Introducción al Procesamiento de Lenguaje Natural

Grupo de PLN – InCo

Redes Neuronales

Métodos de Clasificación

Naïve Bayes

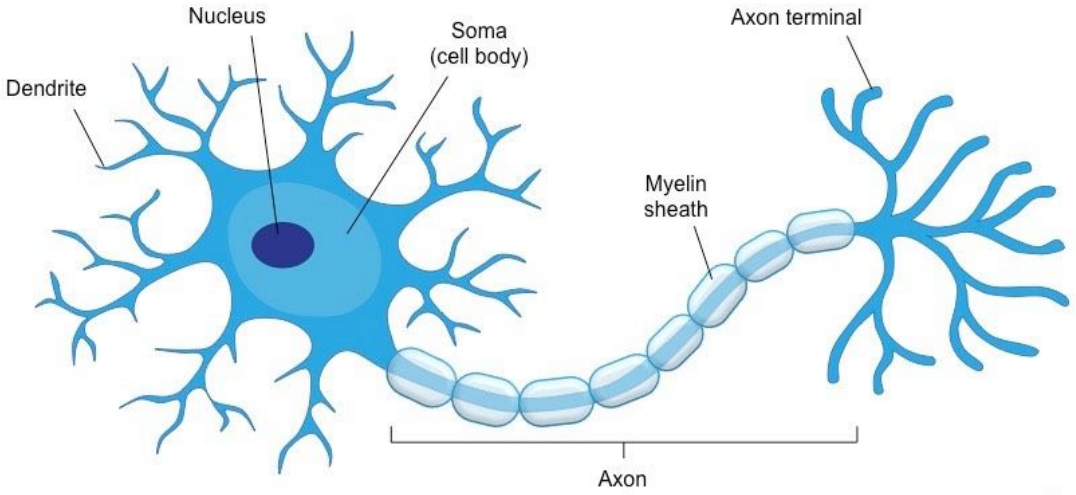
Árboles de Decisión

Regresión Logística

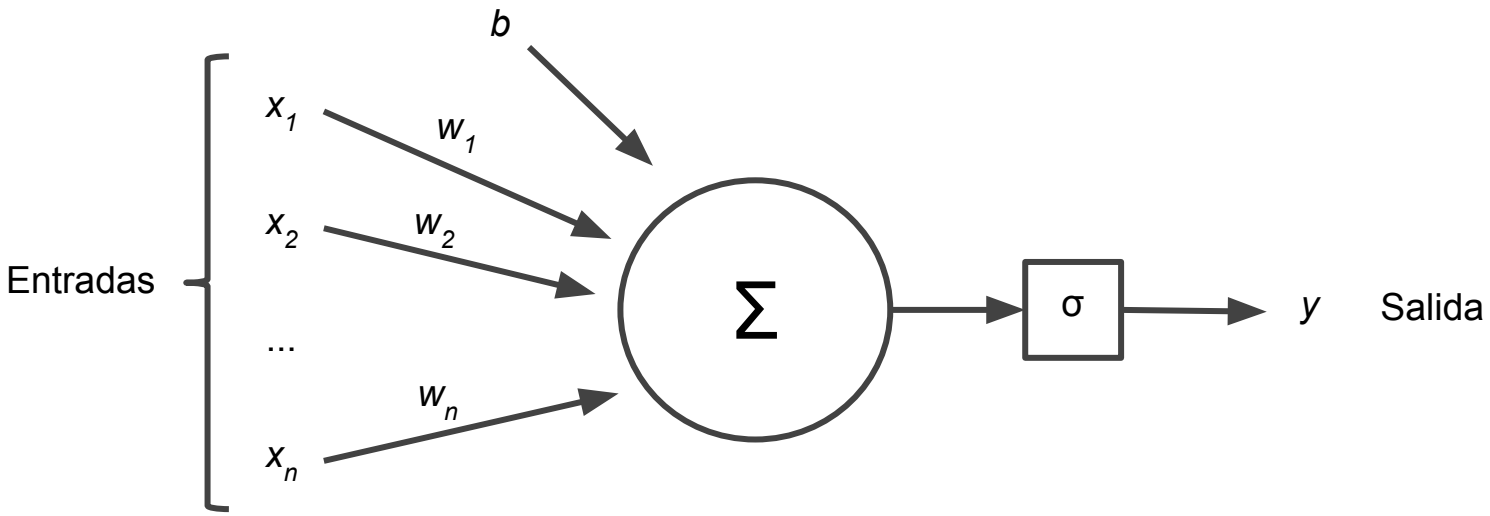
SVM

Redes Neuronales

Redes Neuronales



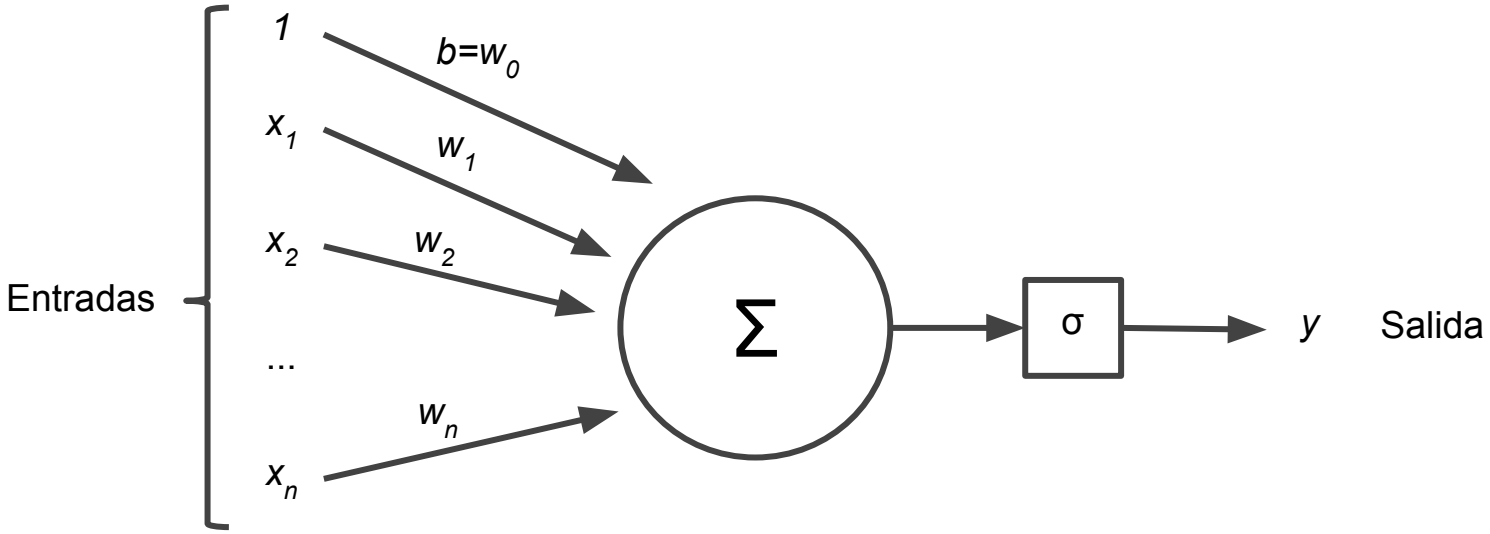
Redes Neuronales



$$y = \sigma(\sum_i x_i w_i + b)$$

Neurona de McCulloch-Pitts, 1943

Redes Neuronales



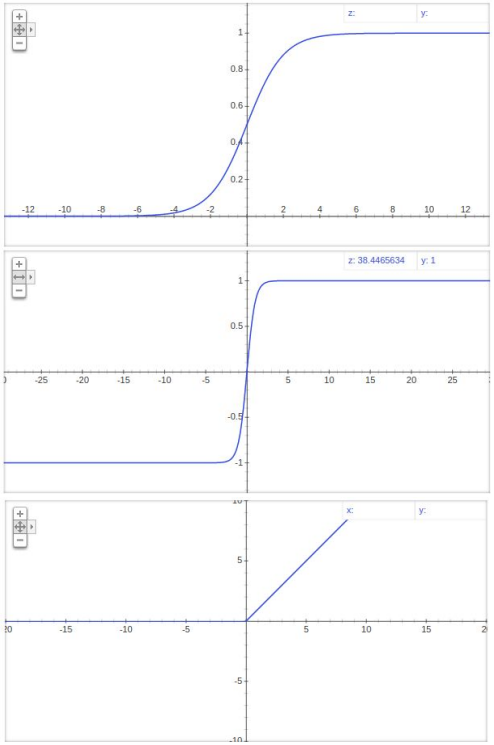
$$\hat{x} = [1, x_1, x_2, ..., x_n]$$
$$\hat{w} = [w_0, w_1, w_2, ..., w_n]$$

