# Representaciones vectoriales de palabras

Ximena Gutierrez-Vasques

Víctor Mijangos

RIIA, 2019

# Los retos de procesar el lenguaje humano

**Gran reto:**

- Modelar el lenguaje humano  desde una perspectiva computacional.
- El lenguaje es ambiguo y complejo

El texto ya no es interpretado como una mera  cadena alfanumérica. Se procesa con conocimiento **lingüístico**

Área **interdisciplinaria**

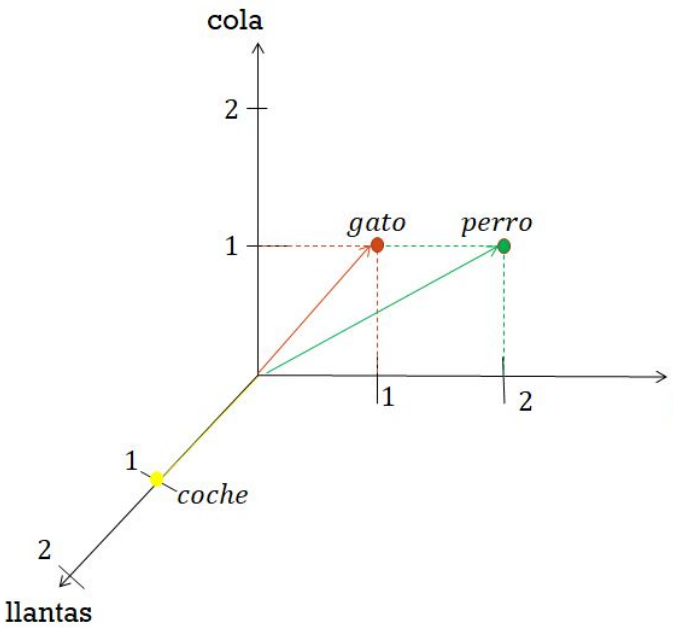
# Modelos de espacio vectorial

Se ha propuesto representar palabras, oraciones o documentos en **espacios vectoriales**

En esta representación geométrica, **palabra similares** deben estar **cercanos**

Por su parte, **palabra distintas** deben estar **alejadas** entre sí

# ¿Cómo representamos al lenguaje en un espacio vectorial?




Sistema de coordenadas:

$$gato = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

$$perro = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$$

$$coche = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

*Ximena Gutiérrez, Víctor Mijangos, 2019*

# Espacios semánticos distribucionales

Unidades lingüísticas con distribuciones similares tienen significados similares (Weaver, 1955; Firth, 1968; Landauer and Dumais, 1997)

¿Me regalas un wapimunk de agua?

El wapimunk de vidrio se rompió

Sirve la leche en el wapimunk

**¿Qué significa wapimunk?**

# Espacios semánticos distribucionales

| | croqueta | mascota | fruta | potasio | sabor | amarillo |
|---|---|---|---|---|---|---|
| gato | 10 | 20 | 0 | 0 | 0 | 9 |
| perro | 16 | 15 | 1 | 0 | 5 | 1 |
| plátano | 0 | 0 | 19 | 28 | 20 | 5 |

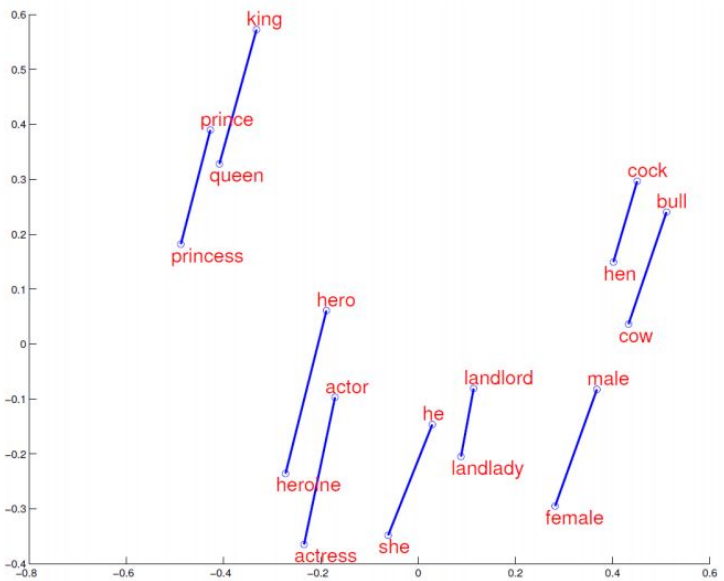
- El significado de una palabra se representa mediante un **vector** que codifica el patrón de **co-ocurrencias** de esa palabra con otras. Para generar estas representaciones se necesitan corpus lingüísticos, de gran tamaño, de una lengua en específico.

