07MIAR – Redes Neuronales y Deep Learning

Universidad Internacional de Valencia

Deep Learning para texto y secuencias lógicas

De:
Planeta Formación y Universidades

Contenidos

1. Procesamiento del Lenguaje Natural (NLP)
2. De texto a representaciones numéricas
3. Word2Vec
4. Redes neuronales recurrentes
5. Introducción a transformers

Contenidos

1. **Procesamiento del Lenguaje Natural (NLP)**
2. De texto a representaciones numéricas
3. Word2Vec
4. Redes neuronales recurrentes
5. Introducción a *transformers*

Procesamiento del Lenguaje Natural (NLP)

- El ***Natural Language Processing*** (NLP) es la **intersección** entre los campos de las **ciencias de la computación**, la **inteligencia artificial** y la **lingüística**. El NLP estudia las **interacciones** entre las **computadoras** y el **lenguaje humano**
- Se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la **comunicación** entre **personas** y **máquinas** por medio del lenguaje natural, es decir, de las **lenguas del mundo**
- Dentro del NLP se identifican una serie de **tareas** a resolver:

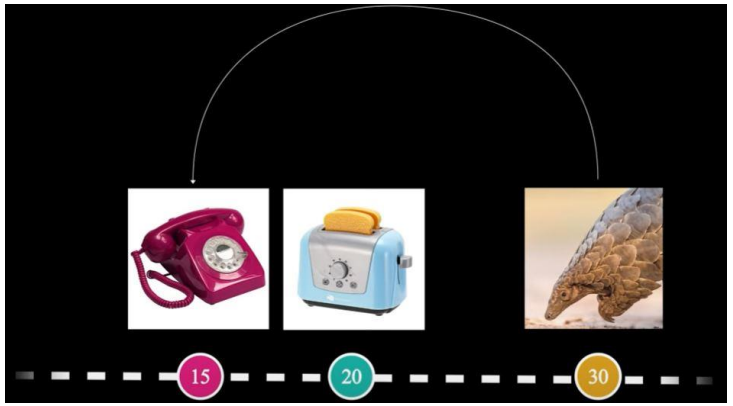
  - Speech Recognition
  - Speech to text
  - Machine Translation
  - Text classification
  - Sentiment analysis
  - Text generation (e.g. Q&A)
  - Text recognition (OCR)
  - Image/Video understanding
  - Text summarization

Contenidos

1. Procesamiento del Lenguaje Natural (NLP)
2. **De texto a representaciones numéricas**
3. Word2Vec
4. Redes neuronales recurrentes
5. Introducción a *transformers*

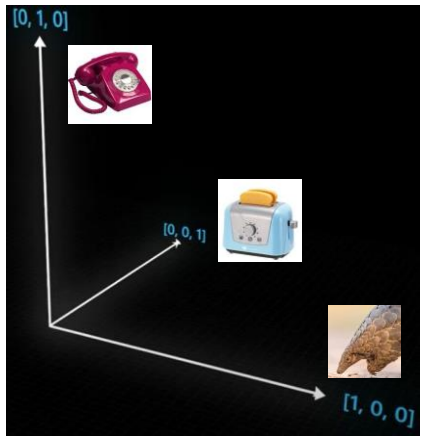
De texto a representaciones numéricas

- Tal y como hemos visto a lo largo del curso, una **red neuronal** (independientemente de su arquitectura) viene **definida por** una serie de **parámetros numéricos** a optimizar. Dichos parámetros **restringen el tipo de** datos a la **entrada**. Una red neuronal no sabe manejar texto a la entrada
- La **solución** más **intuitiva** es **asignar un número entero** (etiqueta categórica) **a cada** una de las **unidades de texto** con las que se trabaje (carácter/palabra)
- Esta **solución no** es **válida** desde el punto de vista de una red neuronal puesto que la **red** intrínsecamente extrae **patrones de ordenación** de los **datos numéricos** mientras que el **texto** sigue una **secuencia lógica gramatical**.



De texto a representaciones numéricas

- Una solución para dotar de **independencia** a la codificación texto – dato numérico es la codificación **one-hot encoding.** Mediante esta codificación podemos establecer **una dimensión a cada carácter/palabra**
- Como sabemos esta codificación se compone de un **vector** de tantos **ceros** como palabras distintas se quieran codificar, indicando **con un 1** cada palabra diferente



dog = [0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 … 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

cat = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 … 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0]

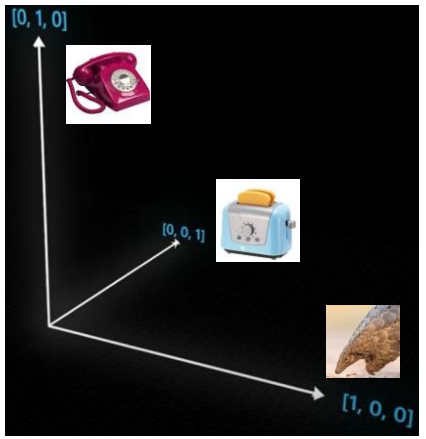
De texto a representaciones numéricas

- El problema de este tipo de codificación es que **en NLP** vamos a manejar **grandes vocabularios** de texto (**Corpus**) para el entrenamiento de modelos

dog = [0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 … 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
cat = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 … 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0]

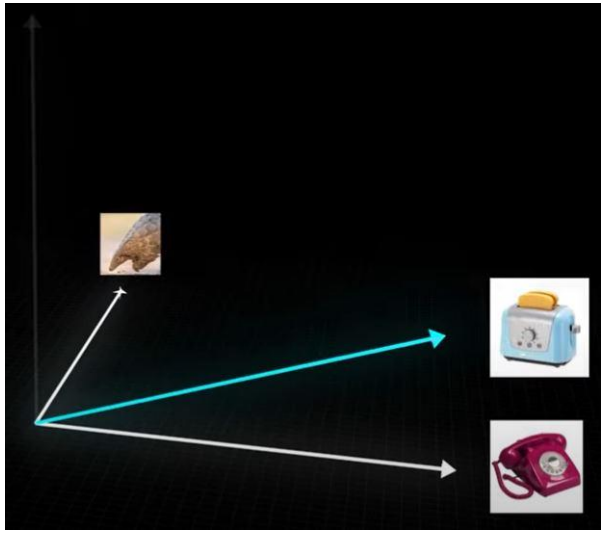
Vocabulary size

- Otro inconveniente es que **palabras similares no tienen** porque estar representadas por **vectores cercanos**
- **Tantas dimensiones como palabras** a codificar
- **Vectores** muy **dispersos** (todo ceros menos un uno)



Word vectors/embeddings

- ¿Cómo podemos evitar las limitaciones anteriores?
  - IDEA: Mapear palabras en un nuevo **espacio vectorial** más **denso** o concentrado (***Word embeddings***)
  - OBJETIVO: Obtener una representación de palabras que obtenga provecho de la "**similitud semántica"** de las mismas
  - ¿CÓMO?: Resolviendo un problema de **aprendizaje Supervisado**
  - VENTAJAS: **Vectores** más pequeños y **compactos**



Contenidos

1. Procesamiento del Lenguaje Natural (NLP)
2. De texto a representaciones numéricas
3. **Word2Vec**
4. Redes neuronales recurrentes
5. Introducción a *transformers*

