Word Embeddings

Fernando Schiaffino
schiaffinofernando@gmail.com

Clase 1
Martes 15/04/2025

¿Cómo entiende el lenguaje una máquina?

Algunas consideraciones previas

Antes de meternos de lleno en los Word Embeddings revisemos dos nociones.

- Tokenización
- Vectorización

Tokenización

- Proceso que consiste en dividir una secuencia en unidades mínimas.
- Podríamos establecer una linea entre la idea de token y la idea de palabra.
- Aunque, las arquitecturas más modernas, como veremos, se alejan esta idea.
- Tokenizar:
  - Input: **'Me encantó la película.'**
  - Tokens: **['Me', 'encantó', 'la', 'pelicula', ' . ']**

Vectorización

Proceso que permite representar un texto con valores numéricos.
La idea subyacente a un modelo de embeddings es justamente representar en ese valor aspectos del significado y el uso de un token o una palabra.

Vectorización

Proceso que permite representar un texto con valores numéricos.
La idea subyacente a un modelo de embeddings es justamente representar en ese valor aspectos del significado y el uso de un token o una palabra.

Entrada:
Cadena de texto: *"Me encantó la película, la actuación fue brillante."*

Vectorización

Proceso que permite representar un texto con valores numéricos.
La idea subyacente a un modelo de embeddings es justamente representar en ese valor aspectos del significado y el uso de un token o una palabra.

Entrada:
Cadena de texto: *"Me encantó la película, la actuación fue brillante."*

Lo que la computadora entiende:

$$\text{Vector numérico: } \begin{bmatrix} 0{,}2 & 0{,}4 & 0{,}7 & 0{,}1 & \dots \end{bmatrix}$$

Vectorización

- Existen diferentes **técnicas de vectorización**, cuyos vectores resultantes variarán según el método utilizado.
- Cada técnica produce vectores con características y rangos de valores únicos.
- Algunas técnicas producen valores binarios (0 o 1), mientras que otras producen valores continuos entre 0 y 1.

One-Hot Encoding

- Representación binaria
- Se genera un vector de dimensión igual al número de palabras en el vocabulario
- Podríamos inferir que cada vector es independiente del resto, o que palabra está representada en su propia dimensión

	el	bar	esta	muy	bueno
el	1	0	0	0	0
bar	0	1	0	0	0
esta	0	0	1	0	0
muy	0	0	0	1	0
bueno	0	0	0	0	1

Bag Of Words

- Frecuencia de palabras
- Ignora la posición de las palabras
- Considera que todas las palabas son 'independientes' entre sí
- Palabras más frecuentes no necesariamente aportan significado

Documento	el	bar	no	está	muy	bueno	restaurant
El bar no está muy bueno	1	1	1	1	1	1	0
El restaurant está muy bueno	1	0	0	1	1	1	1
Vocabulario	2	1	1	2	2	1	1

TF-IDF

- Parte de la idea de BOW
- Ajusta la importancia de palabras comunes en el corpus
- Reduce el peso de las palabras muy frecuentes (como stopwords), a la vez que aumenta la importancia de las menos frecuentes (distintivas para un documento).

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \cdot \log\left(\frac{N}{\text{df}(t)}\right)$$

donde:

- $\text{tf}(t, d)$: Frecuencia de término t en el documento d
- N: Número total de documentos
- $\text{df}(t)$: Número de documentos que contienen el término t

TF-IDF

Ahora consideremos el siguiente escenario:

Documento	Texto crudo	Texto normalizado
Doc1	el bar está muy bueno	bar está bueno
Doc2	el restaurant no está muy bueno	restaurant no está bueno

Al aplicar tf-idf obtenemos:

	bar	bueno	está	no	restaurant
Doc1	0.704909	0.501549	0.501549	0.000000	0.000000
Doc2	0.000000	0.409937	0.409937	0.576152	0.576152