# Representaciones distribuidas

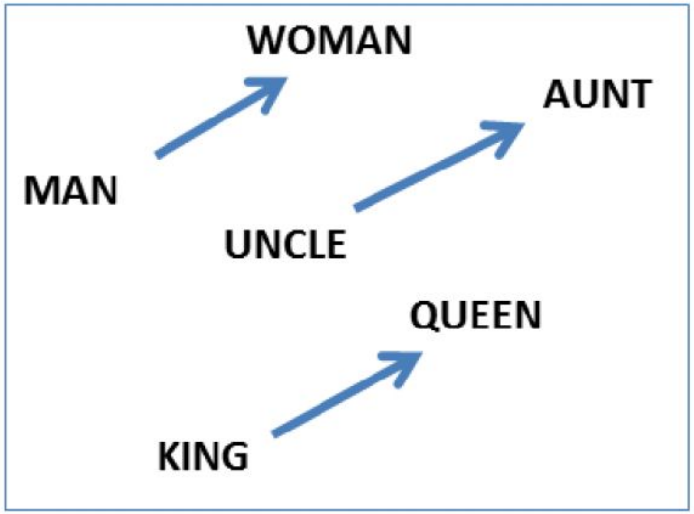
Los vectores de palabras que comparten contextos, estarán cercanos en el espacio vectorial y mostrarán propiedades interesantes.

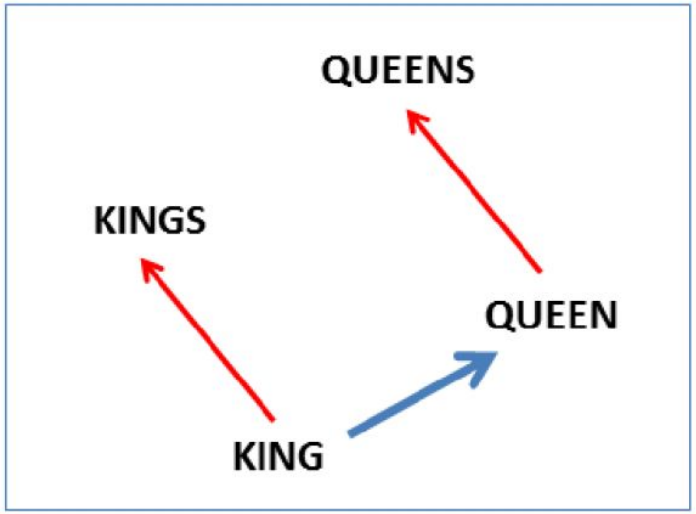
## Propiedades lineales

La diferencia entre vectores de palabras semánticamente relacionadas por género masculino/femenino, es constante en todos los casos




*Ximena Gutiérrez, Víctor Mijangos, 2019*




constant **male-female** difference vector

constant **singular-plural** difference vector

- Vector operations are supported and make intuitive sense:

$$w_{king} - w_{man} + w_{woman} \cong w_{queen}$$

$$w_{einstein} - w_{scientist} + w_{painter} \cong w_{picasso}$$

$$w_{paris} - w_{france} + w_{italy} \cong w_{rome}$$

$$w_{his} - w_{he} + w_{she} \cong w_{her}$$

$$w_{windows} - w_{microsoft} + w_{google} \cong w_{android}$$

$$w_{cu} - w_{copper} + w_{gold} \cong w_{au}$$

*Ximena Gutiérrez, Víctor Mijangos, 2019*

# Algoritmo de Word2Vec

# Representaciones aprendidas a través de redes neuronales

- Representaciones distribuidas, **word embeddings**
- **Word2Vec (~2013)**

$p(w_t|contexto)$ **La probabilidad de encontrar una palabra en una lengua, dado cierto contexto**

