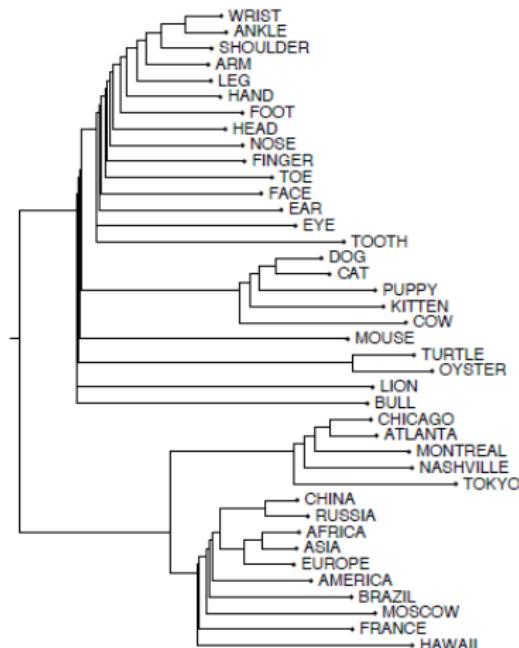
- WordSim-353 (Finkelstein et al., 2002)
- SimLex-999 (Hill et al., 2015)
- Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset: Levied is closest in meaning to: imposed, believed, requested, correlated

Visualización de embeddings

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



Visualización de embeddings (II)



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.

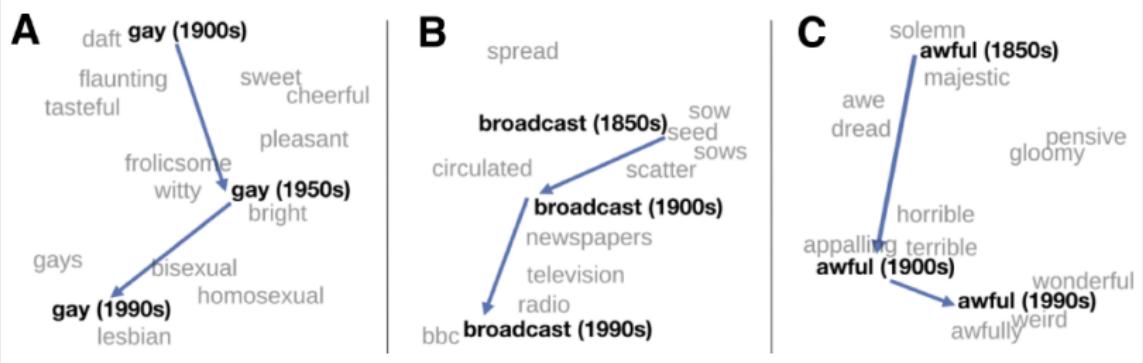
Propiedades (semánticas) de los embeddings

Habilidad para capturar **significados relacionales!!!!:**

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

Propiedades (semánticas) de los embeddings (II)

Cambios en el significado (histórico) de las palabras:



Modelo de Lenguaje y Redes Neuronales

- Las **representaciones BoW** son usualmente criticadas por perder información del **orden** en que las palabras aparecen en el texto.

Modelo de Lenguaje y Redes Neuronales

- Las **representaciones BoW** son usualmente criticadas por perder información del **orden** en que las palabras aparecen en el texto.
- Sin embargo, en muchos problemas de PLN como **WSD**, **POS** tagging, **traducción automática** y **question answering** mantener información sobre la **secuencia** (orden) en que fueron apareciendo las palabras suele ser fundamental.

Modelo de Lenguaje y Redes Neuronales

- Las **representaciones BoW** son usualmente criticadas por perder información del **orden** en que las palabras aparecen en el texto.
- Sin embargo, en muchos problemas de PLN como **WSD**, **POS** tagging, **traducción automática** y **question answering** mantener información sobre la **secuencia** (orden) en que fueron apareciendo las palabras suele ser fundamental.
- Surge así la idea de trabajar con **modelos de lenguajes** (ML) y en el uso de **redes neuronales** (RN) para implementar estos modelos

Modelo de Lenguaje y Redes Neuronales

- Las **representaciones BoW** son usualmente criticadas por perder información del **orden** en que las palabras aparecen en el texto.
- Sin embargo, en muchos problemas de PLN como **WSD**, **POS** tagging, **traducción automática** y **question answering** mantener información sobre la **secuencia** (orden) en que fueron apareciendo las palabras suele ser fundamental.
- Surge así la idea de trabajar con **modelos de lenguajes** (ML) y en el uso de **redes neuronales** (RN) para implementar estos modelos
- En particular, los ML motivan naturalmente el uso de una nueva familia de RNs, las **redes neuronales recurrentes** (RNN)
- Comenzaremos viendo qué se entiende por **modelización de lenguaje**

Modelos de Lenguajes

Modelización de Lenguaje

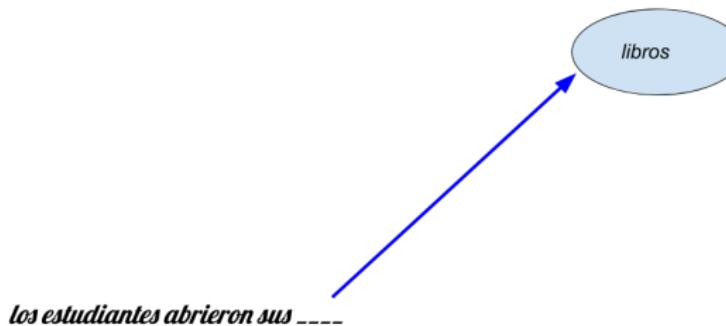
Tarea de predecir **cual es la palabra** que viene **a continuación**.

los estudiantes abrieron sus _____

Modelos de Lenguajes

Modelización de Lenguaje

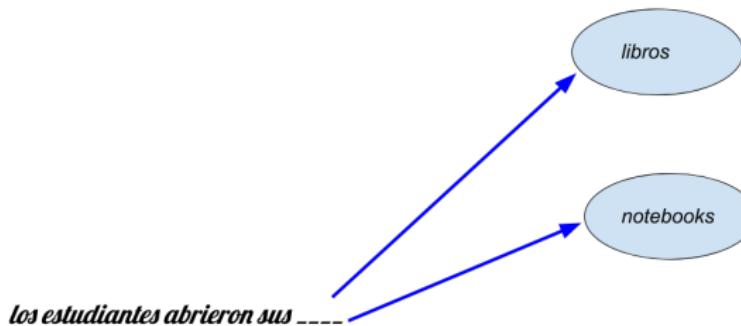
Tarea de predecir **cual es la palabra** que viene **a continuación**.