- No da cuenta de la sinonimia
- No entiende la negación

Word2Vec: palabras como vectores

- Familia de modelos introducida por Mikolov (2013)
- Basado en la hipótesis distribucional:
  - Palabras con significados similares tienden a aparecer en contextos similares.
  - You shall know a word by the company it keeps. Firth (1957)

Word2Vec: palabras como vectores

- Familia de modelos introducida por Mikolov (2013)
- Basado en la hipótesis distribucional:
  - Palabras con significados similares tienden a aparecer en contextos similares.
  - You shall know a word by the company it keeps. Firth (1957)
- Word2Vec transforma palabras en **vectores numéricos** (listas de números).

Word2Vec: palabras como vectores

- Familia de modelos introducida por Mikolov (2013)
- Basado en la hipótesis distribucional:
  - Palabras con significados similares tienden a aparecer en contextos similares.
  - You shall know a word by the company it keeps. Firth (1957)
- Word2Vec transforma palabras en **vectores numéricos** (listas de números).
- Estos vectores se organizan en un espacio donde la **proximidad refleja afinidad semántica**.

Word2Vec: palabras como vectores

- Familia de modelos introducida por Mikolov (2013)
- Basado en la hipótesis distribucional:
  - Palabras con significados similares tienden a aparecer en contextos similares.
  - You shall know a word by the company it keeps. Firth (1957)
- Word2Vec transforma palabras en **vectores numéricos** (listas de números).
- Estos vectores se organizan en un espacio donde la **proximidad refleja afinidad semántica**.
- Por ejemplo, las palabras *rey*, *reina*, *príncipe*, *emperador* terminan cerca entre sí en ese espacio.

¿Cómo medimos la similitud entre palabras?

- Cada palabra se representa como un **vector** en un espacio multidimensional.

¿Cómo medimos la similitud entre palabras?

- Cada palabra se representa como un **vector** en un espacio multidimensional.
- Para saber qué tan parecidas son dos palabras, medimos el **ángulo entre sus vectores**.

¿Cómo medimos la similitud entre palabras?

- Cada palabra se representa como un **vector** en un espacio multidimensional.
- Para saber qué tan parecidas son dos palabras, medimos el **ángulo entre sus vectores**.
- Cuanto más pequeño el ángulo, más parecidas las palabras.
  - Si apuntan en la misma dirección $\rightarrow$ muy similares
  - Si están en direcciones opuestas $\rightarrow$ muy diferentes

¿Cómo medimos la similitud entre palabras?

- Cada palabra se representa como un **vector** en un espacio multidimensional.
- Para saber qué tan parecidas son dos palabras, medimos el **ángulo entre sus vectores**.
- Cuanto más pequeño el ángulo, más parecidas las palabras.
  - Si apuntan en la misma dirección $\rightarrow$ muy similares
  - Si están en direcciones opuestas $\rightarrow$ muy diferentes
- Esta medida se llama **similitud del coseno**.
- Va de -1 a 1, donde:
  - 1 = vectores idénticos (máxima similitud)
  - 0 = sin relación (ángulo de 90º)
  - -1 = opuestos (inusual en lenguaje)

Semántica vectorial y relaciones léxicas

- Word2Vec no representa solo la **similitud**, también captura relaciones más sutiles:
- **Relaciones sintagmaticas**:
  - Palabras que pueden ocupar un mismo lugar en una oración (e.g., “niño”, “chico”, “perro” en “El ___ duerme”)
  - A menudo muestran significados cercanos.

Semántica vectorial y relaciones léxicas

- Word2Vec no representa solo la **similitud**, también captura relaciones más sutiles:
- **Relaciones sintagmaticas**:
  - Palabras que pueden ocupar un mismo lugar en una oración (e.g., “niño”, “chico”, “perro” en “El ___ duerme”)
  - A menudo muestran significados cercanos.
- **Relaciones asociativas**:
  - Palabras que tienden a aparecer juntas (e.g., “doctor” y “hospital”, “mate” y “bombilla”)
  - Pueden no ser sinónimos, pero están conectadas por eventos o campos semánticos.

