Word Embeddings

Andrés Abeliuk, Fabian Villena

Introducción

Vimos cómo representar **una palabra como un vector largo y disperso** con dimensiones correspondientes a las palabras del vocabulario o a los documentos de una colección.

A diferencia de los vectores que hemos visto hasta ahora, **los embeddings son cortos**, con un número de dimensiones que oscilan entre 50 y 1000.

Resulta que los **vectores densos funcionan mejor** en todas las tareas de PLN que los vectores dispersos.

Aprendizaje de representaciones

Word Embeddings se enmarcan dentro del **aprendizaje automático**, específicamente en el **aprendizaje de representaciones**.

- Estas técnicas permiten descubrir automáticamente las representaciones necesarias para modelar un problema.
- En el caso de los Word Embeddings, representan palabras como vectores en un espacio de características que captura relaciones semánticas y contextuales.

Word2vec

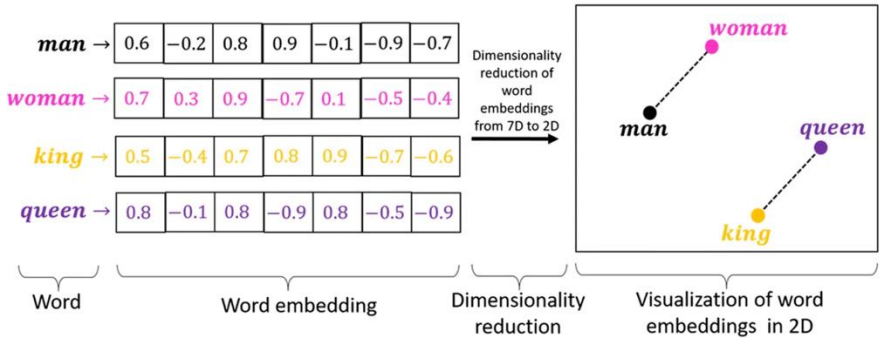
Representación Densa:

Las palabras se representan como vectores densos de baja dimensión, donde:

- Cada dimensión no tiene una interpretación específica.
- Los vectores toman valores reales.

Ventajas de Word2Vec:

- Rápidos y eficientes de entrenar.
- Disponibles en línea con implementaciones y embeddings preentrenados.



Limitaciones de Word2Vec:

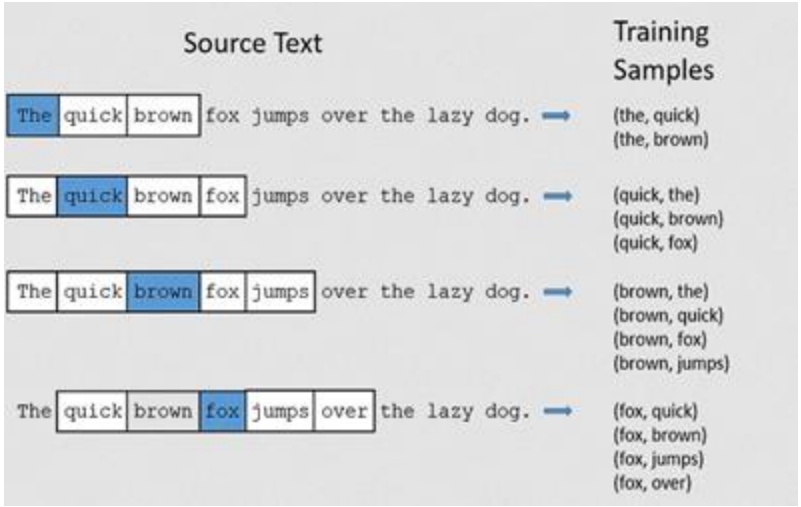
- **Embeddings Estáticos: G**enera un único embedding fijo para cada palabra, independientemente del contexto en que se use.
- Exploraremos métodos para aprender **embeddings contextuales.**

Skip-gram: Método para Calcular Embeddings

Skip-gram es un enfoque basado en redes neuronales para calcular embeddings de palabras, y a menudo se utiliza como sinónimo de Word2Vec.