Modelos de Lenguajes

Modelización de Lenguaje

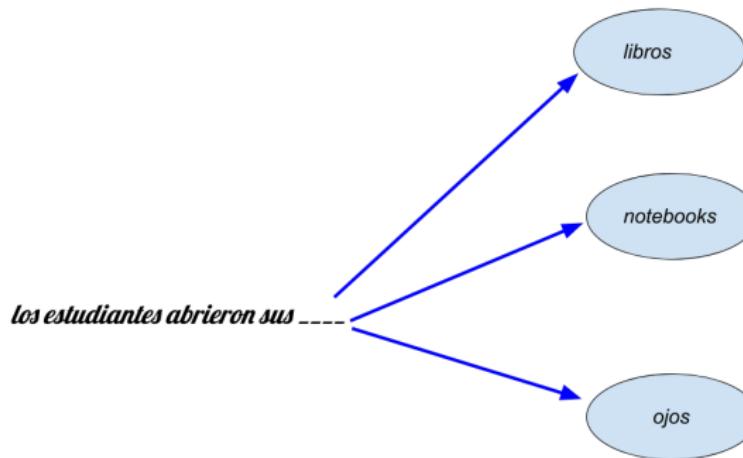
Tarea de predecir **cual es la palabra** que viene **a continuación**.



Modelos de Lenguajes

Modelización de Lenguaje

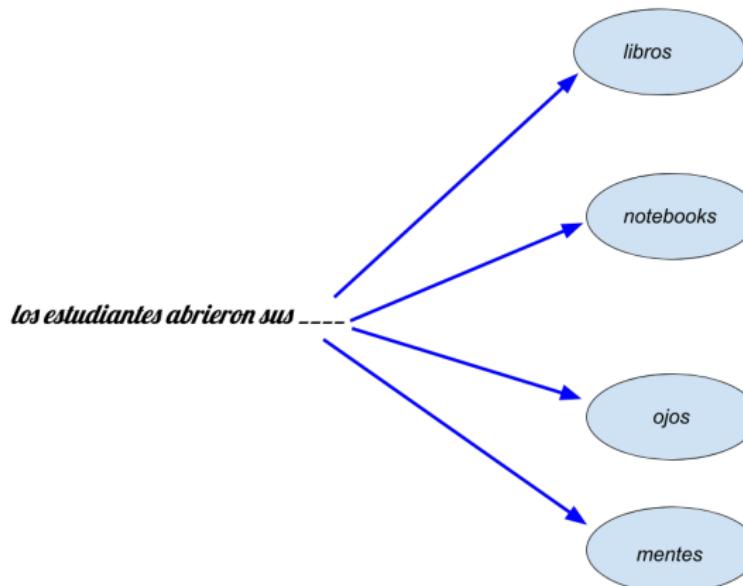
Tarea de predecir **cual es la palabra** que viene **a continuación**.



Modelos de Lenguajes

Modelización de Lenguaje

Tarea de predecir **cual es la palabra** que viene **a continuación**.



más formalmente ...

Dada una secuencia de palabras $x^{(1)}, x^{(2)}, \dots, x^{(t)}$, determinar la distribución de probabilidad de la siguiente palabra $x^{(t+1)}$:

más formalmente ...

Dada una secuencia de palabras $x^{(1)}, x^{(2)}, \dots, x^{(t)}$, determinar la distribución de probabilidad de la siguiente palabra $x^{(t+1)}$:

$$P(x^{(t+1)} | x^{(1)}, \dots, x^{(t)})$$

donde $x^{(t+1)}$ puede ser cualquier palabra en el vocabulario
 $V = \{w_1, \dots, w_{|V|}\}$

más formalmente ...

Dada una secuencia de palabras $x^{(1)}, x^{(2)}, \dots, x^{(t)}$, determinar la distribución de probabilidad de la siguiente palabra $x^{(t+1)}$:

$$P(x^{(t+1)} | x^{(1)}, \dots, x^{(t)})$$

donde $x^{(t+1)}$ puede ser cualquier palabra en el vocabulario
 $V = \{w_1, \dots, w_{|V|}\}$

Un sistema que lleva a cabo ésto es un **modelo de lenguaje**

Modelos de Lenguajes

Tarea relacionada ...

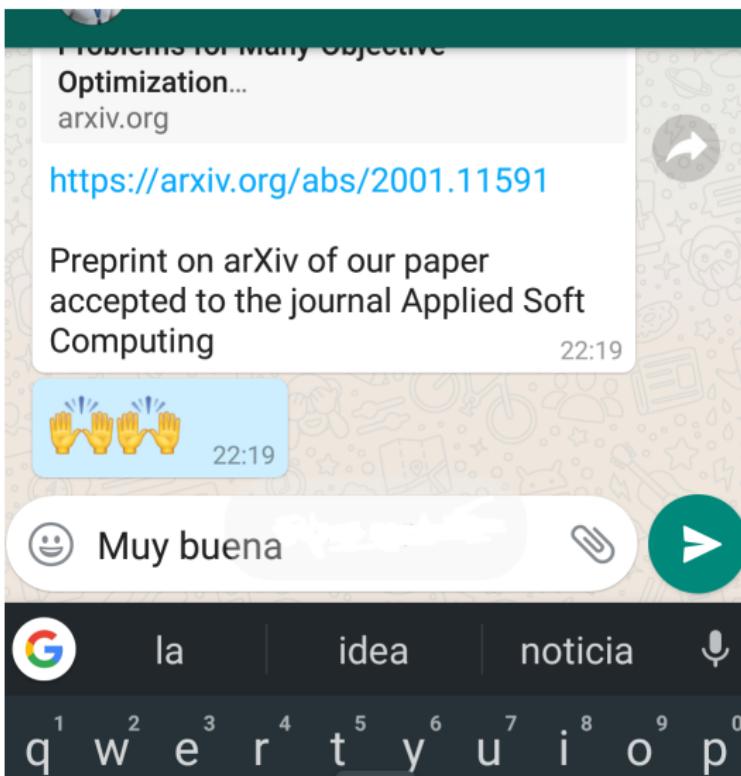
Computar la **probabilidad** de que ocurra una **sentencia** (secuencia de palabras):

$$P(W) = P(x^{(1)}, x^{(2)}, \dots, x^{(n)})$$

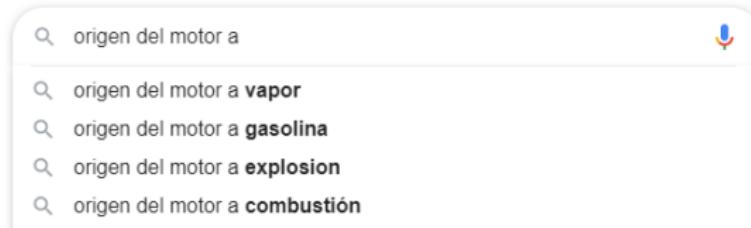
Cualquier modelo que compute $P(W)$ o $P(x^{(t+1)}|x^{(1)}, \dots, x^{(t)})$ es un **modelo de lenguaje**

Modelos de Lenguajes

... nuestro uso diario de modelos de lenguajes ...



... nuestro uso diario de modelos de lenguajes ...