Semántica vectorial y relaciones léxicas

- Word2Vec no representa solo la **similitud**, también captura relaciones más sutiles:
- **Relaciones sintagmaticas**:
  - Palabras que pueden ocupar un mismo lugar en una oración (e.g., “niño”, “chico”, “perro” en “El ___ duerme”)
  - A menudo muestran significados cercanos.
- **Relaciones asociativas**:
  - Palabras que tienden a aparecer juntas (e.g., “doctor” y “hospital”, “mate” y “bombilla”)
  - Pueden no ser sinónimos, pero están conectadas por eventos o campos semánticos.
- Word2Vec explota estas relaciones dadas en grandes corpus para construir un **mapa del significado**.

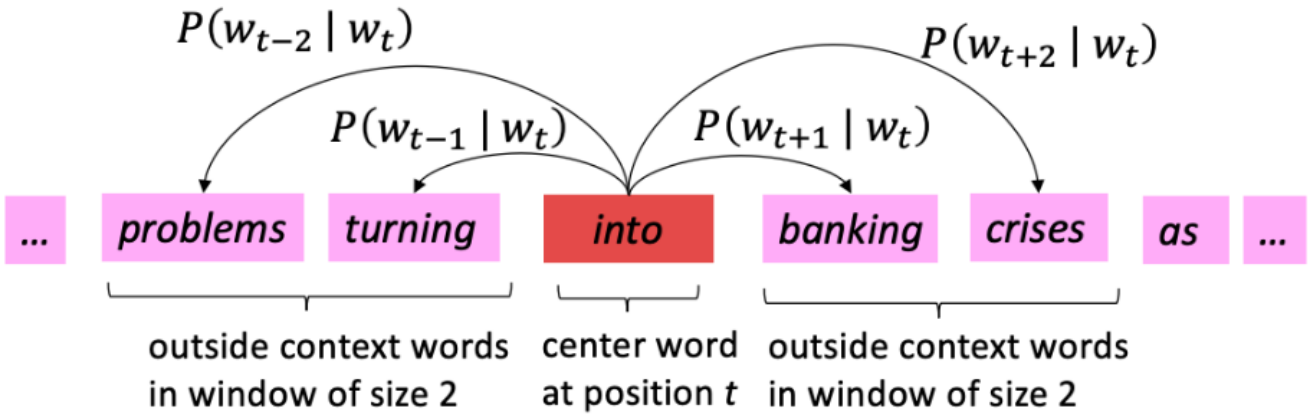
Word2Vec

- Este tipo de modelos hacen uso de redes neuronales para aprender las **probabilidades** de encontrar combinaciones de palabras para un contexto dado
- Por cada palabra, el modelo estima la probabilidad de encontrar cada una del resto de palabras del vocabulario en su contexto
- Representan las palabras en un vector denso de 50, 100 o 300 dimensiones

Word2Vec

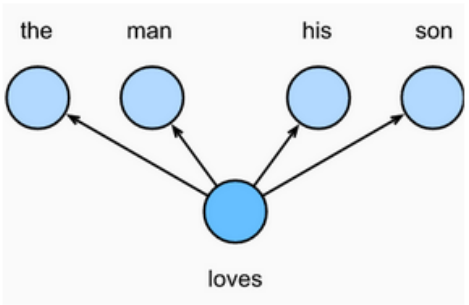
La idea fundamental de Word2Vec es representar cada palabra con dos vectores o dos matrices diferentes:

- Palabra usada como “entrada”.
- Palabra en “contexto”.