$$y = \sigma(\hat{x}.\hat{w})$$

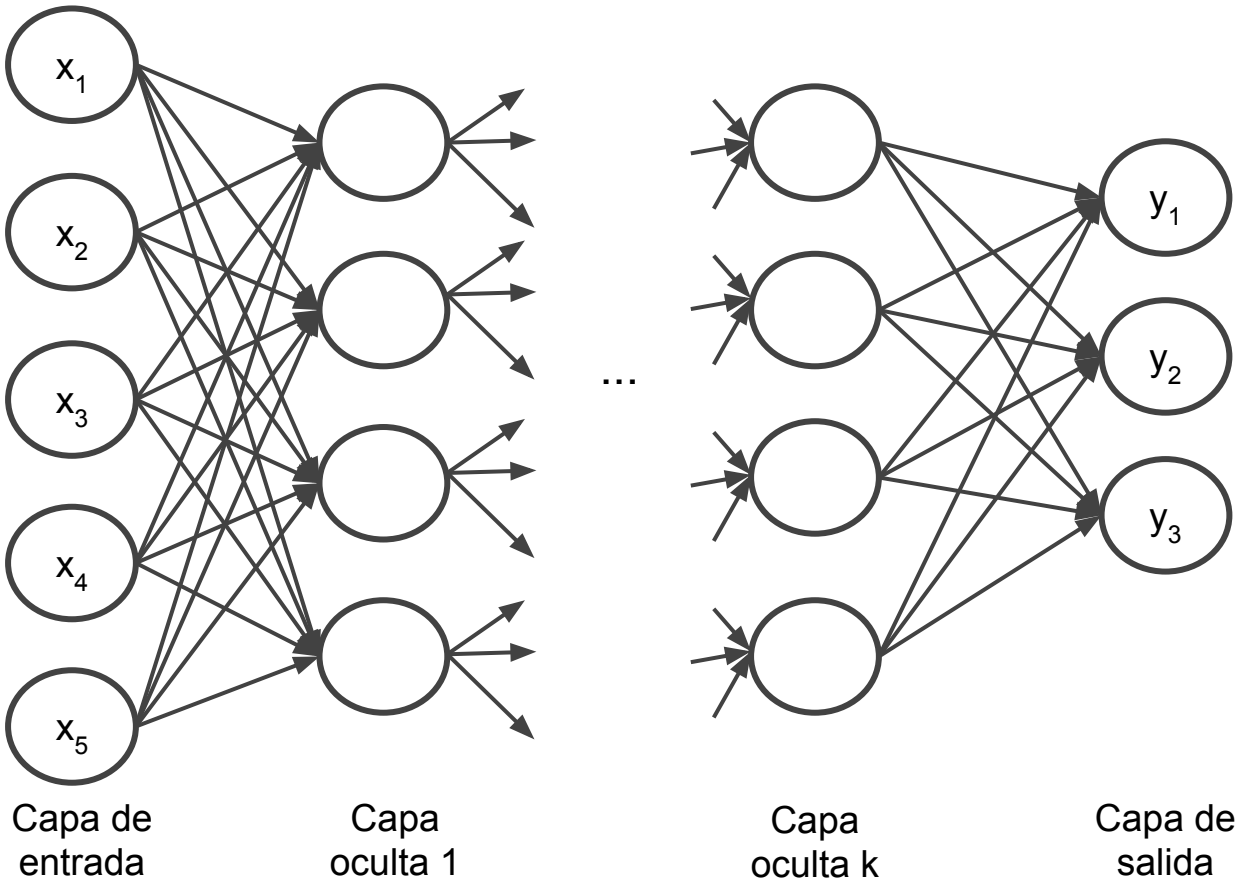
Redes Neuronales

Funciones de activación

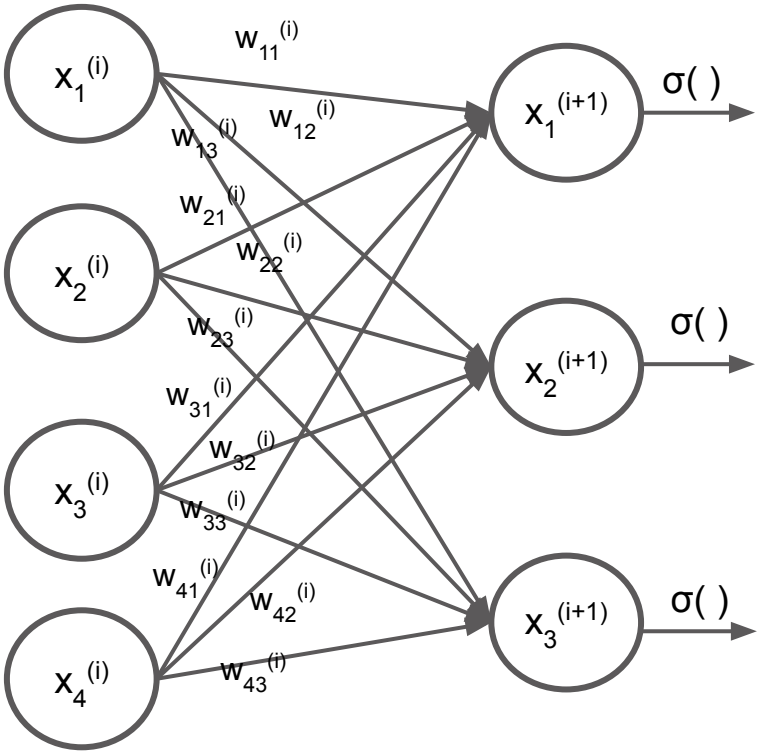
- Función sigmoide o logística: $$\sigma(z) = \frac{1}{1 + e^{-z}}$$
- Tangente hiperbólica: $$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$
- ReLU: $$\mathrm{relu}(z) = max(0, z)$$
- Otras...



Redes Neuronales



Redes Neuronales



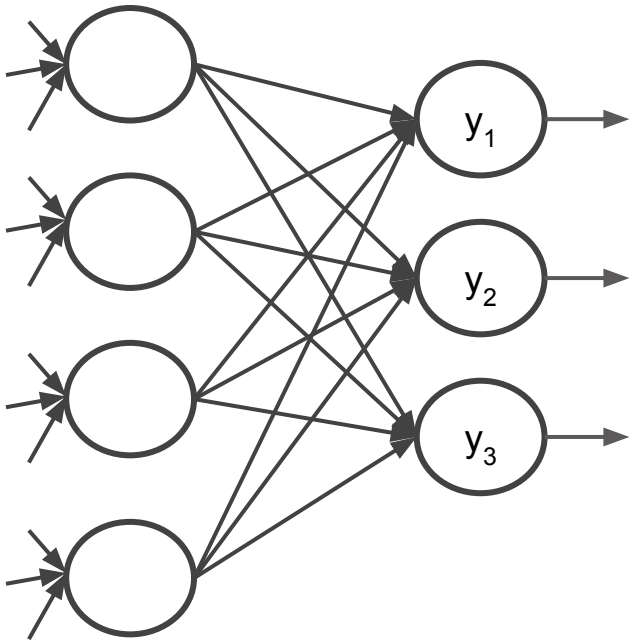
Entrada: $x^{(i)} = [\, x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)} \,]$

Salida: $x^{(i+1)} = [\, x_1^{(i+1)}, x_2^{(i+1)}, x_3^{(i+1)} \,]$

$$W^{(i)} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix}$$

$$x^{(i+1)} = \sigma\left(x^{(i)} W^{(i)}\right)$$

Redes Neuronales



Última capa oculta

Capa softmax

Problemas de clasificación discretos, queremos que la salida sea una distribución de probabilidad

Se suele utilizar una capa softmax

$$P(j|x) = \frac{e^{y_j}}{\sum_k e^{y_k}}$$

Entrenamiento

Funciones de pérdida

N instancias x_i de dimensión M

Valores esperados y_i

Valores predichos $\hat{y}_i$ (resultado de la red)

- Error Cuadrático Medio:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

- Entropía Cruzada:

$$CE = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(\hat{y}_{ij})$$

- Para valores categóricos:

$$CE = -\frac{1}{N}\sum_{i=1, y_j=1}^{N} log(\hat{y}_{ij})$$

Entrenamiento

Se trata de encontrar los pesos que minimicen la función de pérdida

- Descenso por gradiente
- Estocástico
- Backpropagation

¿Puedo encontrar la mejor función?

- Óptimos locales
- Sobreajuste

Vectores de palabras

- En PLN trabajamos principalmente con **texto**
- Las RN y la mayoría de los clasificadores utilizan valores numéricos como entrada, por lo que necesitamos una **representación numérica** de textos
  - palabras
  - oraciones
  - documentos
- Es deseable que esta representación numérica tenga propiedades explotables (e.g. un resultado de *distancia* interpretable)

Hipótesis distribucional

En los 1950s surge la hipótesis distribucional (Firth):

Palabras que aparecen en contextos similares tienden a tener significados similares

La milanesa con queso más rica es la uruguaya.