Word2Vec

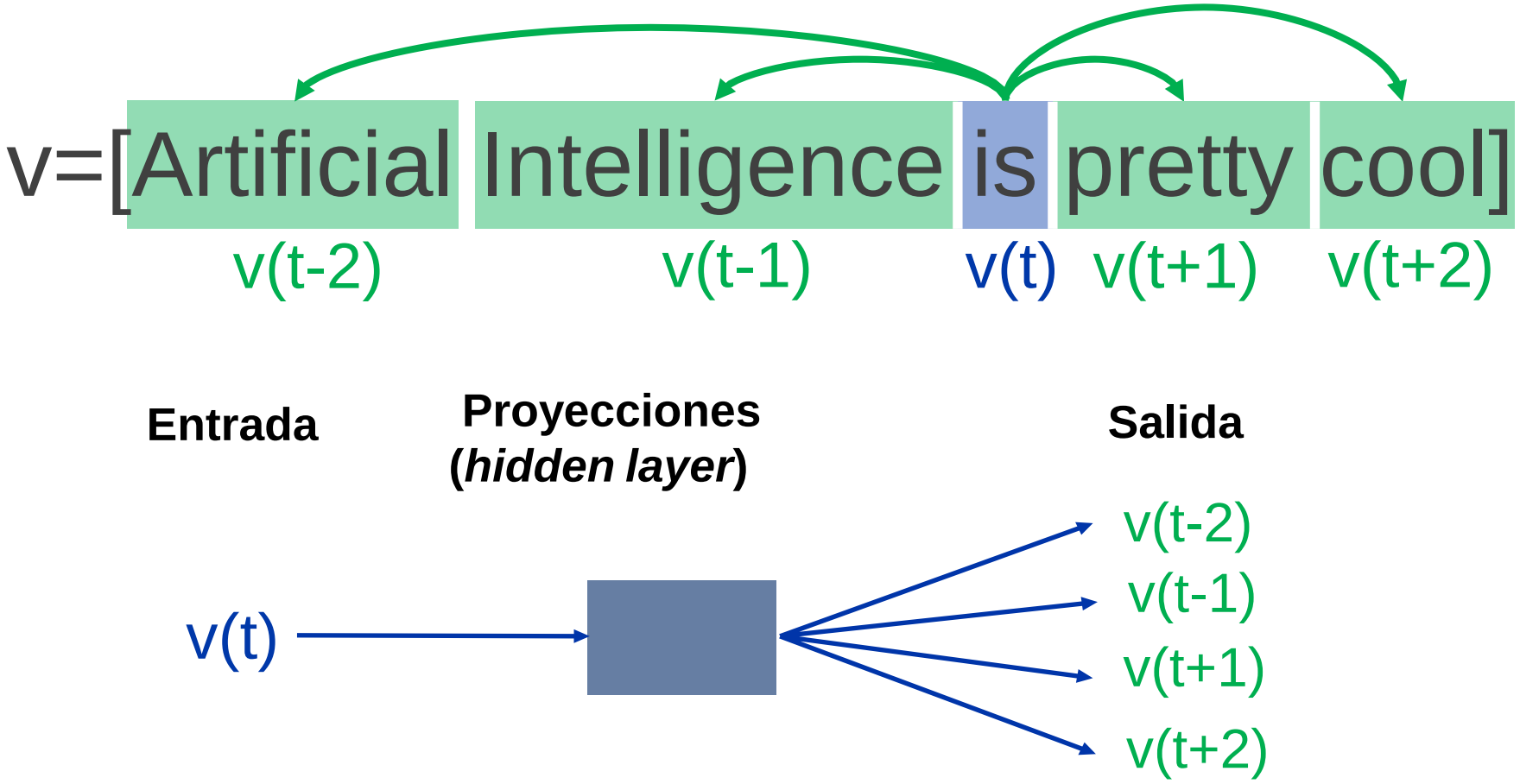
- ¿Cómo podemos obtener estas representaciones densas de palabras (***Word embeddings***)?
- El **algoritmo Word2Vec** es el más popular en la literatura para entrenar *Word embeddings* con propósitos generalistas
- Se basa en **predecir el vecindario de palabras** para cada una de las palabras que componen un texto
- Existen **dos versiones** de este algoritmo:
  - Continuous bag of words (**CBOW**)
  - Modelo **Skip-gram**

Tomáš Mikolov

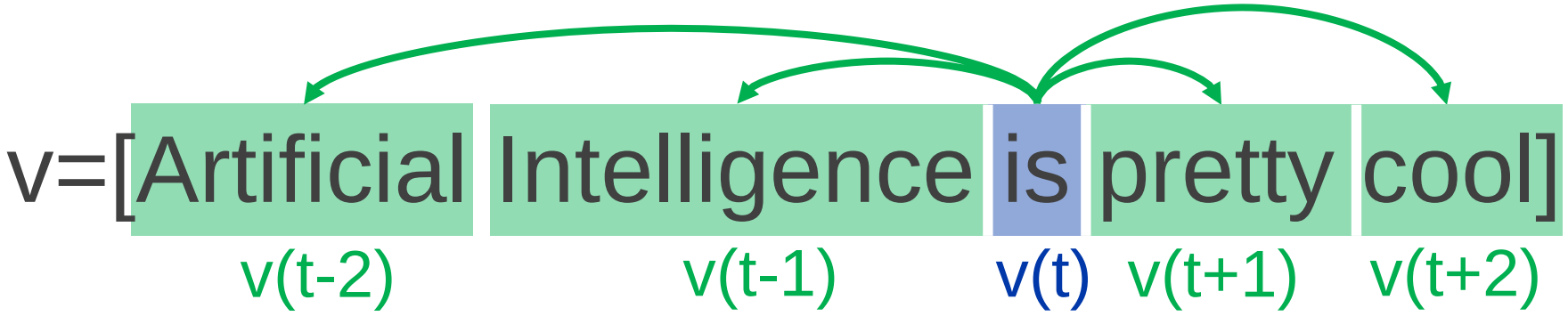
Facebook member
Google ex-member

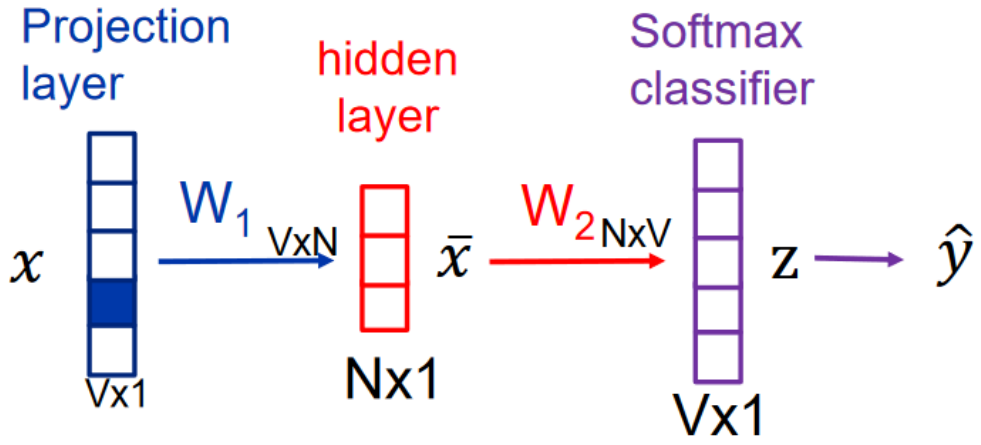
Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*
Mikolov e t al. *Distributed Representations of Words and Phrases and their Compositionality*