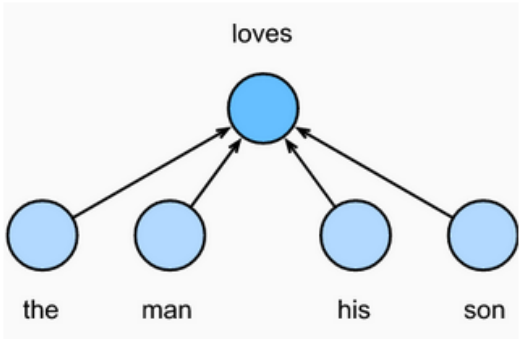
Word2Vecc

Skipgram



Skipgram necesita menor cantidad de datos y se ha encontrado que representa bien las palabras raras.

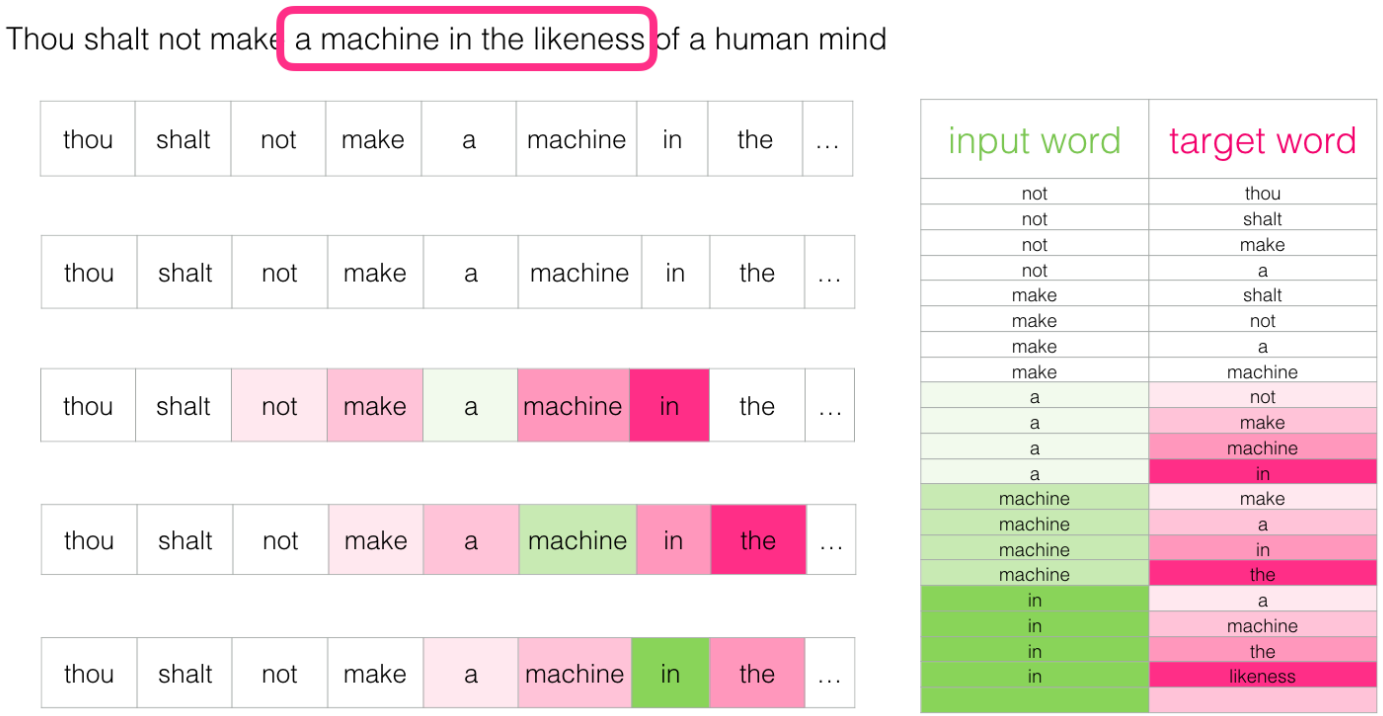
Continuous Bag of Words (CBOW)