Objetivo Principal:
- Predecir las palabras de contexto a partir de una palabra central en una ventana de contexto definida.

Funcionamiento:
- Cada palabra central genera predicciones sobre palabras vecinas.
- La red ajusta los parámetros para minimizar el error en estas predicciones.
- Los embeddings de las palabras son los **parámetros aprendidos.**



Skip-gram

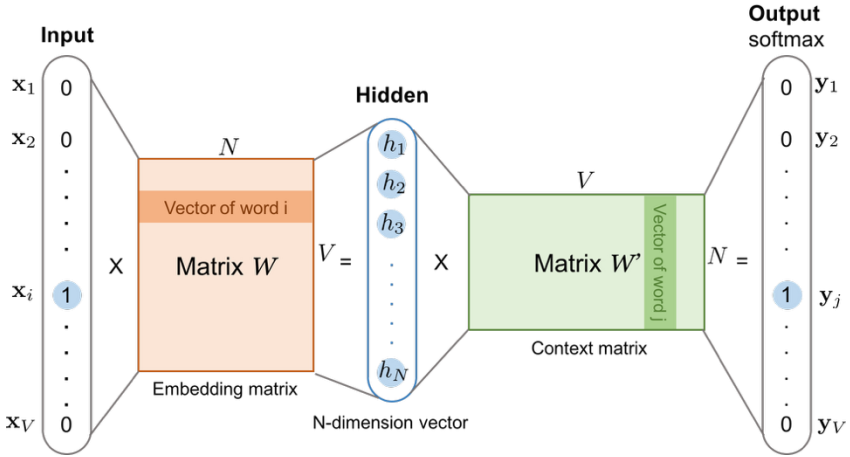
La intuición del skip-gram es:

1. Tratar la palabra objetivo y una palabra contextual vecina como ejemplos positivos.

2. Tomar muestras aleatorias de otras palabras del léxico para obtener muestras negativas.

3. Utilice la regresión logística para entrenar un clasificador que distinga esos dos casos.

4. Utilizar los pesos aprendidos como embeddings.



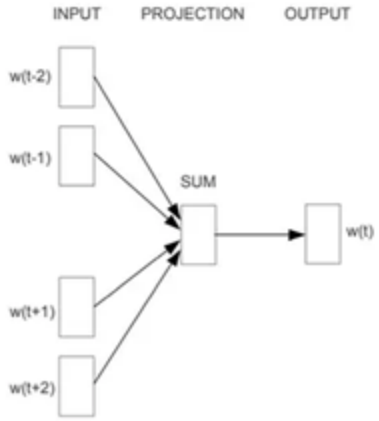
Continuous Bag of Words (CBOW)

CBOW es otro método de Word2Vec que utiliza una red neuronal para aprender embeddings.

Objetivo: Predecir la palabra central basándose en las palabras de su contexto.

Skip-Gram funciona bien con conjuntos de datos pequeños y puede representar mejor las palabras menos frecuentes.

CBOW se entrena más rápido y representa mejor las palabras más frecuentes.



Comparación: Skip-Gram vs. CBOW

Aspecto	CBOW	Skip-Gram
Funcionamiento	Predice la palabra central a partir del contexto.	Predice palabras de contexto a partir de la palabra central.
Velocidad de Entrenamiento	Más rápido de entrenar.	Más lento, pero eficiente con menos datos.
Palabras Frecuentes	Representa mejor palabras frecuentes.	Representa mejor palabras poco frecuentes.
Requisitos de Datos	Funciona mejor con grandes conjuntos de datos.	Funciona bien con conjuntos de datos pequeños.

Embeddings estáticos

Los **embeddings estáticos** son vectores fijos asignados a cada palabra del vocabulario, ajustados como parámetros en algoritmos como Word2Vec.

Ventajas:
- Capturan propiedades semánticas y sintácticas generales de las palabras.
- Son eficientes y ampliamente utilizados en tareas como clasificación de texto o análisis de sentimientos.