Sí, es re rica la hamburguesa con queso de ese lugar.

A la milanesa con queso mozzarella y salsa le decimos napolitana.

El otoño es una de las estaciones del año.

¡El verano es una de mis estaciones favoritas!

En invierno hace pila de frío.

En verano nunca hace frío.

Matriz término-Término

Representa las palabras contando las palabras que las rodean, según un **contexto**.

El **contexto** puede ser el documento entero (archivo, tweet, página web o lo que sea)
pero lo más común es tomar **N palabras de ventana**.

O sea, si *X* es la palabra a modelar: $\text{palabra}_{-N} \ldots \text{palabra}_{-2}\text{palabra}_{-1}\ X\ \text{palabra}_{1}\text{palabra}_{2} \ldots \text{palabra}_{N}$

¿Cómo quedaría la matriz con el ejemplo anterior y usando N=5?

La milanesa con queso más rica es la uruguaya.

Sí, es re rica la hamburguesa con queso de ese lugar.

A la milanesa con queso mozzarella y salsa le decimos napolitana.

El otoño es una de las estaciones del año.

¡El verano es una de mis estaciones favoritas!

En invierno hace pila de frío.

En verano nunca hace frío.

	...	rica	queso	frío	estaciones	...
...						
milanesa		1	2	0	0	
hamburguesa		1	1	0	0	
otoño		0	0	0	1	
verano		0	0	1	1	
invierno		0	0	1	0	
...						

PROBLEMA → **los vectores son enormes y con muchos ceros (*sparse*)**

Word2Vec

En 2013 Mikolov et al. propusieron **word2vec**, un par de algoritmos para crear colecciones de vectores de palabras **densos** (con pocos 0s) y de baja dimensionalidad (típicamente entre 150 o 300).

Idea: en vez de contar las palabras en una ventana de contexto, entrenemos un clasificador que prediga qué tan probable es que la palabra **c** aparezca en el contexto de **w**. O sea: $P(+|w,c)$

Como queremos que las palabras más relacionadas tengan vectores cercanos y los de las menos relacionadas estén más lejos necesitamos **ejemplos negativos**.

Técnica de ***negative sampling***: elegir palabras que no compartan contexto con **w**. Por cada ejemplo positivo (**w**,$\mathbf{c}_{pos}$) tomamos *k* ejemplos negativos (**w**,$\mathbf{c}_{neg}$).

Word2Vec: Algoritmo skip-gram

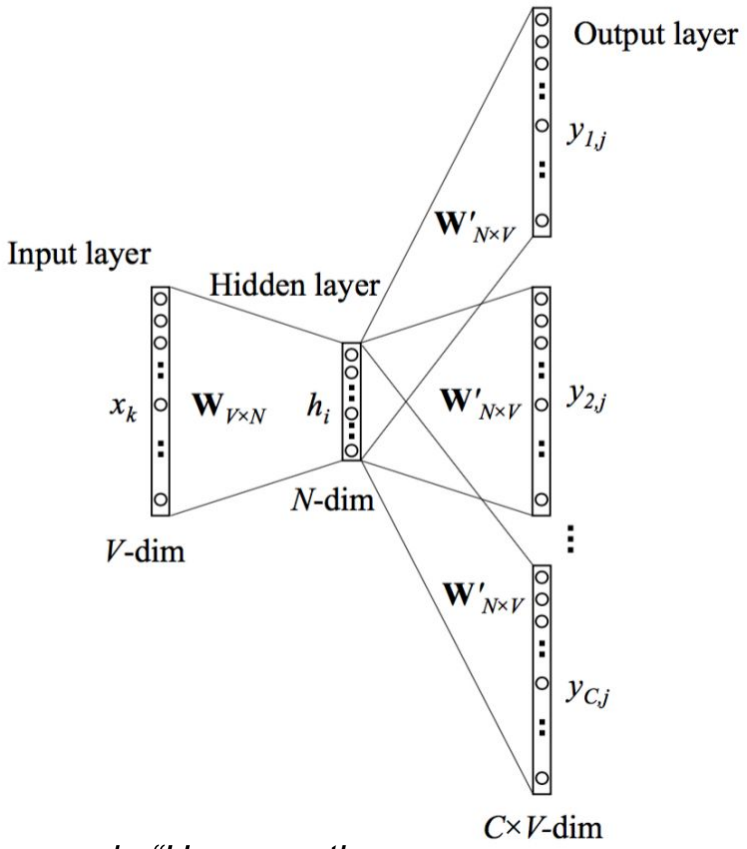


Imagen de "How exactly does word2vec work?" (Meyer, 2016)

skip-gram intenta modelar las palabras más probables que aparecerán alrededor de una palabra

- **Entrada:** Codificación 1-hot de la palabra k
- **Salidas:** Probabilidad (activación sigmoide) de que la palabra j esté en el contexto C alrededor de la palabra k

Los *word embeddings* son el estado de la capa oculta luego del entrenamiento

Word2Vec

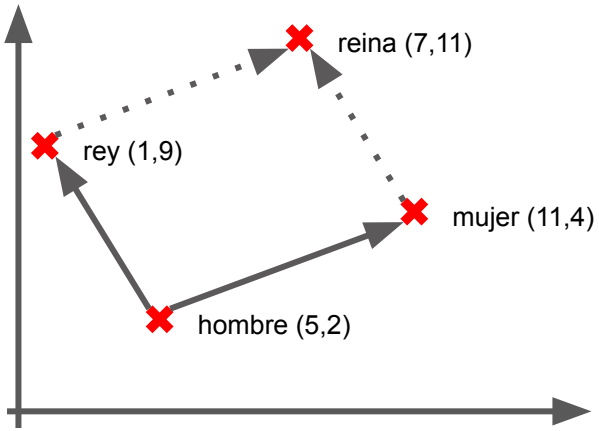
Se asocia una palabra (string) a un vector de reales