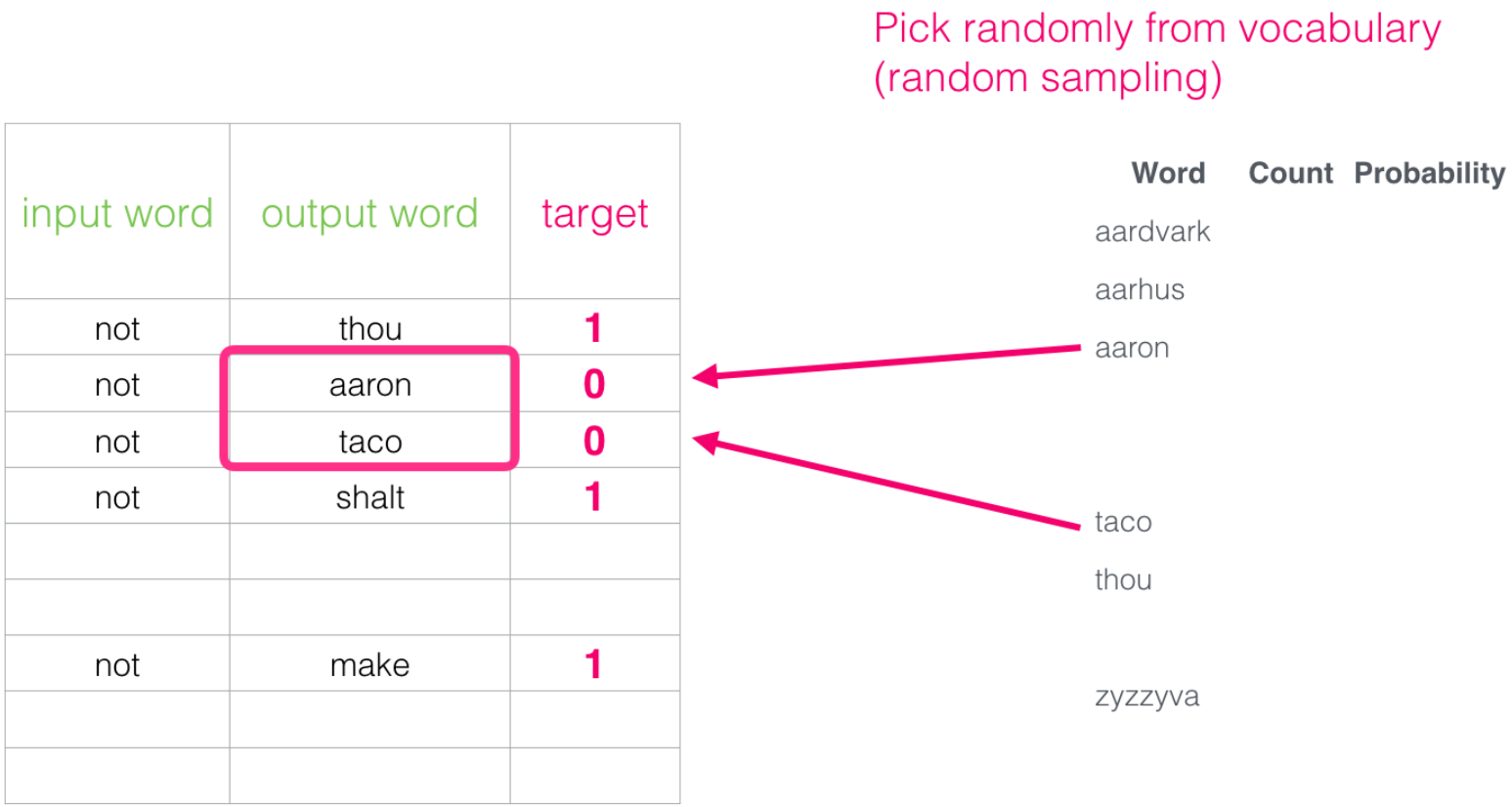
CBOW es más rápido y tiene una mejor representación para palabras más frecuentes.