Modelo Skip-gram

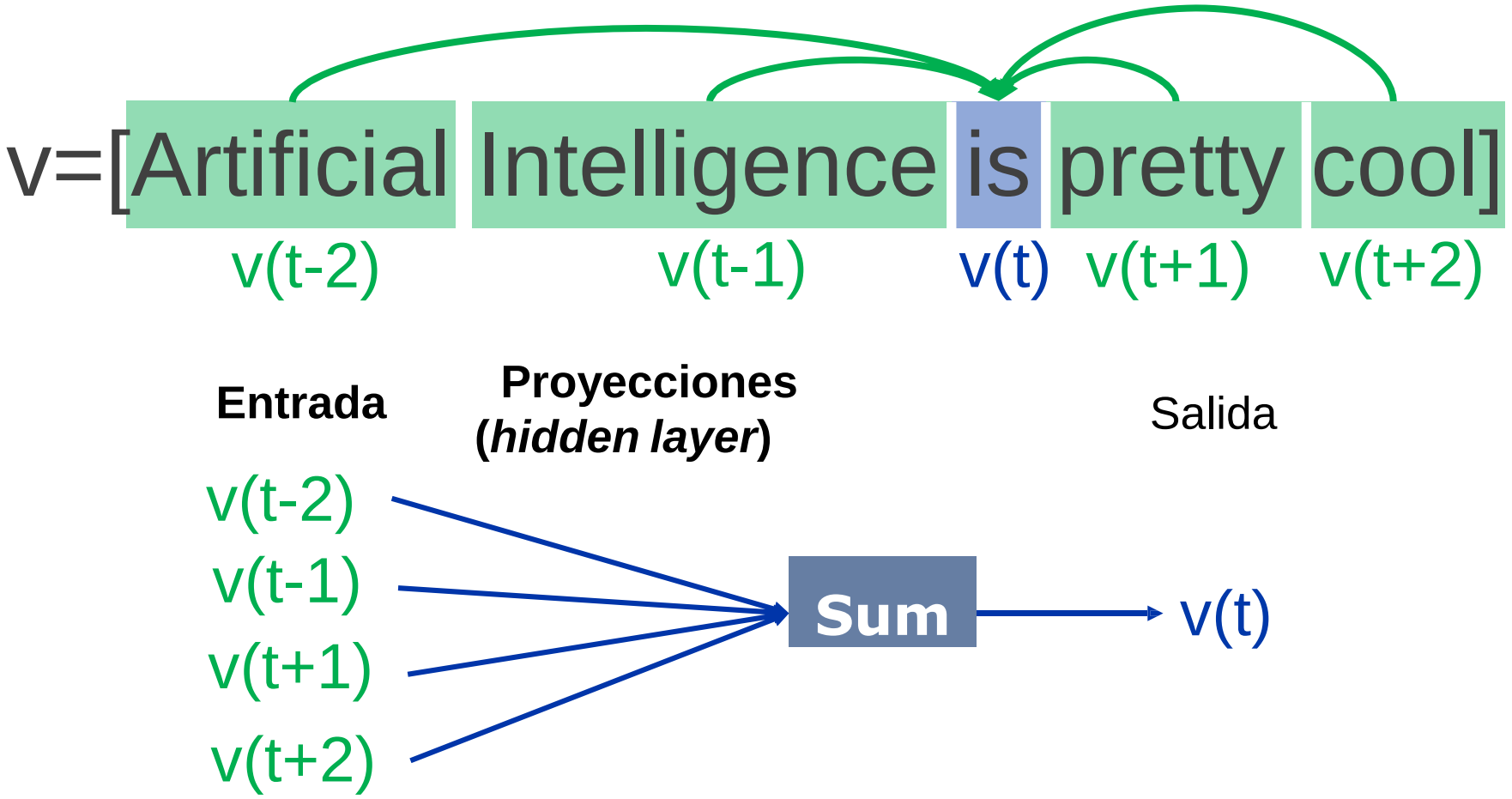


Modelo Skip-gram

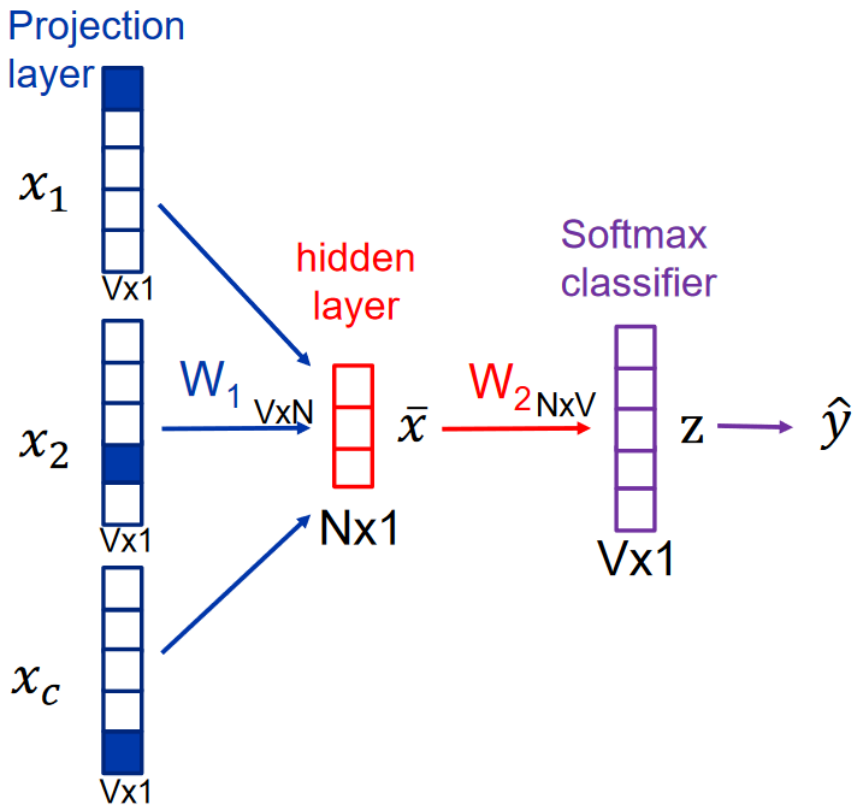




Continuous Bag-of-Words



Continuous Bag-of-Words



$$\bar{x} = \frac{1}{2c}\sum_{i}^{c} W_1 \cdot x_i$$

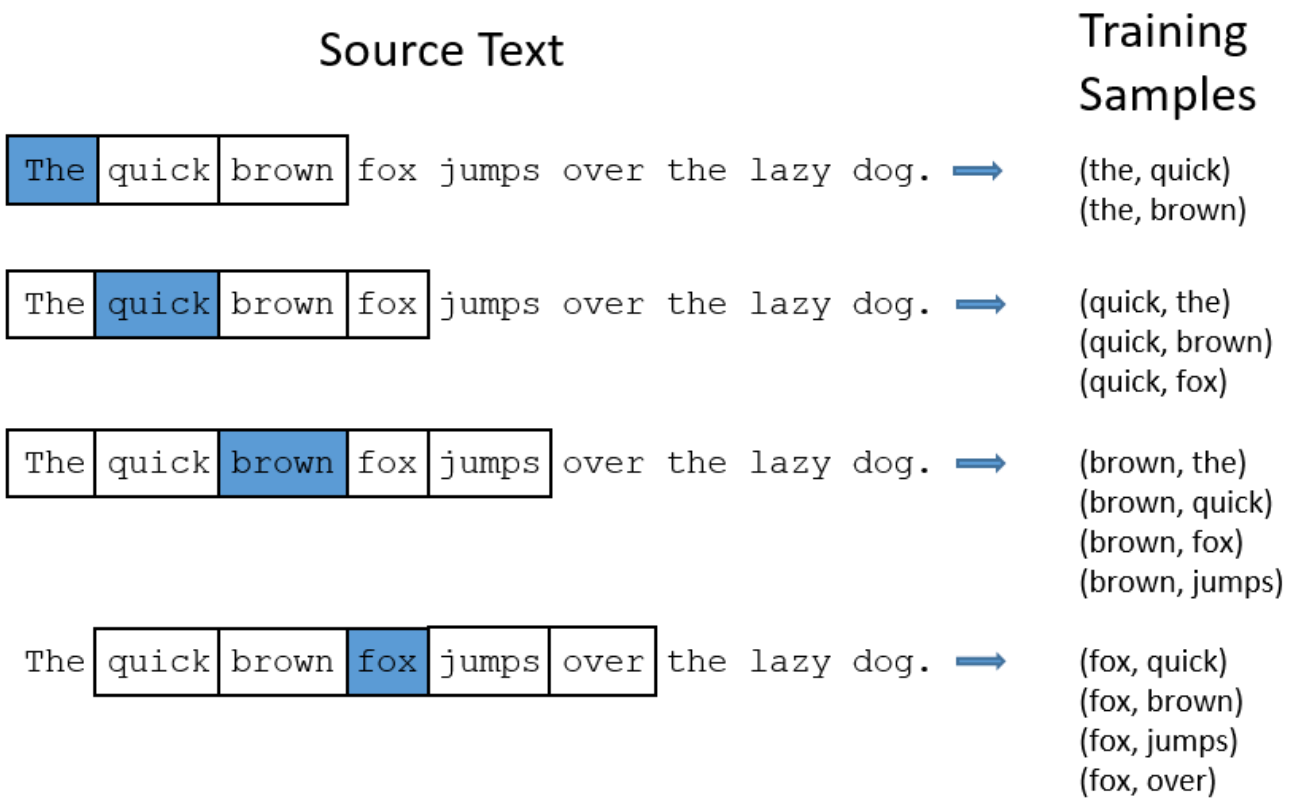
$$z = W_2 \cdot \bar{x}$$

$$\hat{y}_i = softmax(z)$$

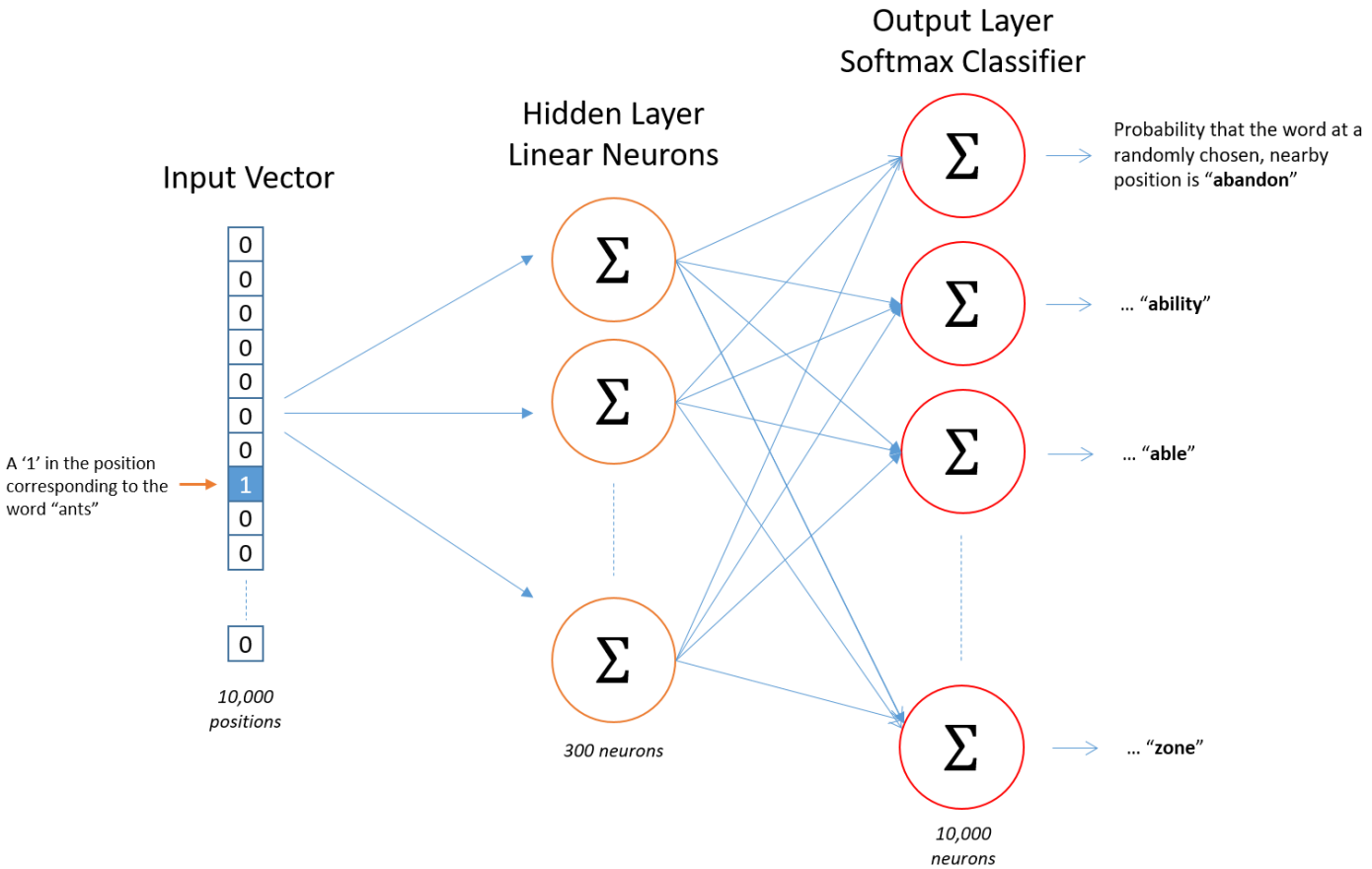