Algunos enfoques usados para modelización de lenguajes

- Modelo de Lenguaje de *n-gramas*
- Modelo de Lenguaje Neuronal (con ventana fija)
- Modelo de Lenguaje basado en RNNs

Modelo de Lenguaje de n -gramas

Modelo de Lenguaje de n -gramas

Idea: utilizar la **regla de la cadena** para calcular $P(W)$:

$$P(W) = P(x^{(1)}, x^{(2)}, \dots, x^{(n)}) = \prod_i P(x^{(i)} | x^{(1)}, x^{(2)}, \dots, x^{(i-1)})$$

Modelo de Lenguaje de n -gramas

Idea: utilizar la **regla de la cadena** para calcular $P(W)$:

$$P(W) = P(x^{(1)}, x^{(2)}, \dots, x^{(n)}) = \prod_i P(x^{(i)} | x^{(1)}, x^{(2)}, \dots, x^{(i-1)})$$

Ejemplo:

$$\begin{aligned} P(\text{"los estudiantes abrieron sus libros"}) &= P(\text{los}) \times \\ P(\text{estudiantes}|\text{los}) \times P(\text{abrieron}|\text{los estudiantes}) \times \\ P(\text{sus}|\text{los estudiantes abrieron}) \times \\ P(\text{libros}|\text{los estudiantes abrieron sus}) \end{aligned}$$

Podemos sólo **contar y dividir**?

$$P(\text{libros}|\text{los estudiantes abrieron sus}) = \frac{\text{cuenta}(\text{los estudiantes abrieron sus libros})}{\text{cuenta}(\text{los estudiantes abrieron sus})}$$

Modelo de Lenguaje de *n*-gramas

Modelo de Lenguaje de *n*-gramas

Problemas:

- Demasiadas sentencias posibles !!
- Nunca veremos suficientes datos para siempre hacer estimaciones confiables

Alternativa:

- Usar un supuesto “simplificante” (propiedad **Markov**)

$$P(\text{libros}|\text{los estudiantes abrieron sus}) \approx P(\text{libros}|\text{sus})$$

- o bien ...

$$P(\text{libros}|\text{los estudiantes abrieron sus}) \approx P(\text{libros}|\text{abrieron sus})$$

El supuesto de Markov

La probabilidad de ocurrencia de una palabra **sólo** es afectada por las k palabras previas

$$P(W) = P(x^{(1)}, x^{(2)}, \dots, x^{(n)}) = \prod_i P(x^{(i)} | x^{(i-k)}, x^{(2)}, \dots, x^{(i-1)})$$

es decir, aproximamos cada componente en el producto mediante:

$$P(x^{(i)} | x^{(1)}, x^{(2)}, \dots, x^{(i-1)}) \approx P(x^{(i)} | x^{(i-k)}, x^{(2)}, \dots, x^{(i-1)})$$

- Modelo de **uni-gramas**, el más simple:

$$P(x^{(1)}, x^{(2)}, \dots, x^{(n)}) = \prod_i P(x^{(i)})$$

- Modelo de **bi-gramas**:

$$P(x^{(1)}, x^{(2)}, \dots, x^{(n)}) = \prod_i P(x^{(i)} | x^{(i-1)})$$

Modelo de Lenguaje de *n*-gramas

- Podemos extender a tri-gramas, 4-gramas y 5-gramas

Modelo de Lenguaje de *n*-gramas

- Podemos extender a tri-gramas, 4-gramas y 5-gramas
- Pero en general es un **modelo insuficiente** del lenguaje
(lenguaje tiene **dependencias de mucha distancia**)

Modelo de Lenguaje de *n*-gramas

- Podemos extender a tri-gramas, 4-gramas y 5-gramas
- Pero en general es un **modelo insuficiente** del lenguaje
(lenguaje tiene **dependencias de mucha distancia**)

“**la impresora** que justo ayer había comprado en la oferta de fin de mes, cuando abrieron la puerta **se cayó** al piso”

Modelo de Lenguaje de *n*-gramas

- Podemos extender a tri-gramas, 4-gramas y 5-gramas
- Pero en general es un **modelo insuficiente** del lenguaje
(lenguaje tiene **dependencias de mucha distancia**)

“**la impresora** que justo ayer había comprado en la oferta de fin de mes, cuando abrieron la puerta **se cayó** al piso”

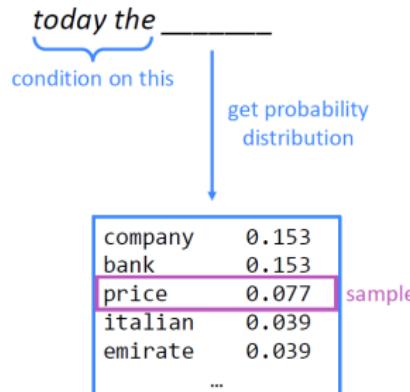
- Continuan problemas de **dispersión** y **almacenamiento**

Modelo de Lenguaje de *n*-gramas

- Podemos extender a tri-gramas, 4-gramas y 5-gramas
- Pero en general es un **modelo insuficiente** del lenguaje (lenguaje tiene **dependencias de mucha distancia**)

“**la impresora** que justo ayer había comprado en la oferta de fin de mes, cuando abrieron la puerta **se cayó** al piso”

- Continuan problemas de **dispersión** y **almacenamiento**
- Pero desempeño **aceptable** en la **generación de texto**



Modelo de Lenguaje Neuronal (con ventana fija) [Manning]