Vectores más cercanos tienden a ser semánticamente similares (similaridad coseno)

“Descubre” relaciones entre palabras

rey – hombre + mujer ≅ reina

uruguay – montevideo + francia ≅ parís

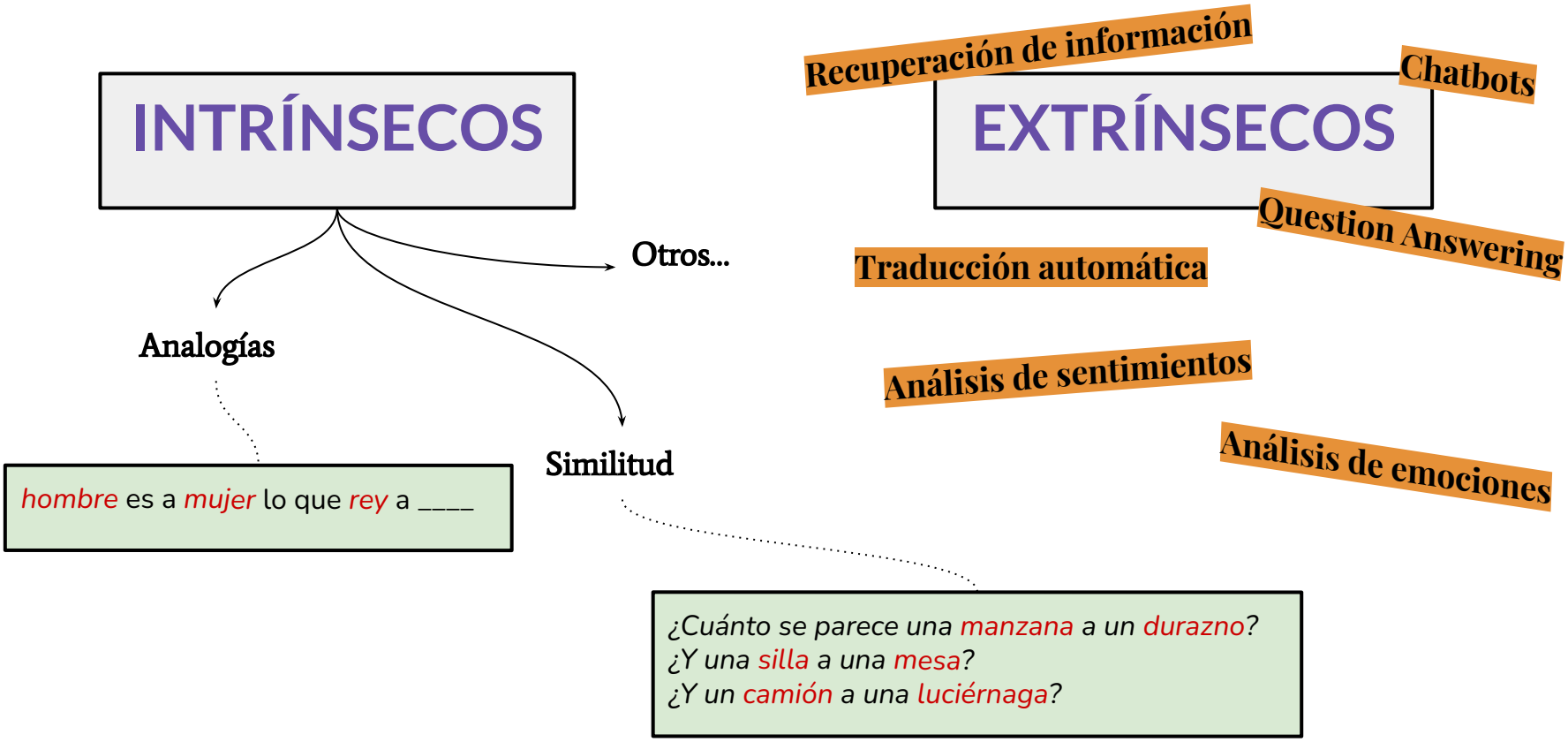


Se considera palabra a nivel de *string*, por lo que “vela” 🕯️ y “vela” ⛵ van a estar representadas por el mismo vector

PROBLEMA → no hay distinción entre diferentes significados de una palabra

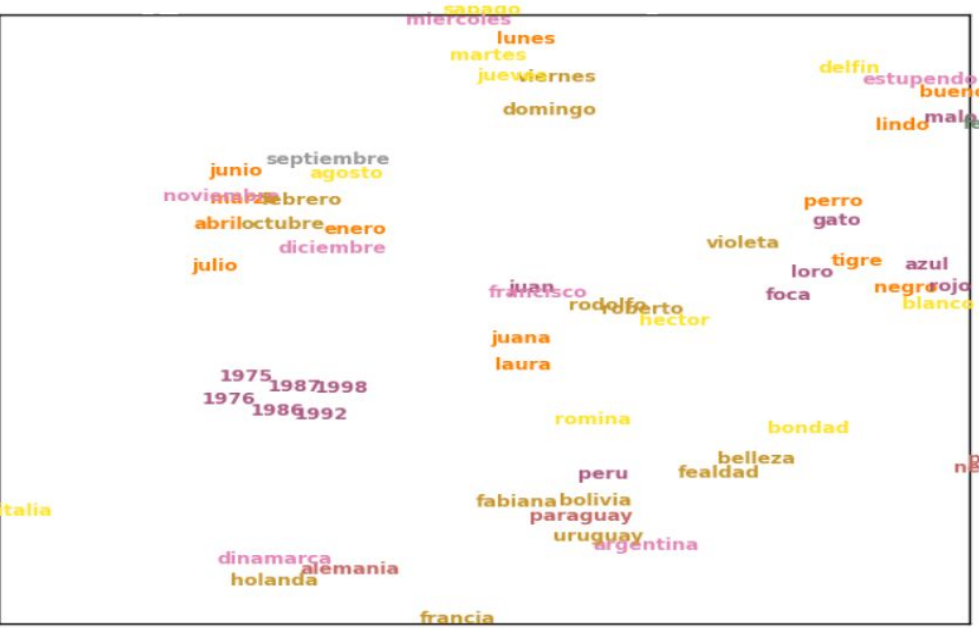
Evaluación

¿Cómo sabemos si una colección de embeddings está bien?