**En general, el contexto es una ventana de k palabras cercanas**

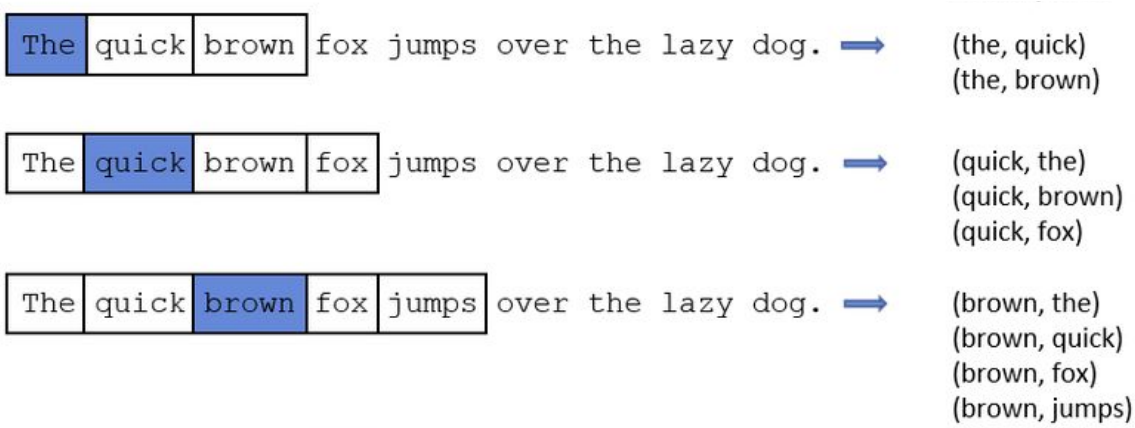
$p(w|w_{t-k}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k})$ *(a.k.a modelo del lenguaje)*