output distribution

$$\hat{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$$

hidden layer

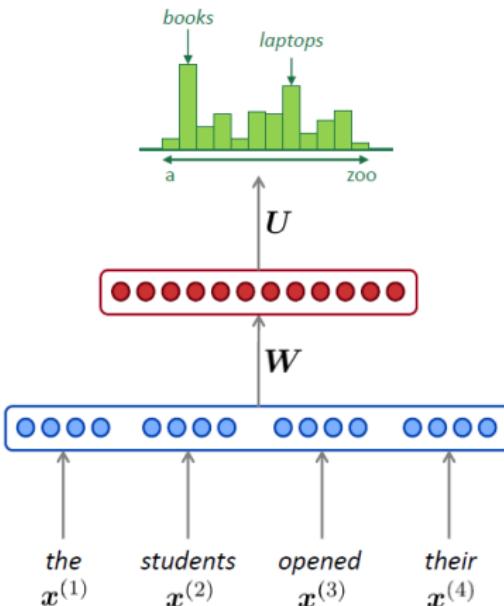
$$h = f(We + b_1)$$

concatenated word embeddings

$$e = [e^{(1)}; e^{(2)}; e^{(3)}; e^{(4)}]$$

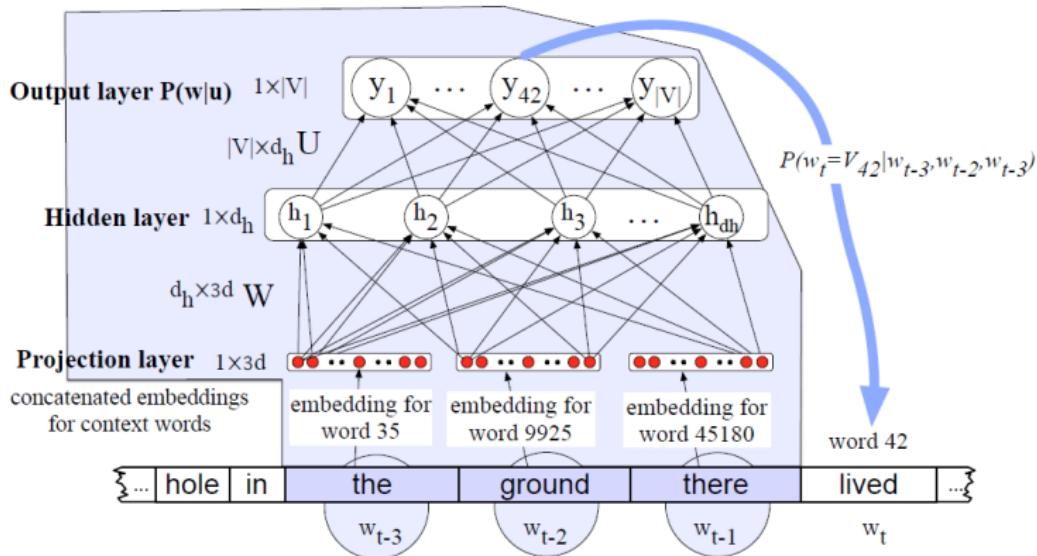
words / one-hot vectors

$$x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$$



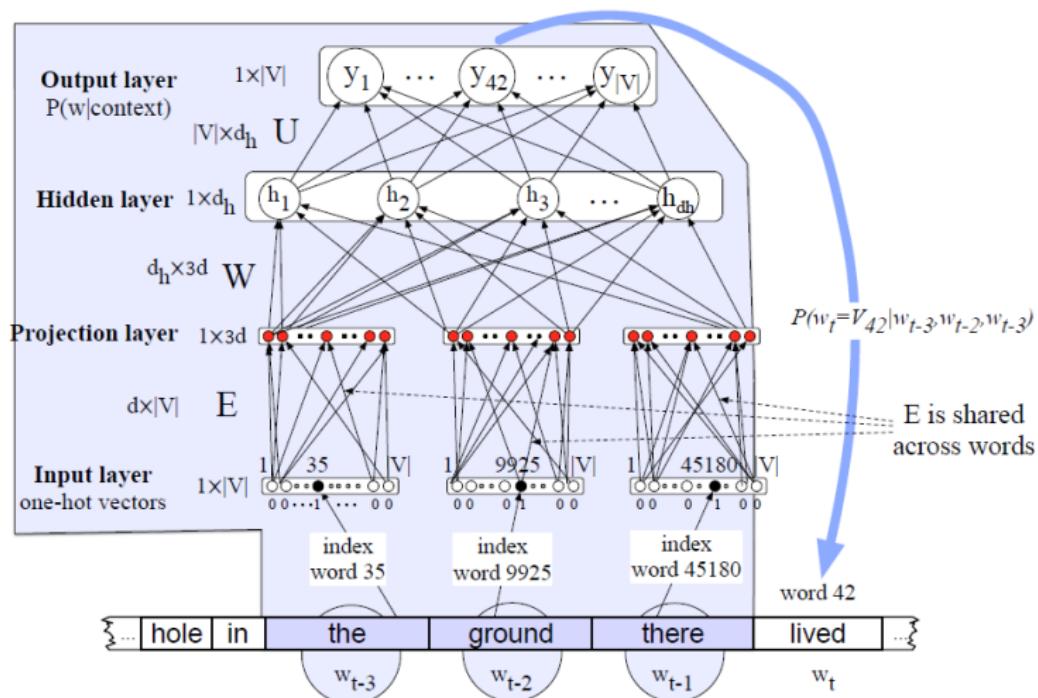
Modelo de Lenguaje Neuronal (con ventana fija)

algunos detalles adicionales ... [Jurafsky]



Modelo de Lenguaje Neuronal (con ventana fija)

y sobre la capa de embedding ...



Modelo de Lenguaje Neuronal (con ventana fija)

Modelo de Lenguaje Neuronal (con ventana fija)

Mejoras (sobre el ML de n -gramas)

- No hay problemas de dispersión de datos
- No es necesario almacenar todos los n -gramas observados

output distribution

$$\hat{y} = \text{softmax}(\mathbf{U}\mathbf{h} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

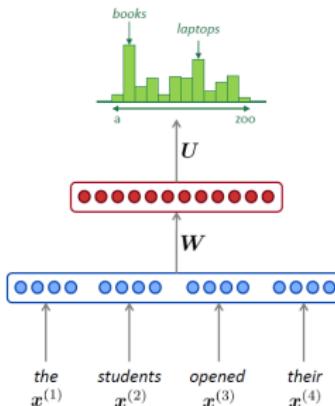
$$\mathbf{h} = f(\mathbf{W}\mathbf{e} + \mathbf{b}_1)$$

concatenated word embeddings

$$\mathbf{e} = [\mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \mathbf{e}^{(3)}; \mathbf{e}^{(4)}]$$

words / one-hot vectors

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}$$

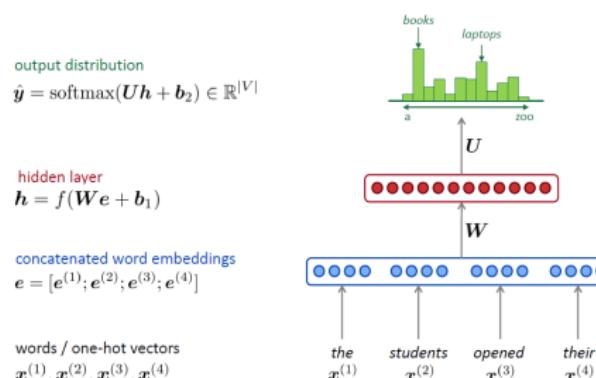


Modelo de Lenguaje Neuronal (con ventana fija)

Modelo de Lenguaje Neuronal (con ventana fija)

Problemas que persisten

- La ventana fija es **demasiado pequeña**
- **Agrandar la ventana agranda** W .
- La ventana **nunca** puede ser lo **suficientemente larga**
- $x^{(1)}$ y $x^{(2)}$ son multiplicados por diferentes pesos en W (no hay **simetría** en el procesamiento de las entradas)

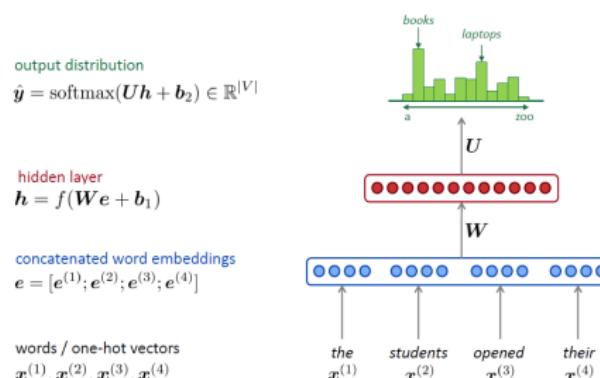


Modelo de Lenguaje Neuronal (con ventana fija)

Modelo de Lenguaje Neuronal (con ventana fija)

Problemas que persisten

- La ventana fija es **demasiado pequeña**
- **Agrandar la ventana agranda** W .
- La ventana **nunca** puede ser lo **suficientemente larga**
- $x^{(1)}$ y $x^{(2)}$ son multiplicados por diferentes pesos en W (no hay **simetría** en el procesamiento de las entradas)



Modelo de Lenguaje Neuronal (con ventana fija)

Modelo de Lenguaje Neuronal (con ventana fija)

Conclusión: necesitamos una arquitectura neuronal que pueda procesar entradas de cualquier longitud !!!