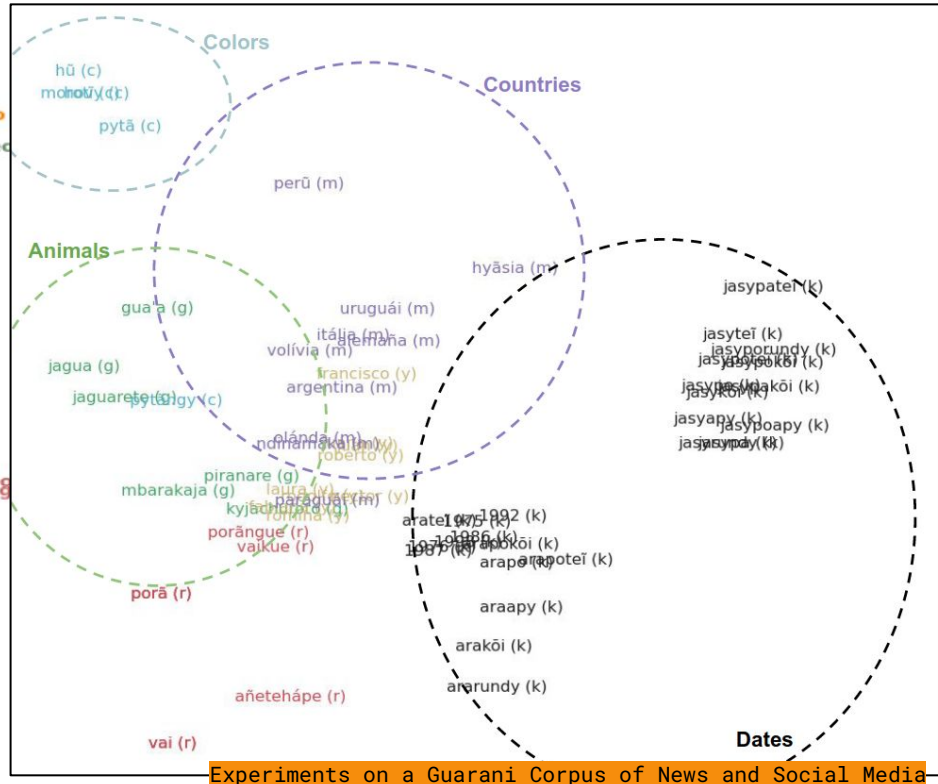


Evaluación

Visualización: reducir la dimensionalidad del espacio vectorial y graficar algunas palabras para ver cómo se agrupan



Spanish Word Vectors from Wikipedia (Etcheverry & Wonsever, LREC 2016)



Experiments on a Guarani Corpus of News and Social Media (Góngora, Giossa & Chiruzzo, AmericasNLP 2021)

Aprendizaje Profundo

Con word embeddings podemos…

- Usar arquitecturas de redes neuronales más complejas
  - CNN
  - LSTM
  - Transformers
- Usarlos como features en métodos de aprendizaje clásicos
  - Centroide

Redes Convolutivas

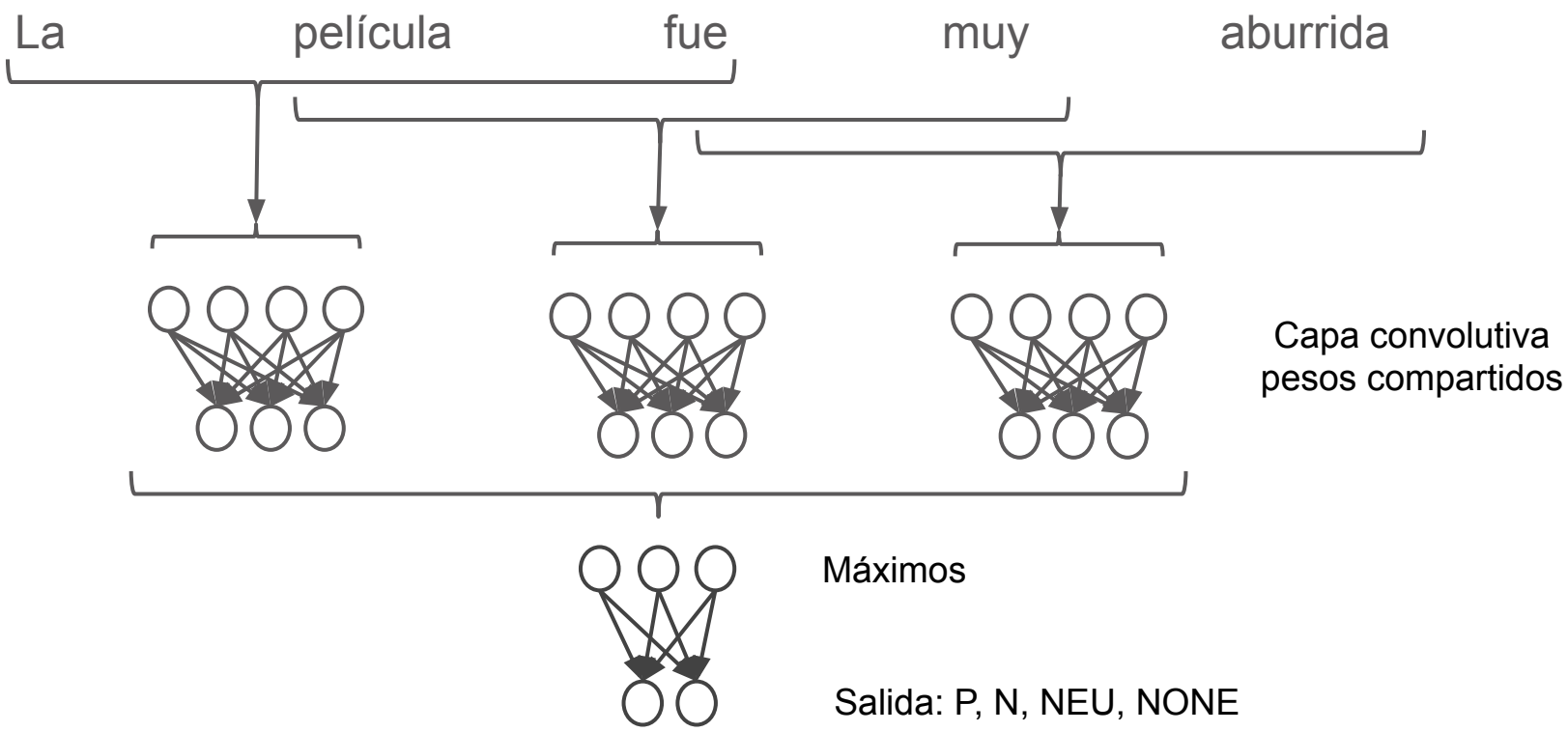
Las redes convolutivas (CNN) toman como entrada secuencias