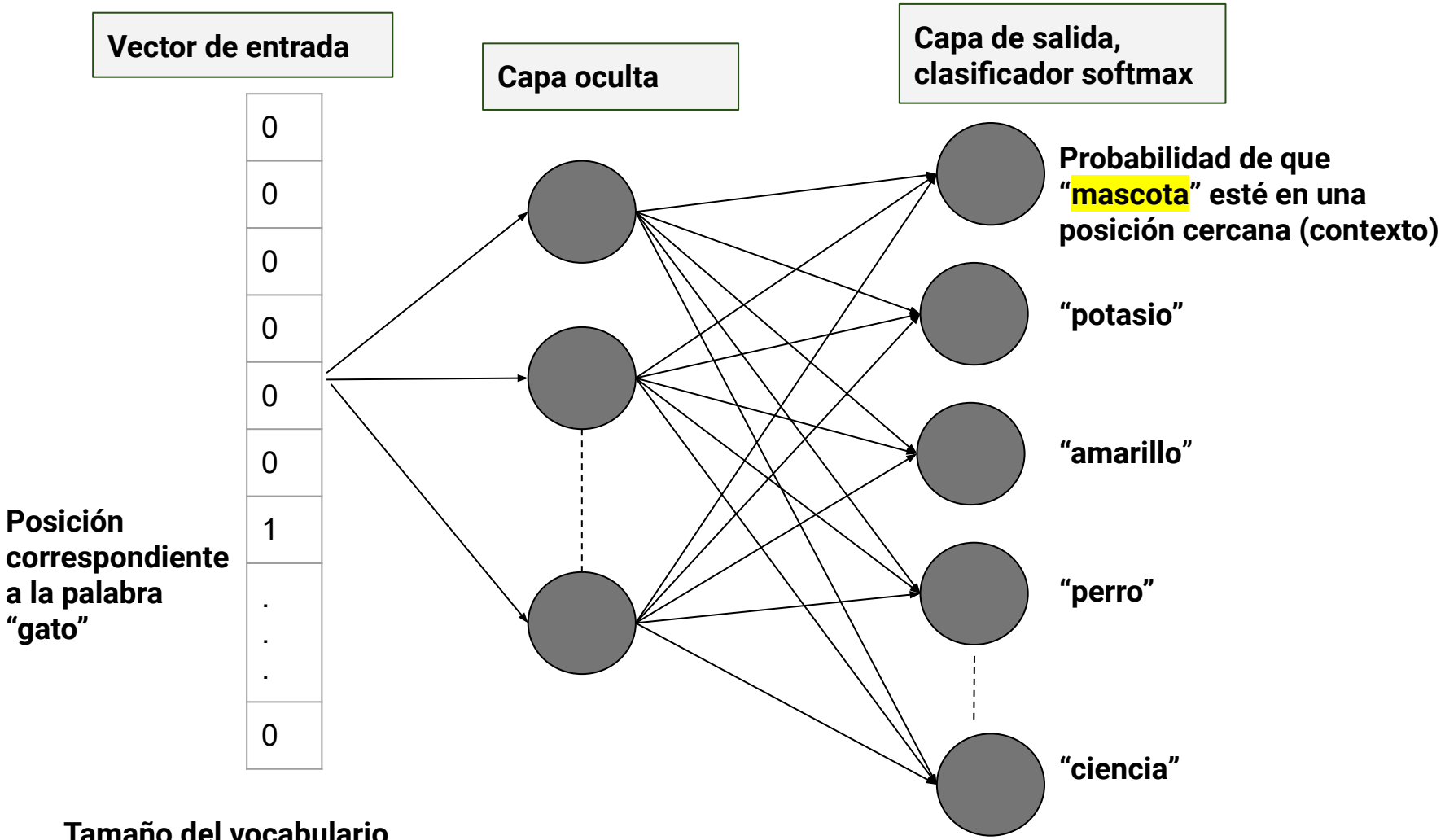
| | | | | |
|---|---|---|---|---|
| Hola | ¿ | cómo | estás | ? |