Cost function

argmin $H(\hat{y}, y) = -y_i \log(\hat{y}_i)$

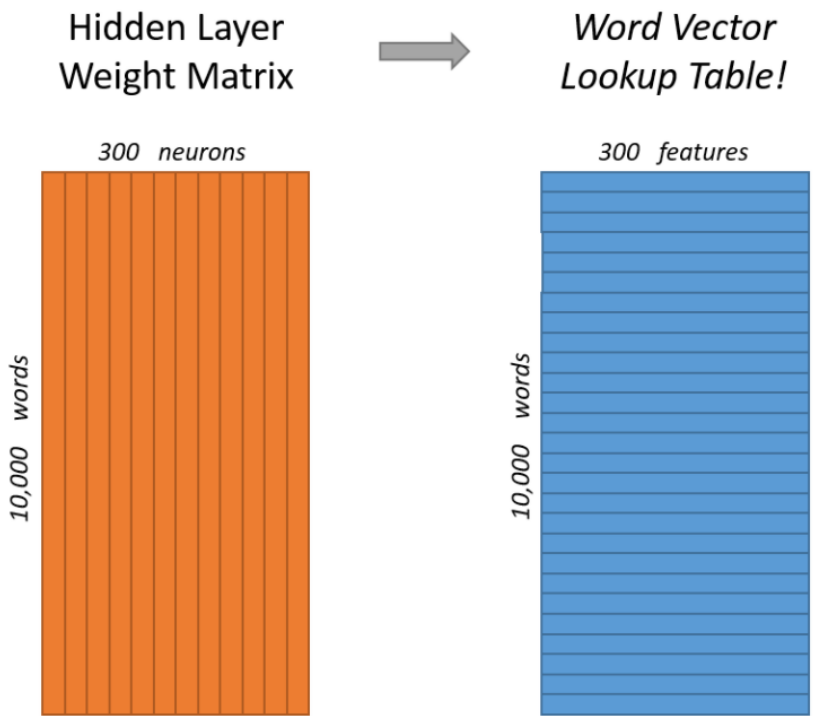
Word2Vec


Source Text
Training Samples
The quick brown fox jumps over the lazy dog.
(the, quick)
(the, brown)
The quick brown fox jumps over the lazy dog.
(quick, the)
(quick, brown)
(quick, fox)
The quick brown fox jumps over the lazy dog.