output distribution

$$\hat{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$$

hidden layer

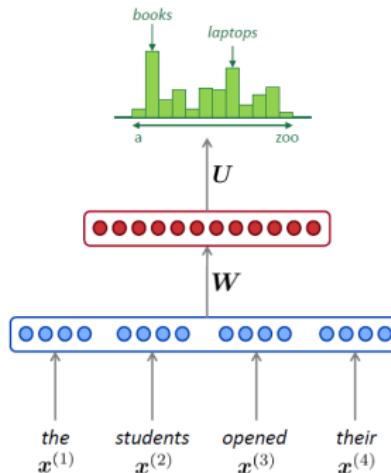
$$h = f(We + b_1)$$

concatenated word embeddings

$$e = [e^{(1)}; e^{(2)}; e^{(3)}; e^{(4)}]$$

words / one-hot vectors

$$x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$$



Redes Neuronales Recurrentes

- **Familia** de arquitecturas de redes neuronales

Redes Neuronales Recurrentes

- **Familia** de arquitecturas de redes neuronales
- Pueden procesar entradas de **cualquier longitud**

Redes Neuronales Recurrentes

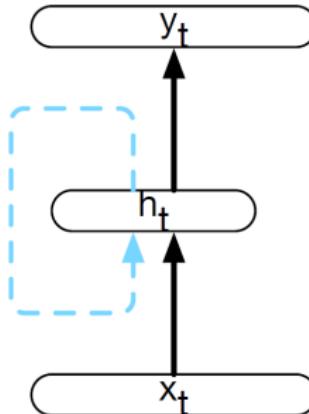
- **Familia** de arquitecturas de redes neuronales
- Pueden procesar entradas de **cualquier longitud**
- Para ello necesitan alguna forma de **memoria**

Redes Neuronales Recurrentes

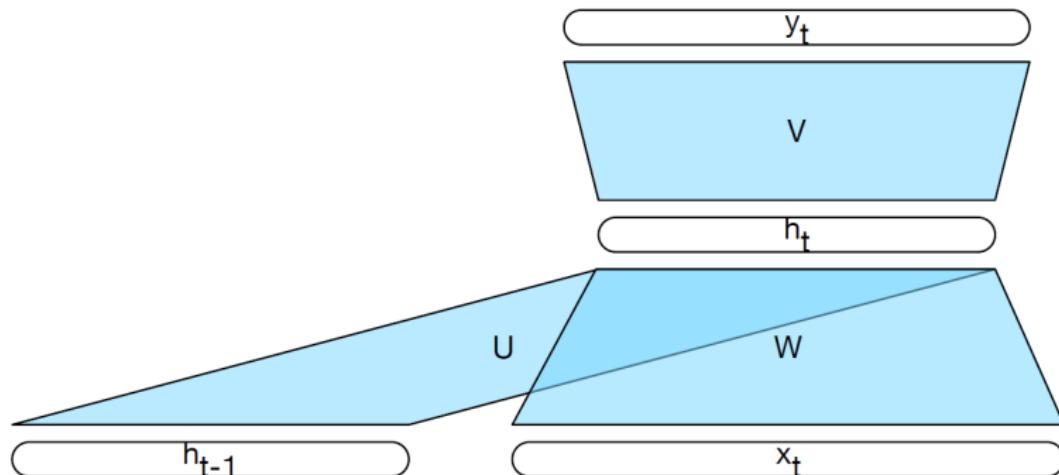
- **Familia** de arquitecturas de redes neuronales
- Pueden procesar entradas de **cualquier longitud**
- Para ello necesitan alguna forma de **memoria**
- Así, la respuesta (salida) de una RNN **depende**, además de su **entrada actual**, de la información del pasado almacenada en su **memoria**

Redes Neuronales Recurrentes

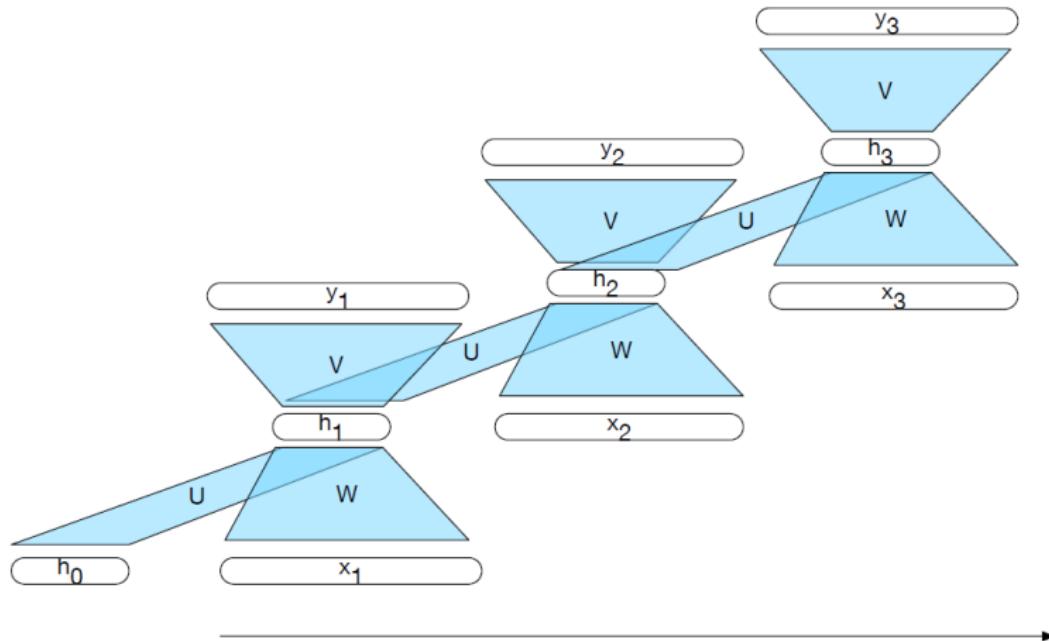
- **Familia** de arquitecturas de redes neuronales
- Pueden procesar entradas de **cualquier longitud**
- Para ello necesitan alguna forma de **memoria**
- Así, la respuesta (salida) de una RNN **depende**, además de su **entrada actual**, de la información del pasado almacenada en su **memoria**
- **Ejemplo:** RNN simple (Elman)



Red Neuronal Recurrente simple (visión feedforward) [Jurafsky]

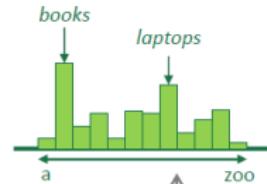


Red Neuronal Recurrente simple (desenrollada) [Jurafsky]



Modelos de lenguaje basados en RNN [Manning]

$$\hat{y}^{(4)} = P(x^{(5)} | \text{the students opened their})$$



output distribution

$$\hat{y}^{(t)} = \text{softmax}(\mathbf{U}h^{(t)} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma(\mathbf{W}_h h^{(t-1)} + \mathbf{W}_e e^{(t)} + b_1)$$