# Representaciones aprendidas a través de redes neuronales

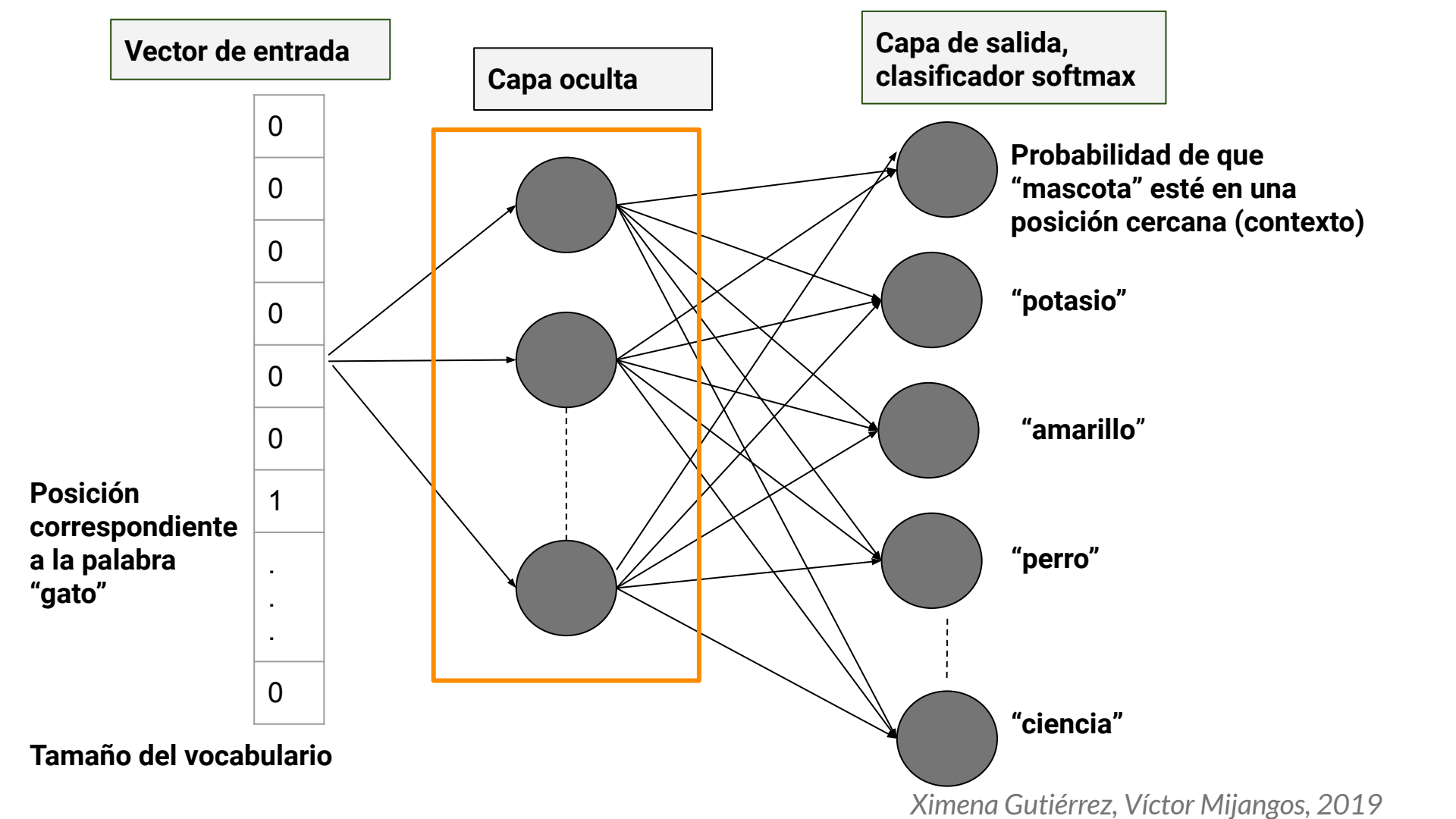
- Word2Vec (Cbow): Dado un contexto, predecir la palabra
- Word2Vec (Skip-gram): Dada una palabra, predecir el contexto

Vamos a entrenar a la red con pares de palabras extraídas de un gran corpus:





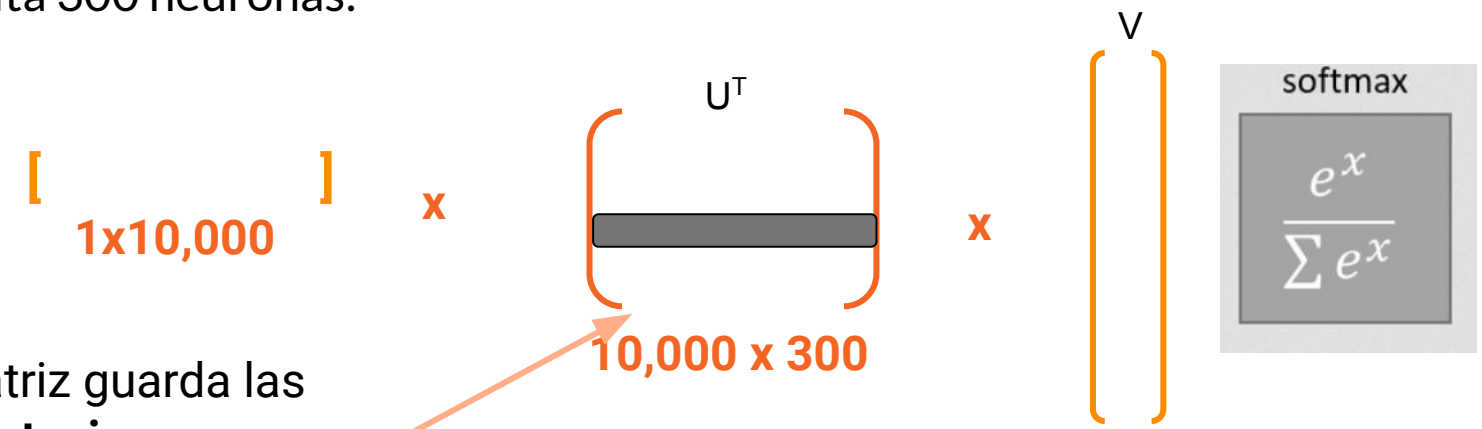

*Ximena Gutiérrez, Víctor Mijangos, 2019*




*Ximena Gutiérrez, Víctor Mijangos, 2019*

# La capa oculta

La podemos ver como una **matriz de pesos** (que se va ajustando durante el entrenamiento) Por ejemplo, si el vocabulario tiene 10,000 palabras y la capa oculta 300 neuronas:




Esta matriz guarda las **representaciones** que nos interesan para cada una de las **palabras d**el vocabulario

# Notación

- Utilizaremos la notación ***U*** para referir a la matriz en la capa oculta (matriz de embeddings).
- La matriz de pesos de salida la llamaremos ***V*** (matriz de Nxd).
- Debe tomarse en cuenta qué representan los renglones y las columnas de cada matriz.

***U*** será una matriz de ***dxN*** (*N* = tamaño vocabulario y *d* = dimensiones del embedding). Nótese que:

$$U \cdot x^{(i)} = U^i = x^{(i)} \cdot U^T$$

# Formalización del modelo

La función de pérdida (riesgo) está dada por la entropía cruzada:

$$R(\theta) = -\sum_i \sum_o y_o \log p(w_o|w_i)$$

Donde $y_o$ indica si $w_o$ es contexto de $w_i$ ; $p(w_o|w_i)$ se define por la función Softmax:

$$p(w_o|w_i) = \frac{e^{V_o \cdot U^i}}{\sum_j e^{V_o \cdot U^j}}$$

Tal que $V_o$ es el vector que representa la palabra de salida y $U^i$ es el vector en la capa oculta (que representa a la palabra de entrada).

# Backpropagation

A partir de esta función de pérdida, se puede derivar el backpropagation:

**De la capa de salida a la capa oculta:**

$$d_{out}(o) = p(w_o|w_i) - y_o$$

De tal forma que la actualización de los pesos se hará como:

$$V_o^j \leftarrow V_o^j - \eta \cdot d_{out}(o) \cdot U_j^i$$

# Backpropagation

**De la capa oculta a la capa de entrada:**

$$d_h(j) = \sum_q V_j^q d_{out}(q)$$

Por lo que la actualización de los pesos se da como:

$$U_j^i \leftarrow U_j^i - \eta \cdot d_h(j) \cdot x^{(i)}$$

$$U_j^i \leftarrow U_j^i - \eta \cdot d_h(j)$$

**Nota:** Ya que $x^{(i)}$ es una representación one-hot, sólo se actualiza el vector columna $U^i$.

# Resumen del algoritmo

**Input:** Conjunto de pares de palabras asociadas a contextos en un vocabulario; estas se asocian a representaciones one-hot $x^{(i)}$. Rango de aprendizaje y número de iteraciones ($k$).

**Inicialización:** Se inician aleatoriamente las matrices $U$ y $V$.

**Entrenamiento**: Por $k$ iteraciones (epochs):

Para cada $x^{(i)}$ asociado a un contexto $y_o$ hacer:

1. Forward

$$U^i = U \cdot x^{(i)}$$

$$p(w_o|w_i) = \frac{e^{V_o \cdot U^i}}{\sum_j e^{V_o \cdot U^j}}$$

2. Backward

$$V_o^j \leftarrow V_o^j - \eta \cdot d_{out}(o) \cdot U_j^i$$
$$U_j^i \leftarrow U_j^i - \eta d_h(j)$$

**Finalización:** El algoritmo termina al completar las $k$ iteraciones. Se actualizan los pesos $U$ y $V$.

**Output:** Cada palabra se representa por un **vector columna** de la matriz $U$. O bien, un **vector renglón** de $U^T$