(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)
The quick brown fox jumps over the lazy dog.
(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)

Word2Vec

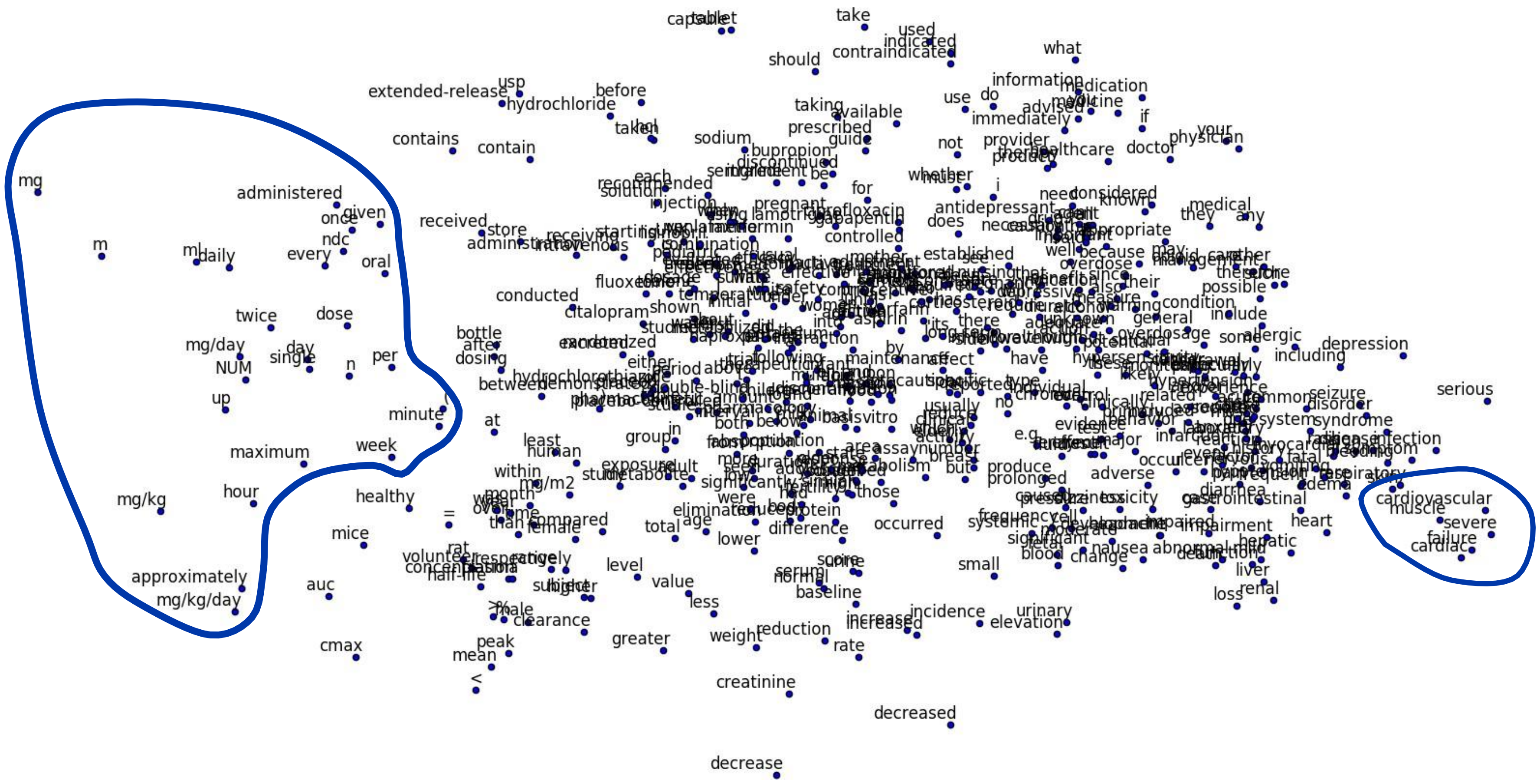

Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"
A '1' in the position corresponding to the word "ants"
0
0
0
0
0
0
0
1
0
0
0
10,000 positions
300 neurons
10,000 neurons