Aplican la misma transformación a todas las subsecuencias de cierto largo

También pueden aplicarse a subsecuencias de distinto largo

Devuelven una salida que combina todas las salidas intermedias

Redes Convolutivas



Redes LSTM

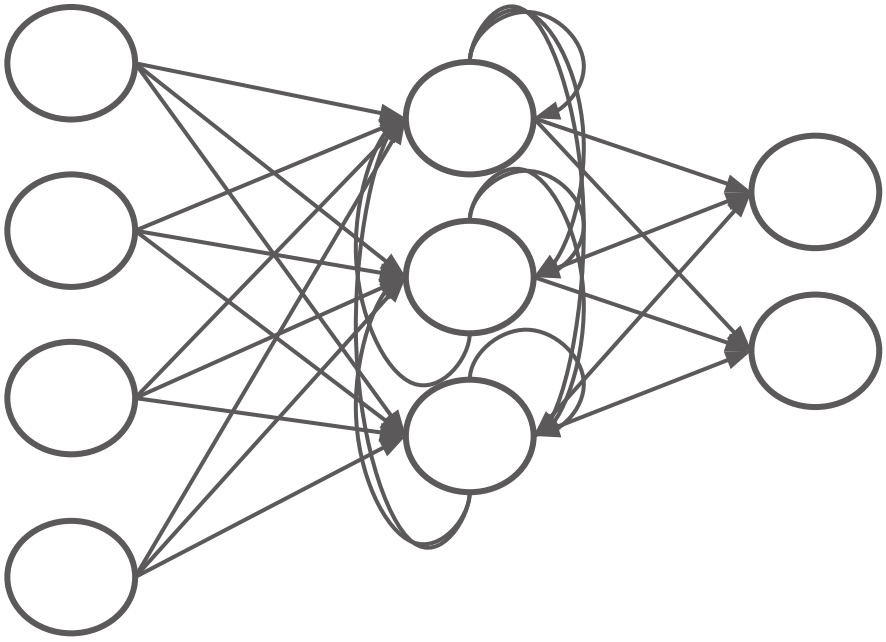
Las redes *long short term memory* (LSTM) toman como entrada secuencias

Tienen una o más capas recurrentes (loop de la salida a la entrada)

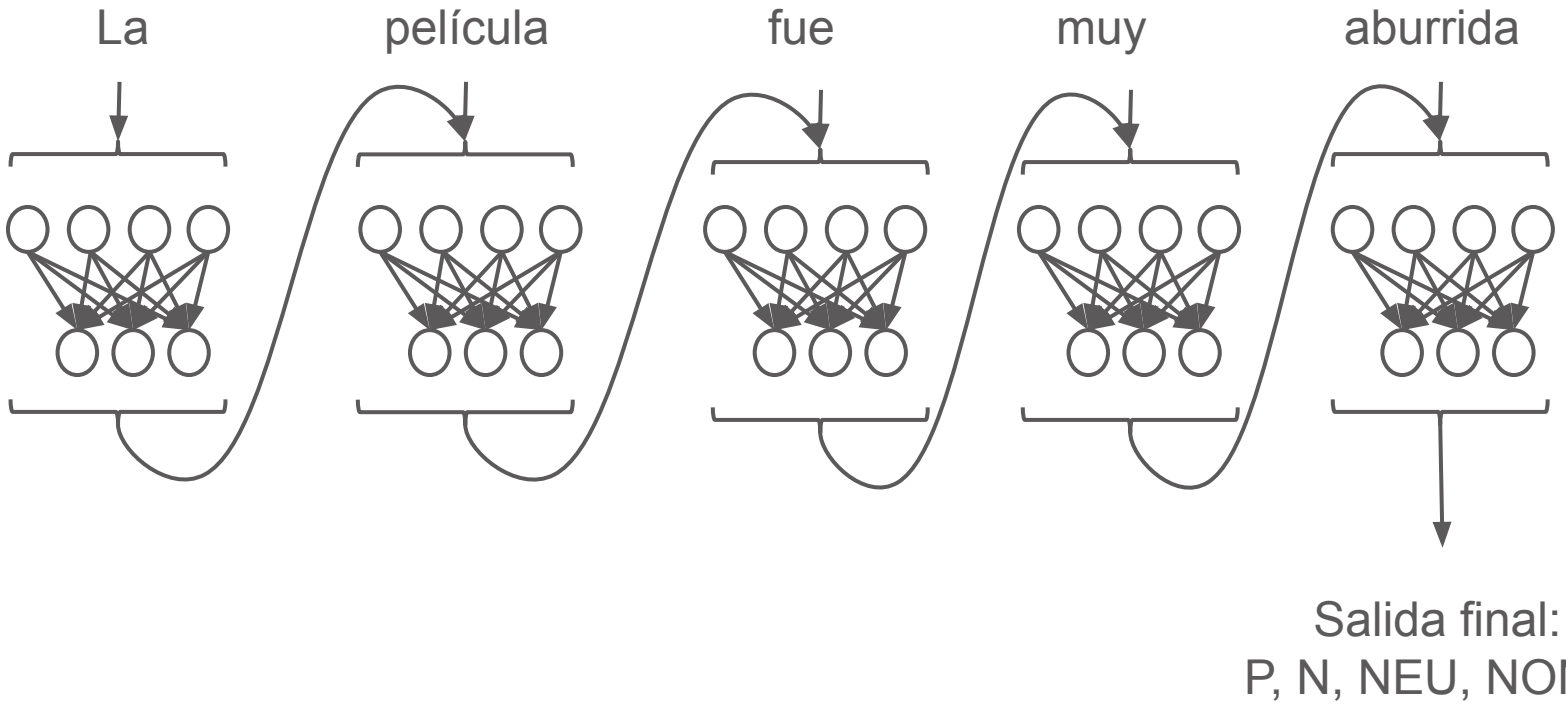
Usan un tipo de neurona especial que permite inhibir o habilitar salidas