*Ximena Gutiérrez, Víctor Mijangos, 2019*

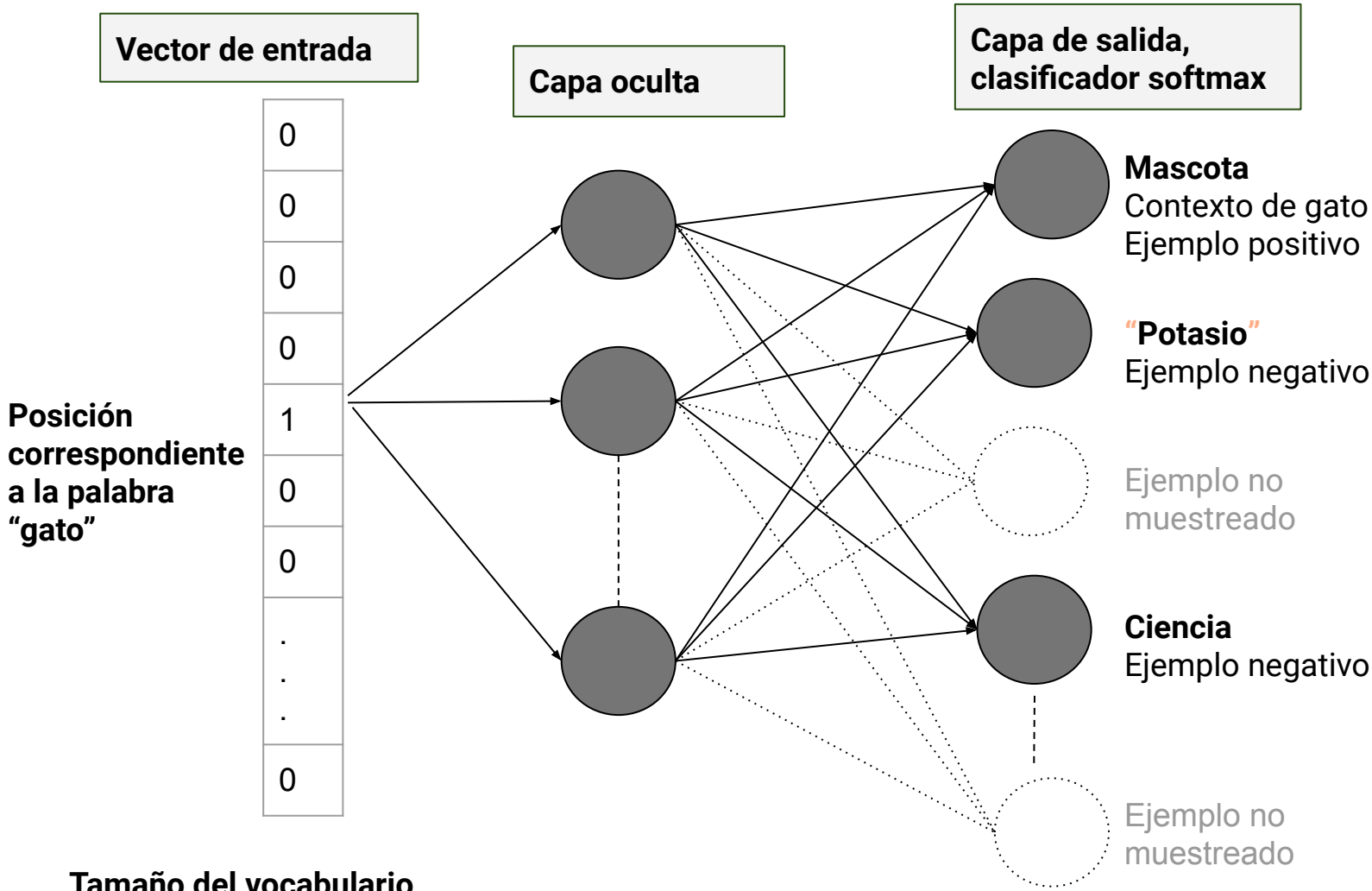
# Negative-sampling

El método así planteado tiene algunas dificultades técnicas:

- Es costoso calcular el Softmax
- Dado el tamaño requerido para entrenar la red, el tiempo de ejecución es largo

Se propone una solución a partir de negative-sampling. Este se basa en:

- La red se enfoca en las palabras que sí son contextos de un input (ejemplo positivo).
- Para contrarrestar, se muestrea un *k* número limitado de palabras no contextuales (ejemplos negativos).