Word2Vec



$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Word2Vec

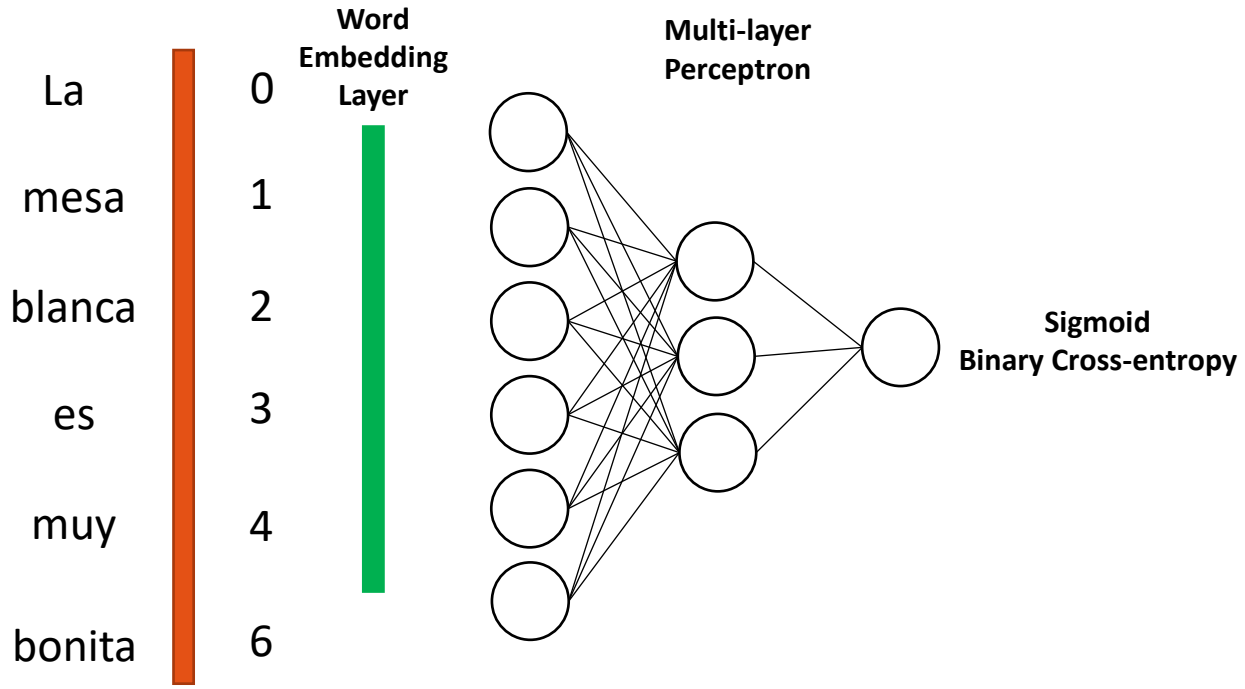

take
used
should
what
extended-release
hydrochloride
before
information
contains
contain
sodium
not
doctor
mg
administered
given
once
received
store
ndc
every
oral
m
daily
twice
dose
conducted
citalopram
mg/day
day
single
per
NUM
up
minute
at
maximum
week
mg/kg
hour
healthy
mice
approximately
mg/kg/day
auc
cmax
clearance
greater
peak
mean
level
value
less
weight
reduction
rate
creatinine
decreased
decrease
increased
incidence
urinary
elevation
small
occurred
difference
total
lower
baseline
depression
including
serious
seizure
syndrome
cardiovascular
muscle
severe
failure
cardiac
heart
hepatic
liver
renal
loss
adverse
produce
prolonged
gastrointestinal
medical
physician
possible
general
include
allergic

Contenidos

1. Procesamiento del Lenguaje Natural (NLP)
2. De texto a representaciones numéricas
3. Word2Vec
4. **Redes neuronales recurrentes**
5. Introducción a *transformers*

MLP en análisis de texto

- ¿Es posible llevar a cabo tareas de ***Natural Language Processing*** empleando un **Perceptrón Multicapa**?¿Es la arquitectura de red más adecuada? Imaginemos la tarea de clasificación binaria de secuencias de texto:

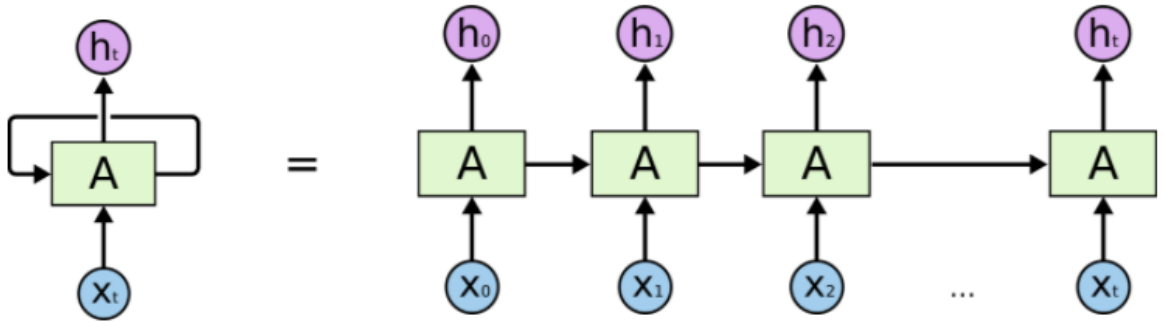


- Es posible emplear un **MLP** para tareas de NLP pero **no es la arquitectura** más **adecuada** ya que se pierde la **"secuencialidad gramatical"** de las palabras

Redes Neuronales Recurrentes

- Un **MLP** es **incapaz** de **retener dependencias entre muestras** y estamos ante un tipo de dato puramente secuencial
- Las **unidades básicas** que componen el **texto** (caracteres o palabras) son totalmente **dependientes** unas de otras, i.e. “secuencialidad gramatical”
- Las redes neuronales recurrentes o ***recurrent neural networks*** (RNN) surgen para superar esta limitación
- Necesidad de que la **información** de entrada persista entre muestras (Loop)

El tío de María nació en Francia. Recuerda ciertas palabras en ____

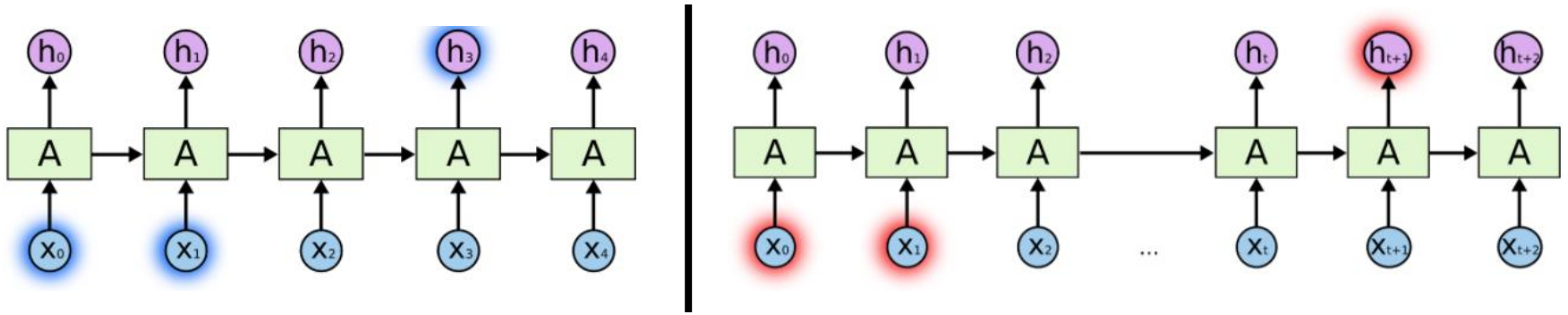


Redes Neuronales Recurrentes

- Podemos pensar en **una RNN** como **múltiples copias** de la **misma red**, cada una **pasándole** el **mensaje a la siguiente**
- No solo se emplean en texto, son la **mejor opción** siempre que se trabaje con **datos** en los que exista una **componente secuencial** (e.g. series temporales)
- ¿Pueden las RNN retener la información relevante de **muchos instantes atrás** en la secuencia? Conforme **aumenta el "gap"** las **RNN pierden** esta **propiedad**

Las nubes están en el *cielo*

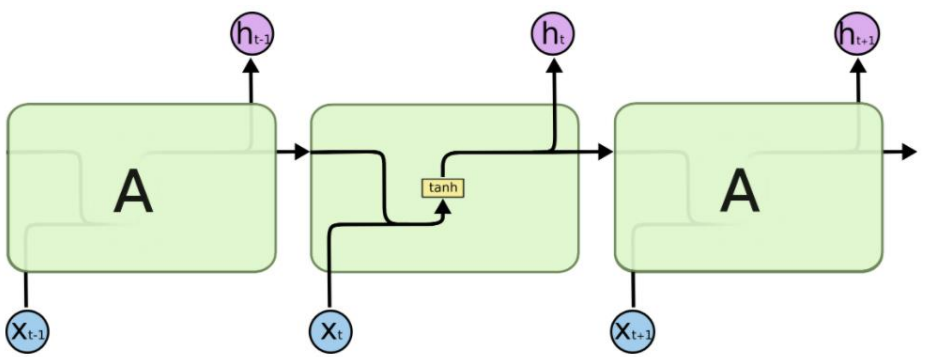
Yo crecí en Alemania… Puedo hablar *Alemán* de manera flúida