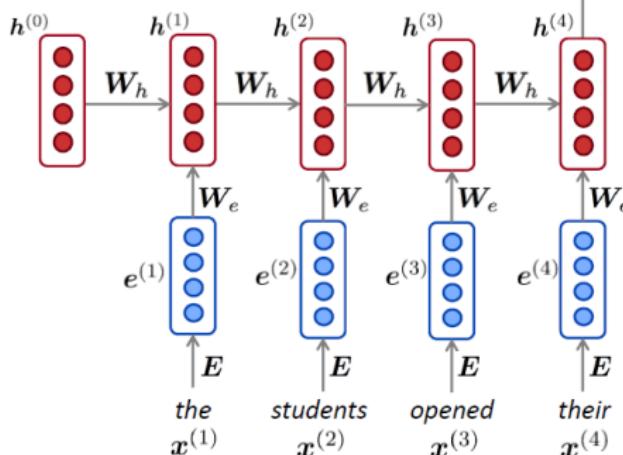
$h^{(0)}$ is the initial hidden state

word embeddings

$$e^{(t)} = \mathbf{E}x^{(t)}$$

words / one-hot vectors

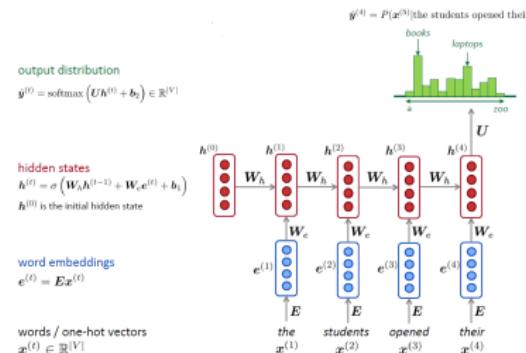
$$x^{(t)} \in \mathbb{R}^{|V|}$$



Modelos de lenguaje basados en RNN

Ventajas

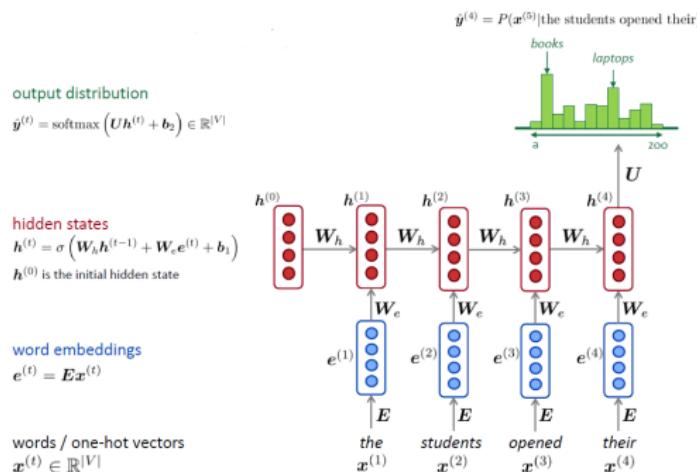
- Puede procesar entradas de **cualquier longitud**
- Computación para el **paso t** puede (en teoría) usar información de **muchos pasos atrás**
- **Tamaño del modelo no aumenta** con entrada más larga
- Los mismos pesos (W) aplicados en cada paso de tiempo
⇒ **simetría** en cómo las entradas son procesadas



Modelos de lenguaje basados en RNN

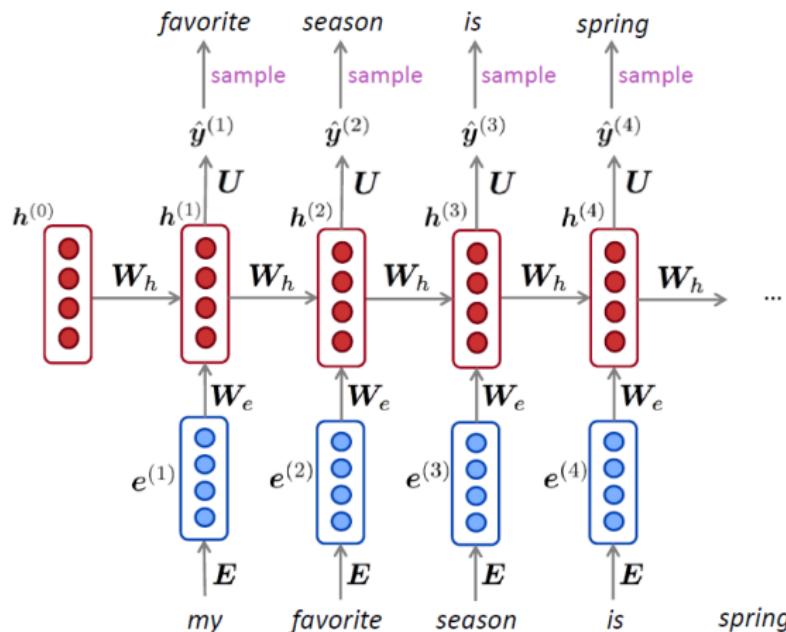
Desventajas

- La computación recurrente es **lenta**
- En **la práctica** es muy difícil acceder a información de muchos pasos atrás



Generando texto con un modelo RNN

Como antes, se puede **generar texto por muestreo repetido**. La salida de un muestreo, es la entrada del próximo paso.