Pueden devolver una salida por elemento o una sola salida final

Redes LSTM



Redes LSTM



Transformers

Los transformers toman secuencias como entrada

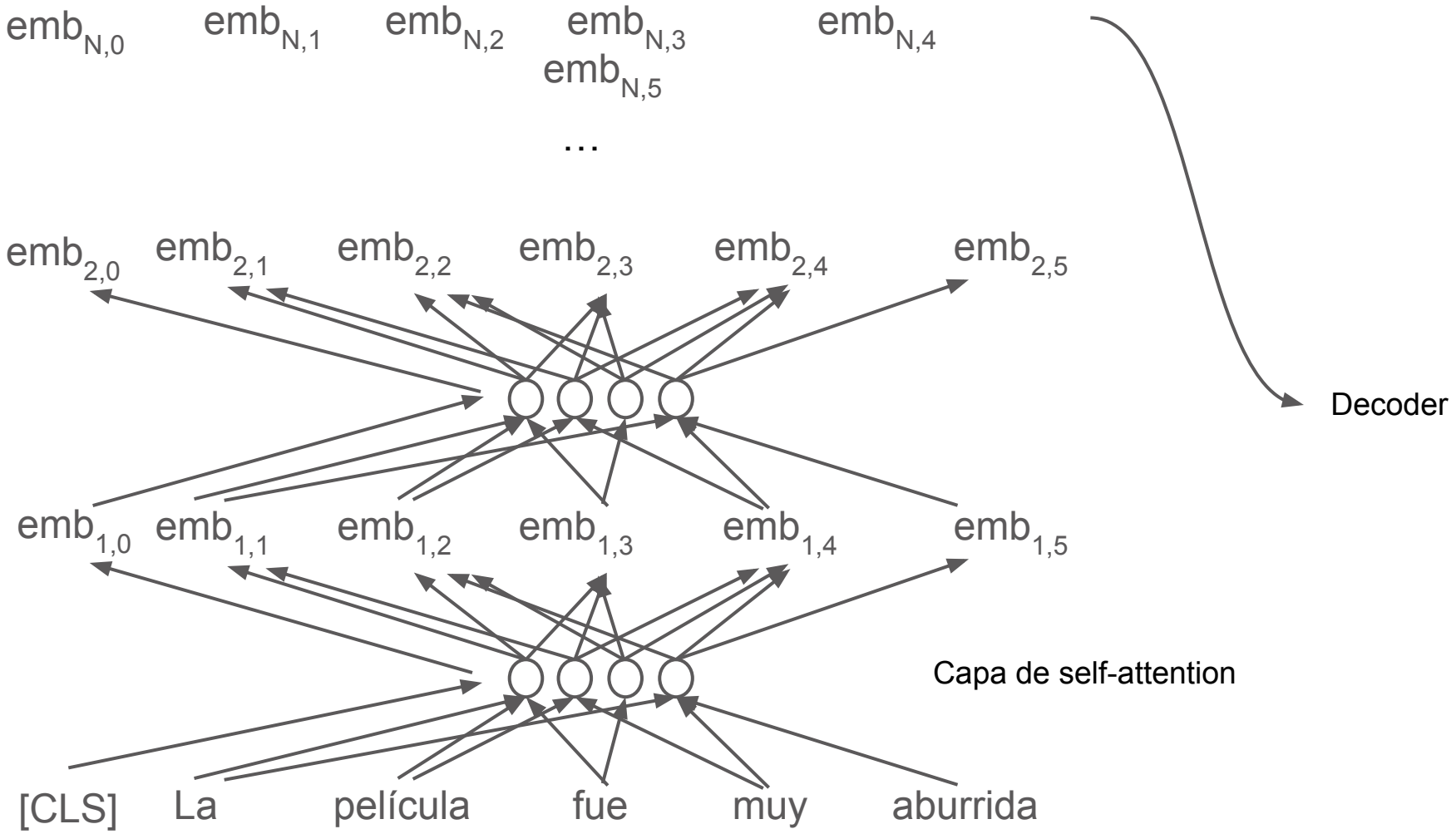
Se componen de dos redes: codificador / decodificador

Capas auto-atencionales

Embeddings contextuales para cada palabra

Un embedding total de la oración

Transformers



Software

Tensorflow - https://www.tensorflow.org/

Keras - http://keras.io/

PyTorch - https://pytorch.org/

Huggingface - https://huggingface.co/