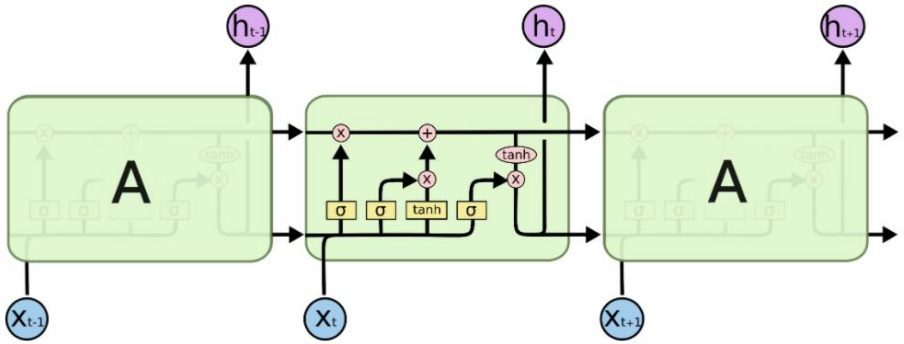
Unidades Long-Short Term Memory

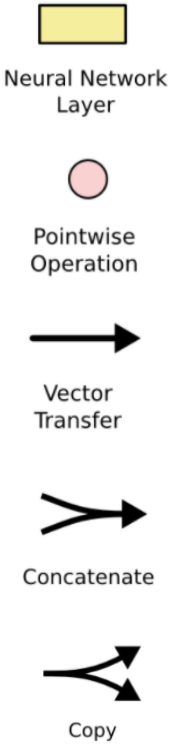
- Con el objetivo de preservar el máximo número de instantes la información secuencial nacen un tipo especial de unidades en las RNN, las **unidades Long-Short Term Memory** (LSTM)

Red neuronal recurrente



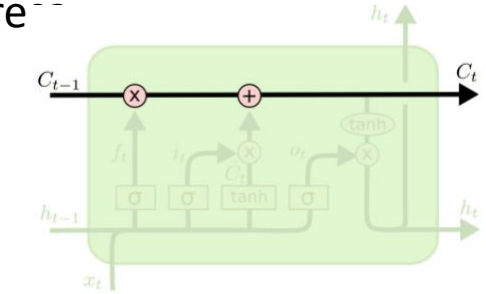
Red neuronal recurrente con unidades LSTM



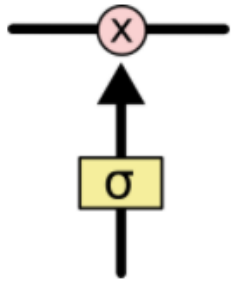


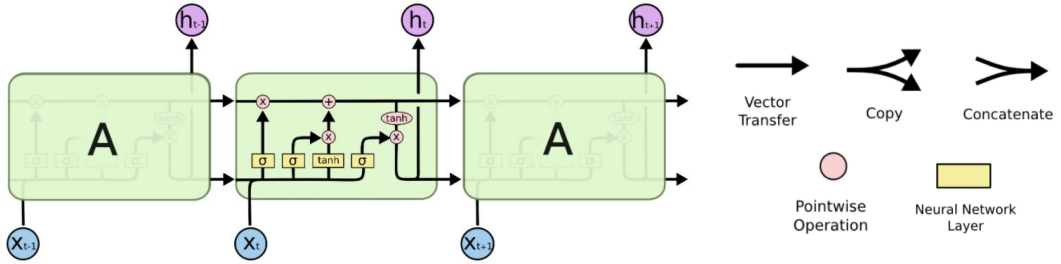
Unidades Long-Short Term Memory

- La **clave** de la **unidad LSTM** es la celda de estado (***cell state***), el camino recto superior por el que **fluyen los datos** a lo largo de los instantes **secuenciales** y van siendo alterados según intere



- El resto de unidad LSTM tiene la capacidad de **eliminar o añadir información sobre este camino** a partir de unas estructuras denominadas **compuertas gestionadas por una capa sigmoide [0-1]**

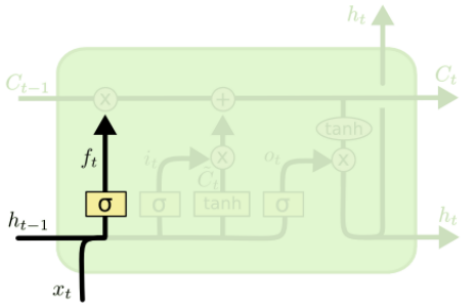




Unidades Long-Short Term Memory

- El primer paso dentro de la celda LSTM es decidir que **información** del ***cell state* quiero preservar** para el **nuevo estado**. Esto se lleva a cabo mediante la compuerta de olvido (***forget gate layer***)

La entrada a la capa es la salida del instante anterior y la nueva entrada

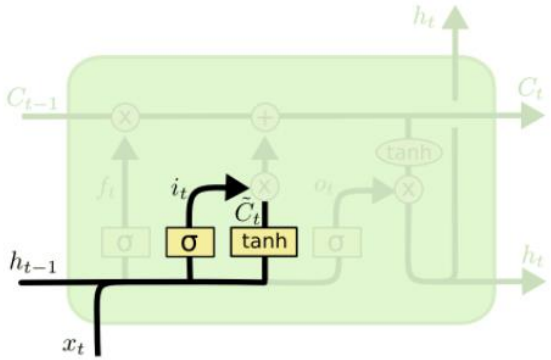


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

La salida (f_t) es un valor entre [0,1] para cada número del Cell State C_{t-1}

Unidades Long-Short Term Memory

- El siguiente paso consiste en decidir que nueva información va a hacer modificar el ***cell state*** del estado anterior. Este paso se divide en tres partes:
  - La compuerta de entrada (***input gate layer***) decide que valores actualizará $\boldsymbol{i_t}$
  - Una capa activada con ***tanh*** crea un vector de nuevos candidatos $\boldsymbol{\widetilde{C}_t}$
  - Se **combinan** ambos para para **actualizar** el ***cell state***



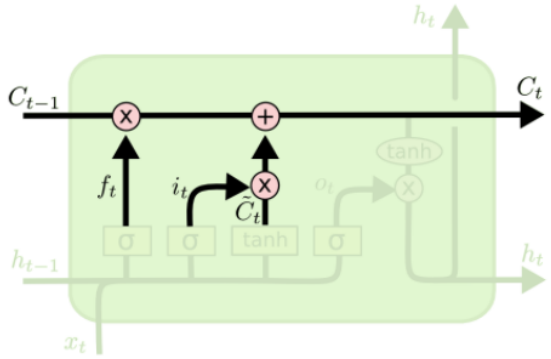
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Unidades Long-Short Term Memory