*Ximena Gutiérrez, Víctor Mijangos, 2019*

# Negative sampling

Para muestrear los ejemplos negativos se utiliza una distribución definida como:

$$q(w_i) = \frac{fr(w_i)^{3/4}}{\sum_k fr(w_k)^{3/4}}$$

Cada ejemplo negativo se elige bajo esta distribución. Palabras altamente frecuentes tendrán mayor probabilidad de muestrearse.

El factor ¾ está determinado empíricamente.

# Negative sampling

El negative sampling tiene consecuencias en la red. No se puede utilizar la función Softmax (¿por qué?)

Se replantea la función de probabilidad de la siguiente forma:

$$p(D = 1|w_o, w_i) = \frac{1}{1 + e^{-V_o \cdot U^i}}$$

La variable *D* indica si las palabras pertenecen o no a un contexto. Así, para los ejemplos negativos, tenemos que:

$$p(D = 0|w_o, w_i) = 1 - p(D = 1|w_o, w_i)$$

# Entrenamiento con Negative sampling

Esto cambia la forma de la función de pérdida:

$$R(\theta) = \sum_{i} (\log \sigma(V_o \cdot U^i) + \sum_{j=1}^{k} \mathbb{E}[\log \sigma(-V_j \cdot U^i)])$$

Donde $\sigma$ es la función de probabilidad sigmoide.

Al tener una nueva función de pérdida, el proceso de backpropagation también varía.

# Backpropagation con negative sampling

Para realizar el backpropagation, basta con modificar las variables que guardan la derivación:

$$d_{out}(o) = y_o - p(D = 1|w_o, w_i)$$

$y_o$ = 1 si se trata del ejemplo positivo; $y_o$ = 0 en los ejemplos negativos.

En la capa oculta, la fórmula es similar, con la salvedad de que se toma en cuenta variable de salida que aquí se muestra.

En el algoritmo, sólo se actualizan los pesos correspondiente a los ejemplos negativos muestrados y al ejemplo positivo.

# Comentarios finales

- El método de Word2Vec es un método eficiente para obtener representaciones vectoriales de palabras
- Es un método con una arquitectura neuronal sencilla
- Su fuerza radica en la hipótesis distribucional y en el uso de modelos del lenguaje
- Actualmente, existen otros modelos de representación vectorial, pero la mayoría de estos se siguen basando en modelos del lenguaje y en la hipótesis distribucional

# Algunas herramientas para empezar

**Python,** muy popular en estas áreas. Algunos paquetes para trabajar so:

- **Gensim** (generar representaciones word2vec)
- **Scikit-learn**, **pandas** (diferentes métodos de aprendizaje de máquina)

Para construir arquitecturas neuronales:

- **Pytorch**
- **TensorFlow**

# Modelos estado del arte 2019

- **Intuiciones similares contextuales,** pero con diferentes tipos de arquitecturas neuronales, que requieren billones de parámetros, gran capacidad de cómputo y grandes corpus de entrenamiento.
- Uso de multi-task learning, transformers, transfer-learning
- Ejemplo: **BERT, ELMo, UML-FiT, GPT-2.**

| | |
|---|---|
| SYSTEM PROMPT (HUMAN-WRITTEN) | *In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.* |
| MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES) | `The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.`<br><br>`Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.`<br><br>`Dr. Jorge Pérez, an evolutionary biologist from the University of` |

* Generado con modelo GPT-2. (entrenado con 1.5 billones de parámetros con un corpus de entrenamiento de 8 millones de páginas web). Predice una palabra viendo las anteriores.

*Ximena Gutiérrez, Víctor Mijangos, 2019*