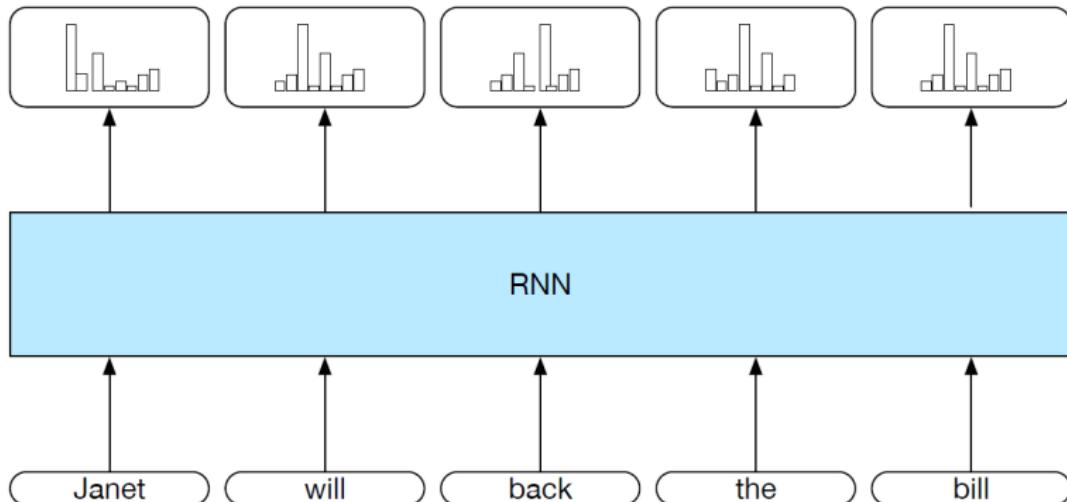


Generando texto con un modelo RNN

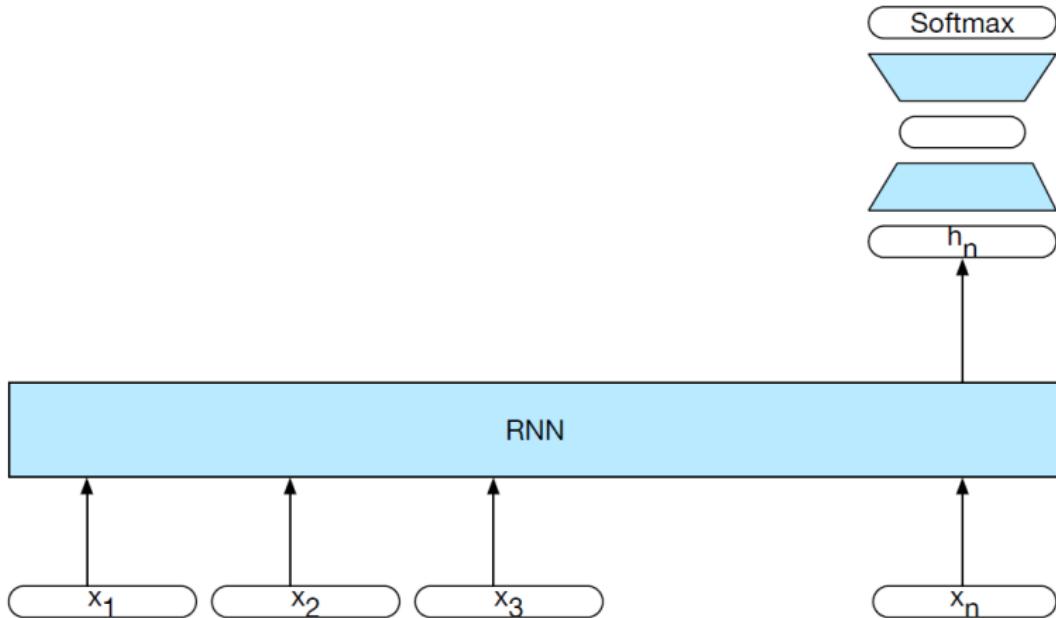
Se puede entrenar un modelo de lenguaje RNN sobre cualquier tipo de texto, y luego generar texto con ese estilo:

- Discursos de Obama
- Harry Potter
- Recetas de cocina

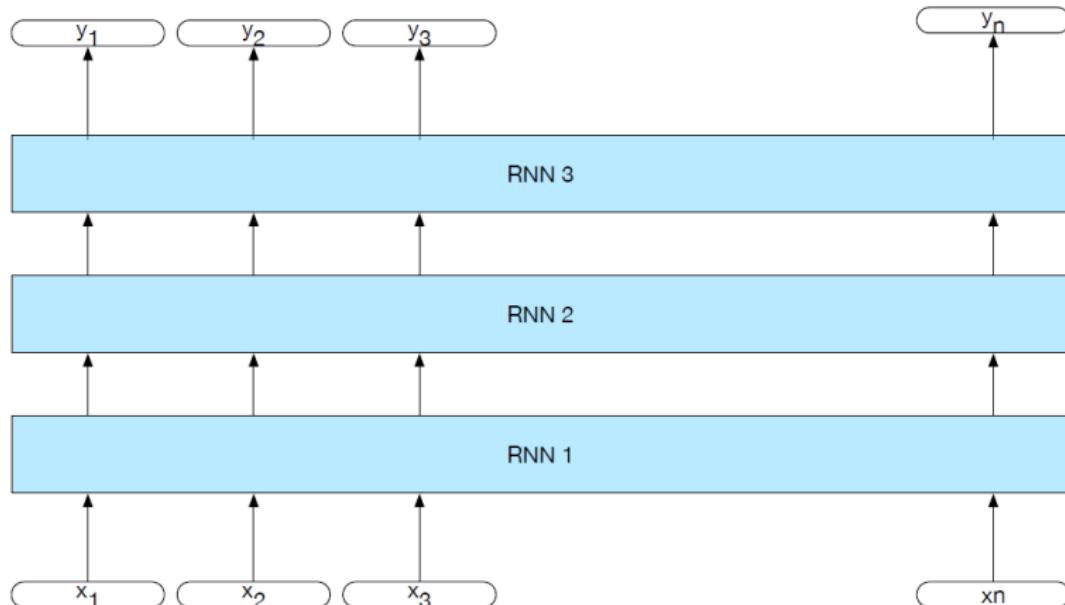
Aplicaciones de RNNs: POS tagging



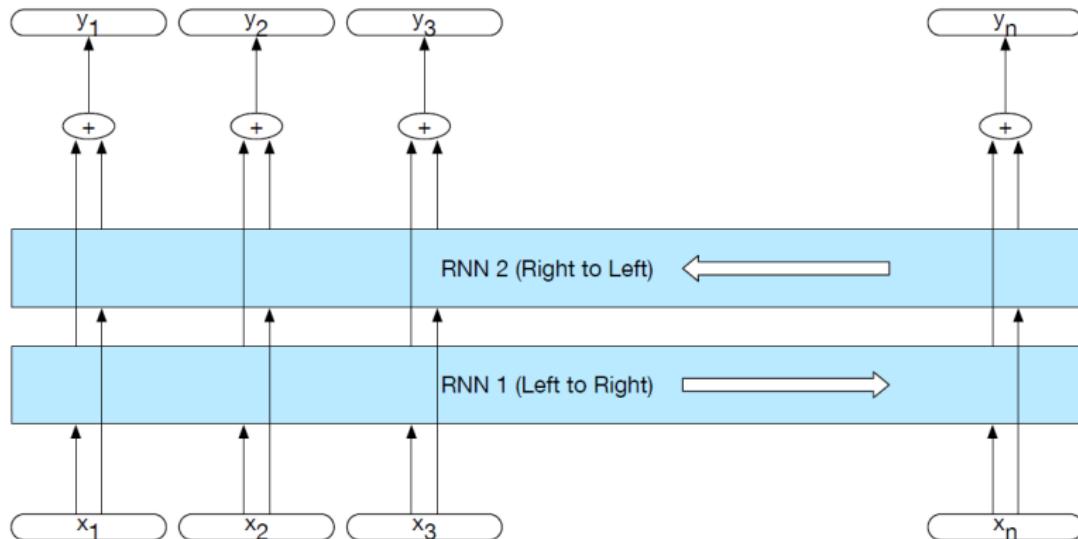
Aplicaciones de RNNs: clasificación de secuencias



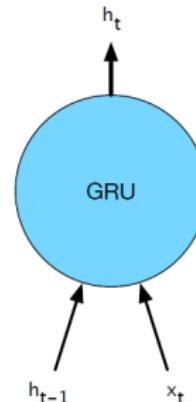
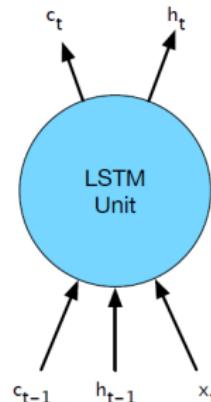
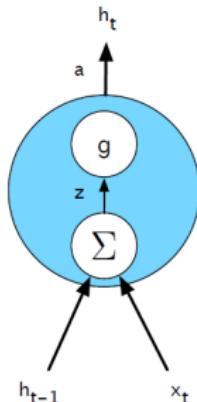
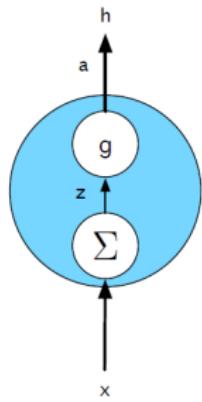
Otras arquitecturas RNNs: apiladas



Otras arquitecturas RNNs: bi-direccionales



Resumen: unidades neuronales básicas



(a)

(b)