- Ahora es momento de actualizar el ***cell state*** de la unidad anterior ($\boldsymbol{C_{t-1}}$) según lo que marcan la ***forget gate layer*** ($\boldsymbol{f_t}$) , la ***input gate layer*** ($\boldsymbol{i_t}$) y el vector $\boldsymbol{\widetilde{C}_t}$ de nuevos candidatos, dando lugar al nuevo ***cell state*** $\boldsymbol{C_t}$:

  - Se multiplica el estado viejo $\boldsymbol{C_{t-1}}$* $\boldsymbol{f_t}$ eliminando lo que no interesa del estado anterior

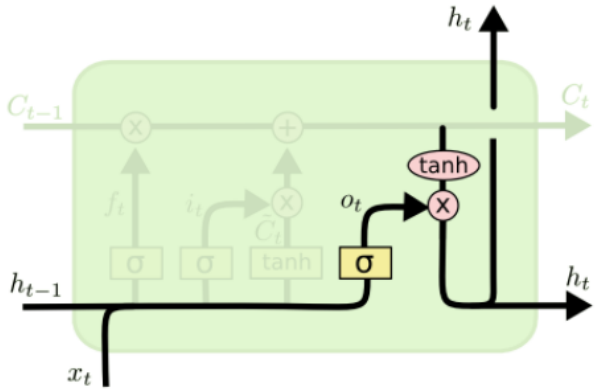
  - Se suma al $\boldsymbol{C_{t-1}}$ la nueva información ($\boldsymbol{i_t}$* $\boldsymbol{\widetilde{C}_t}$) dada por los nuevos valores candidatos escalados por el peso que tendrán en el nuevo estado $\boldsymbol{C_t}$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Unidades Long-Short Term Memory

- Finalmente, debemos decidir cuál va a ser la salida ($\boldsymbol{h_t}$) del estado actual. Esta salida será nuestro nuevo ***cell state*** ($\boldsymbol{C_t}$) pero en versión filtrada
- Concretamente, mediante una compuerta de salida o ***output gate layer*** se regulan las partes del ***cell state*** ($\boldsymbol{C_t}$) que se quieren sacar como salida
- Mediante una ***tanh*** se normalizan los **valores de $\boldsymbol{C_t}$** al rango **[-1,1]** y se lleva a cabo el **producto** por $\boldsymbol{o_t}$ que define la **salida final** de la celda ($\boldsymbol{h_t}$)



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTMs en Keras: ¿Qué necesito saber?

- Necesitaré entrenar una **capa** de **Embedding** según el **problema a resolver**. También se pueden **reentrenar embeddings** más generalistas (**word2Vec**) a partir del conocimiento sobre **grandes Corpus** (similar a **transfer learning** en imágenes)
- Instanciar **capas LSTM**, escogiendo el **número de unidades** (hiperparámetro)

`Embedding` class

```
tf.keras.layers.Embedding(
    input_dim,
    output_dim,
    embeddings_initializer="uniform",
    embeddings_regularizer=None,
    activity_regularizer=None,
    embeddings_constraint=None,
    mask_zero=False,
    input_length=None,
    **kwargs
)
```

`LSTM` class

```
tf.keras.layers.LSTM(
    units,
    activation="tanh",
    recurrent_activation="sigmoid",
    use_bias=True,
    kernel_initializer="glorot_uniform",
    recurrent_initializer="orthogonal",
    bias_initializer="zeros",
    unit_forget_bias=True,
    kernel_regularizer=None,
    recurrent_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    recurrent_constraint=None,
    bias_constraint=None,
    dropout=0.0,
    recurrent_dropout=0.0,
    return_sequences=False,
    return_state=False,
    go_backwards=False,
    stateful=False,
    time_major=False,
    unroll=False,
    **kwargs
)
```

Contenidos

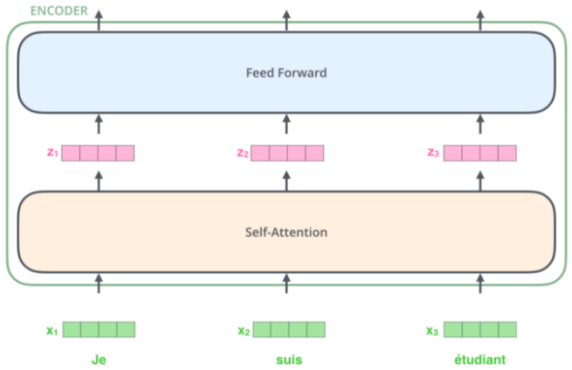
1. Procesamiento del Lenguaje Natural (NLP)
2. De texto a representaciones numéricas
3. Word2Vec
4. Redes neuronales recurrentes
5. **Introducción a *transformers***

Introducción a transformers

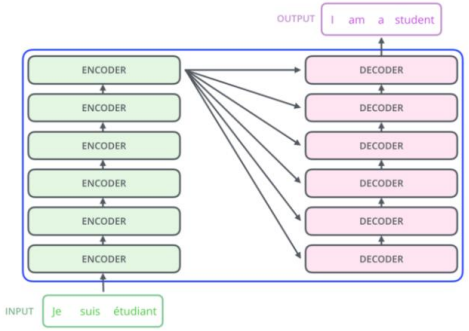
- A pesar de que las **LSTMs** han sido **muy exitosas** en la mayoría de **tareas** que envuelven el **NLP** poseen algunas desventajas
- Su ***performance*** **decrece** conforme **aumenta** la **longitud** de la **oración**. La probabilidad de mantener el contexto de una palabra lejana para predecir la actual, decrece exponencialmente con la distancia a ella
- Es **difícil** de **paralelizar** el **entrenamiento** de una LSTM ya que procesan secuencialmente las palabras y la salida de una celda la necesita la siguiente

Introducción a transformers

- Recientemente se presenta una nueva arquitectura capaz de procesar todas las palabras en paralelo y relacionarlas entre sí mediante un **módulo de atención**



- Se compone por **seis *encoders*** y **seis *decoders***



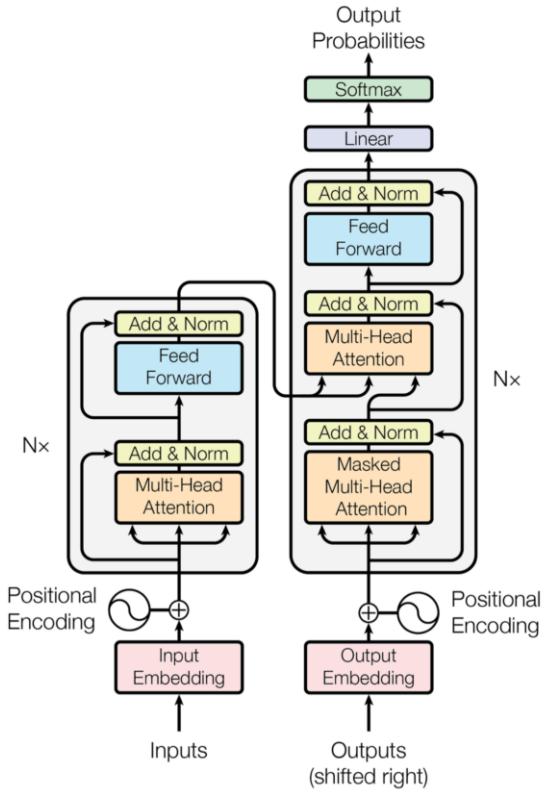


Figure 1: The Transformer - model architecture.

07MIAR – Redes Neuronales y Deep Learning

Subtítulo presentación