Limitación Principal:
- **Ambigüedad Contextual:** Una palabra puede tener múltiples significados dependiendo de su contexto, pero los embeddings estáticos siempre asignan el mismo vector.
  Ejemplo:
  "Banco" como institución financiera.
  "Banco" como asiento en un parque.

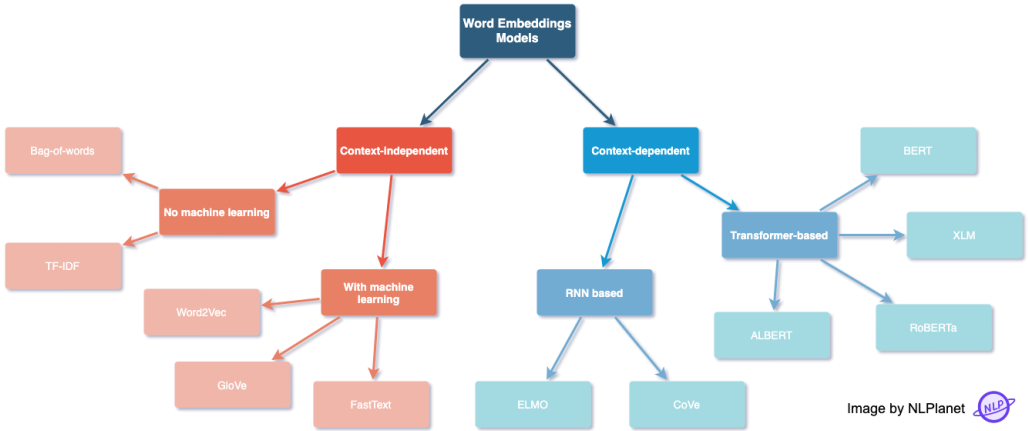
Los **embeddings contextualizados** resolverán esta limitación, ajustando la representación de cada palabra según su contexto en la oración..

Autosupervisión

La **autosupervisión** utiliza datos no etiquetados (como texto crudo) para definir tareas supervisadas de manera implícita, eliminando la necesidad de etiquetado manual.

Concepto Clave:

- El propio **texto crudo** actúa como su propia supervisión.
- Se crean tareas auxiliares o pretextuales, como:
  - Predecir palabras faltantes en una oración.
  - Identificar el orden correcto de las palabras o frases.



El Clasificador en el Modelo Word2vec

Supongamos que tenemos la frase "La manzana cayó al suelo" y utilizamos una ventana de 2 palabras contextuales.

Tarea:

- El objetivo es entrenar un clasificador que, dado un par de palabras (w, c), devuelva la probabilidad de que c sea una palabra de contexto real de w.
  - Ejemplo: (w, c) = (manzana, cayó) => probabilidad = 1 (contexto real).
  - Ejemplo: (w, c) = (manzana, perro) => probabilidad = 0 (contexto no válido).

Entrenamiento:

- El modelo ajusta los parámetros para predecir correctamente si un par de palabras (w, c) es **real** (contexto válido) o **falso** (contexto no válido).

Funcionamiento del Clasificador

¿Cómo calcula el clasificador la probabilidad?

La intuición del modelo skipgram consiste en basar esta probabilidad en la similitud de embeddings:

Dicho de otro modo: **Similitud(w,c) ≈ $\mathbf{c} \cdot \mathbf{w}$**

El producto punto $\mathbf{c} \cdot \mathbf{w}$ no es una probabilidad. Para convertir el producto punto en una probabilidad, utilizaremos la logística o sigmoide:

$$P(+|w,c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c} \cdot \mathbf{w})}$$

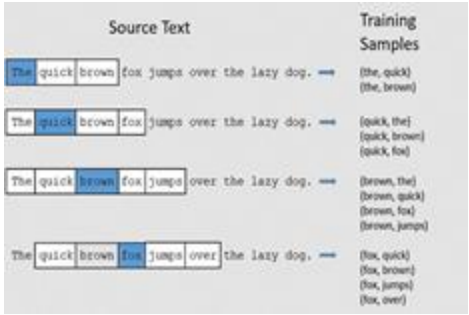
Funcionamiento del Clasificador

La ecuación anterior nos da la probabilidad de una palabra, pero hay muchas palabras de contexto en la ventana.