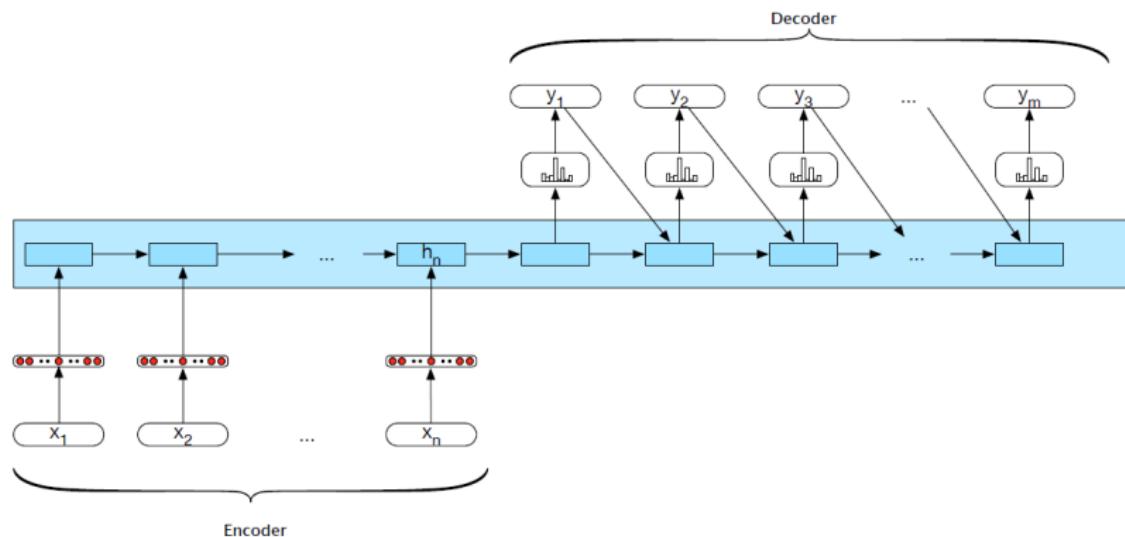
(c)

(d)

- (a) Red FF simple
- (b) Red neuronal recurrente simple
- (c) Long short-term memory (LSTM)
- (d) Gated recurrent units (GRU)

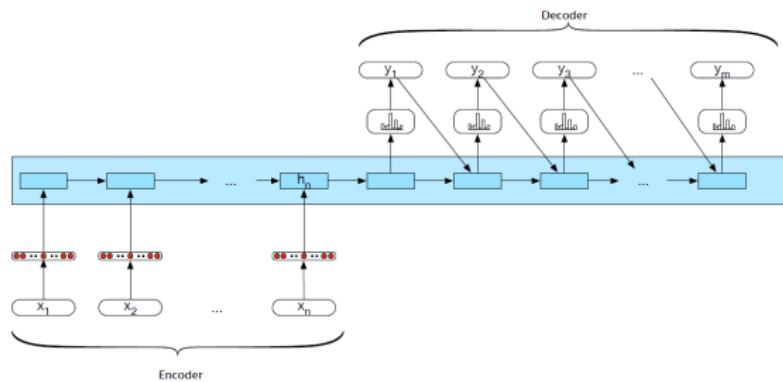
Enfoques recientes

Redes/modelos **encoder-decoder** (o **sequence-to-sequence**)



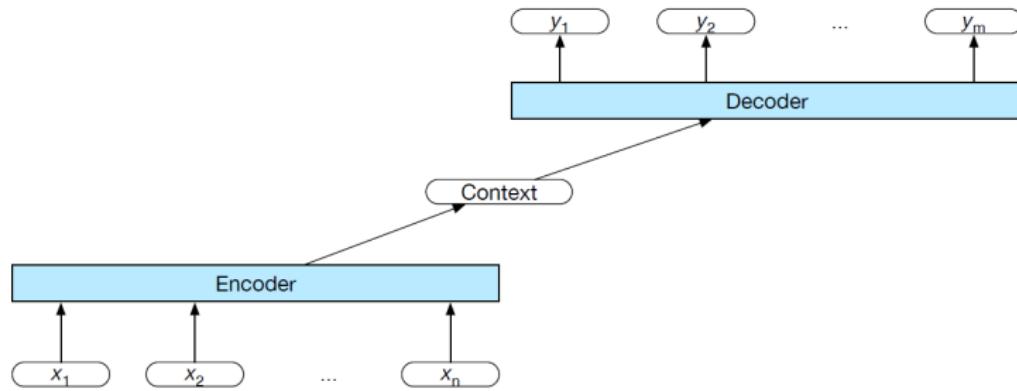
Modelos *encoder-decoder*

Ejemplo: arquitectura básica **encoder-decoder** basada en RNNs



El estado oculto final (h_n) de la **RNN encoder** sirve como contexto para el **decoder** en su rol como (h_0) de la **RNN encoder**

Redes encoder-decoder genéricas



El **contexto** es una **función** del vector de representaciones de estado contextualizadas y puede ser usada por el decoder de **distintas maneras**

Redes encoder-decoder *genéricas*

- Ya no asumimos que el **encoder** y **decoder** tienen **igual** estructura interna (RNNs)

Redes encoder-decoder *genéricas*

- Ya no asumimos que el **encoder** y **decoder** tienen **igual estructura interna** (RNNs)
- Tampoco que el estado final del encoder es el **único contexto disponible** para el decoder

Redes encoder-decoder *genéricas*

- Ya no asumimos que el **encoder** y **decoder** tienen **igual estructura interna** (RNNs)
- Tampoco que el estado final del encoder es el **único contexto disponible** para el decoder
- Ni que este contexto es disponible **únicamente** al decoder como su **estado oculto inicial**

Redes encoder-decoder *genéricas*

- Ya no asumimos que el **encoder** y **decoder** tienen **igual estructura interna** (RNNs)
- Tampoco que el estado final del encoder es el **único contexto disponible** para el decoder
- Ni que este contexto es disponible **únicamente** al decoder como su **estado oculto inicial**
- Forman la base de los mecanismos de **atención**

Redes encoder-decoder *genéricas*

- Ya no asumimos que el **encoder** y **decoder** tienen **igual estructura interna** (RNNs)
- Tampoco que el estado final del encoder es el **único contexto disponible** para el decoder
- Ni que este contexto es disponible **únicamente** al decoder como su **estado oculto inicial**
- Forman la base de los mecanismos de **atención**
- Extienden su uso a un ámbito más amplio que el original de **machine translation**, incluyendo **summarization**, **question answering**, **image captioning** y **visual question answering**

Material Complementario

- [Manning] [https://www.youtube.com/playlist?
list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z](https://www.youtube.com/playlist?list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z)

Material Complementario

- [Manning] [https://www.youtube.com/playlist?
list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z](https://www.youtube.com/playlist?list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z)
- [Jurafsky]
<https://web.stanford.edu/~jurafsky/slp3/>
(Caps. 3, 6, 7, 9, 10)

Material Complementario

- [Manning] [https://www.youtube.com/playlist?
list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z](https://www.youtube.com/playlist?list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z)
- [Jurafsky]
<https://web.stanford.edu/~jurafsky/slp3/>
(Caps. 3, 6, 7, 9, 10)
- [Chollet] “Deep Learning with Python”. Francois Chollet.
[Cap. 6] *Deep Learning for text and sequences.*