Skip-gram hace la suposición simplificadora de que todas las palabras de contexto son independientes, lo que nos permite simplemente multiplicar sus probabilidades:

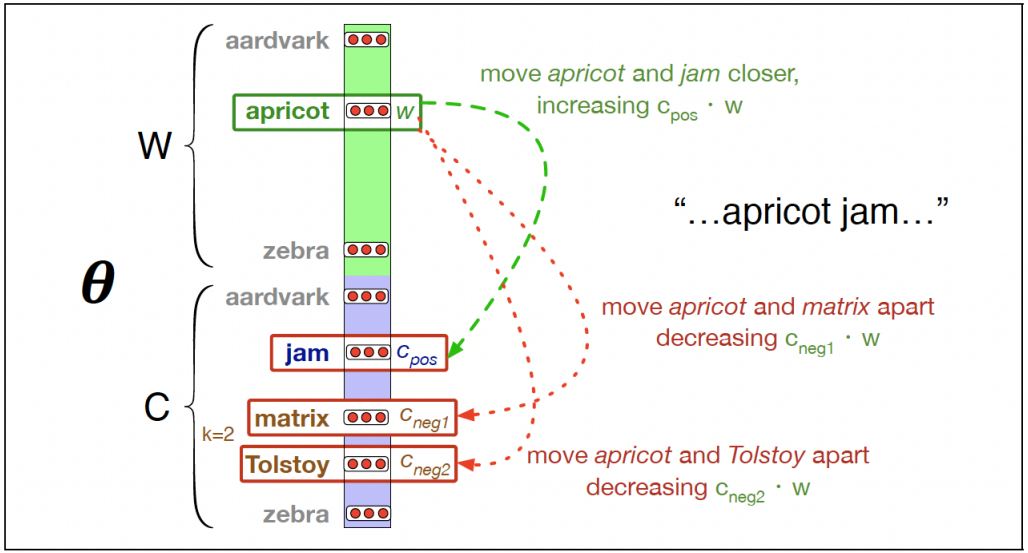
$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} \sigma(\mathbf{c_i} \cdot \mathbf{w})$$

$$\log P(+|w, c_{1:L}) = \sum_{i=1}^{L} \log \sigma(\mathbf{c_i} \cdot \mathbf{w})$$



Aprendizaje de skip-gran Embeddings

1. Se asigna un vector aleatorio a cada una de las N palabras del vocabulario.
2. Este vector es ajustado de forma iterativa (descenso gradiente).
    1. El vector de cada palabra **w** se modifica para:
    2. **Parecerse más** a las palabras que aparecen cerca en los textos (contexto positivo).
    3. **Parecerse menos** a las palabras que n aparecen cerca (contexto negativo).



Speech and Language Processing, Daniel Jurafsky, James H. Martin

Visualizando embeddings

Un vector de **100 dimensiones** es difícil de interpretar o visualizar directamente.

Solución 1: Palabras Similares:

- Una forma simple de entender el significado de una palabra en el espacio de embeddings es listar las **palabras más similares** según la cercanía en el espacio vectorial.

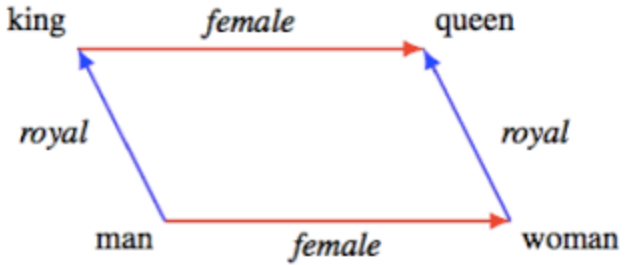
Solución 2: Proyección en 2D:

- El método más común para visualizar embeddings es proyectar las **n dimensiones** (por ejemplo, 100) a **2 dimensiones**.
- Esto se realiza mediante técnicas de reducción de dimensionalidad, como **t-SNE** (Stochastic Neighbor Embedding).

Relaciones entre significados

Los embeddings tienen la capacidad de capturar significados relacionales entre palabras.

- Se refiere al grado de similitud entre las relaciones de diferentes conceptos.
- **Ejemplo clásico:**
  - Relación entre **rey y reina** es similar a la relación entre **hombre y mujer.**
  - Puede expresarse como:
  - **rey - hombre + mujer ≈ reina**



- Los embeddings permiten **comparar y evaluar relaciones** en el espacio vectorial.
- Esto los hace útiles en tareas como analogías, similitud semántica y más.

Analogías de palabras

Estas pruebas de analogía que pueden ser resueltas con el método del paralelogramo y nos sirven para evaluar intrínsecamente el rendimiento de nuestro embedding.

Existen conjuntos de datos con estas pruebas para medir el rendimiento.

```
perro:ladrar    :: gato:maullar
otoño:hojas     :: invierno:nieve
libro:leer      :: película:ver
fútbol:balón    :: tenis:raqueta
lápiz:escribir  :: pincel:pintar
sol:día         :: luna:noche
pájaro:ala      :: pez:aleta
cuchillo:cortar :: martillo:golpear
café:cafeína    :: té:teína
nube:lluvia     :: sol:brillo
```

Sesgos en los Embeddings

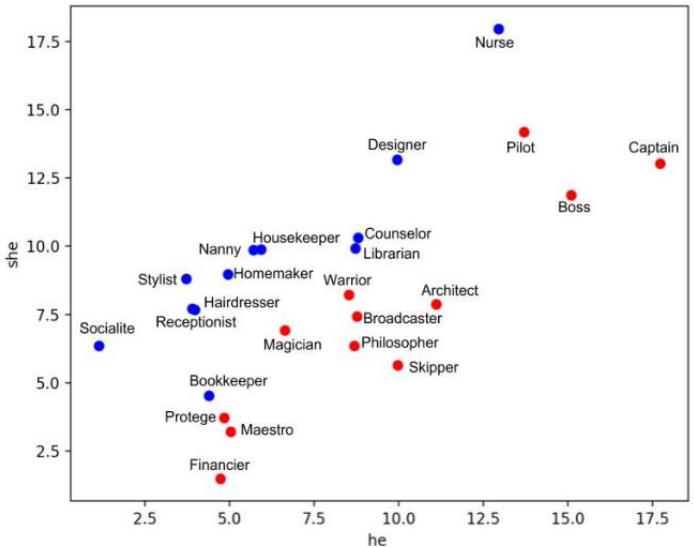
Además de su capacidad para aprender el significado de las palabras a partir del texto, **los embeddings también reproducen los prejuicios y estereotipos implícitos en el texto.**

Algunas analogías pueden exhibir estereotipos de género.

Por ejemplo, Bolukbasi et al. (2016) encuentran que en las noticias se aprehende las siguientes analogías:

programador:hombre :: ama de casa:mujer
padre:doctor :: madre:enfermera

Esto podría dar lugar a lo que se conoce como **perjuicio de asignación**, cuando un sistema asigna recursos (puestos de trabajo o créditos) de forma injusta a distintos grupos.



https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6835575.pdf

fastText

FastText es un algoritmo derivado de Word2Vec para calcular embeddings.

Diferencia Clave:

- Los embeddings no se calculan para palabras completas, sino a nivel de piezas de palabras (tokens o subpalabras).

Ventajas de FastText:

- Palabras fuera de vocabulario (OOV):
- Puede generar representaciones para palabras que no aparecen en el vocabulario entrenado.
- Lo logra componiendo las representaciones de las subpalabras que forman la palabra nueva.

Representaciones preentrenadas

Los embeddings mapean el significado de una palabra hacia un **espacio vectorial** matemático.

Esto es útil porque estos vectores pueden ser utilizados como parámetros de inicialización en arquitecturas de redes neuronales para resolver tareas de procesamiento de lenguaje natural.

Hay múltiples **representaciones preentrenadas** accesibles públicamente.
Ejemplos: **GloVe**, **Word2Vec**, **FastText**, entre otros.