Entrenamiento Auto-supervisado

- Supongamos que tomamos todo Wikipedia
- Con esos textos generamos nuestros datasets (tomando pares de palabras en una ventana de contexto deslizante)

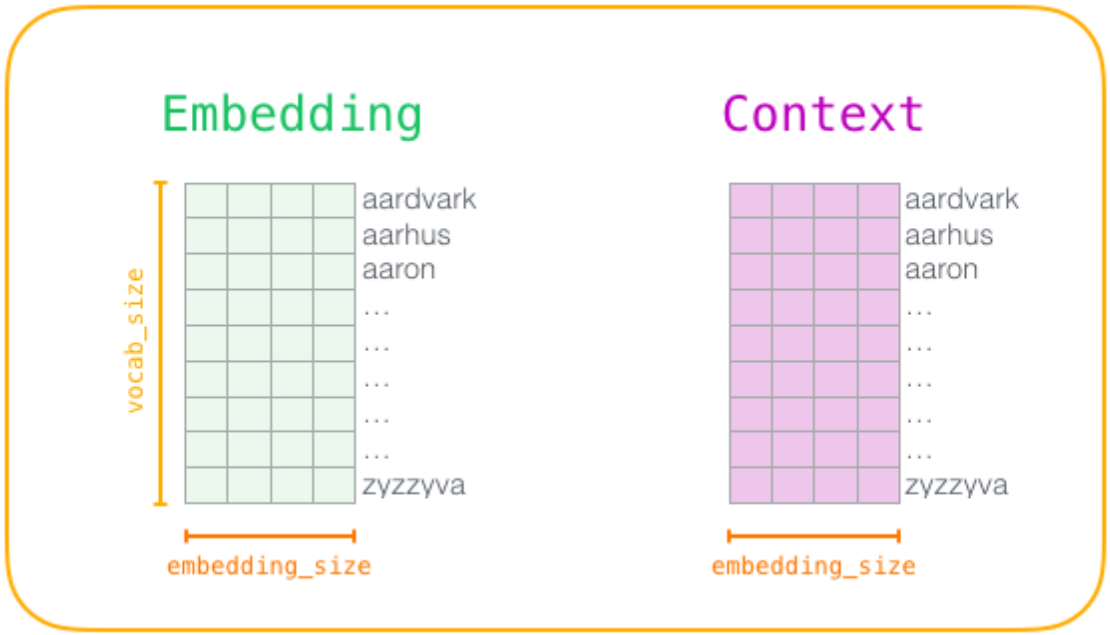


- Además, por cada palabra agregamos una serie de ejemplos negativos tomando palabras random



Entrenamiento

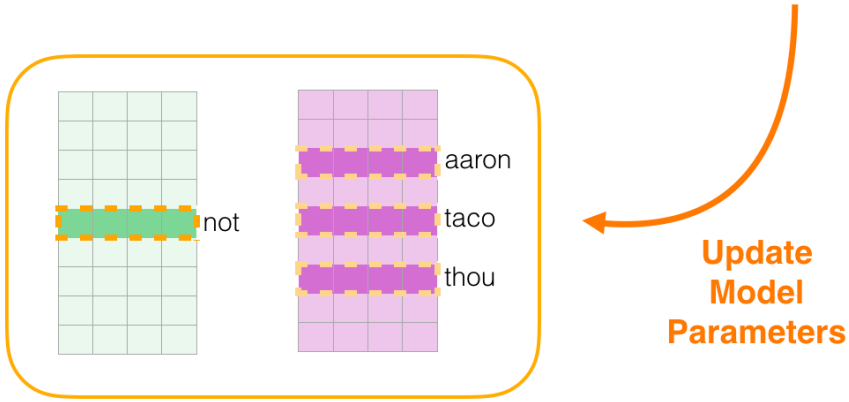
- Recordemos que este modelo representa cada palabra como dos matrices, una cuando la palabra esta siendo usada como central y otra cuando se usa como contexto.



embedding_size representa la dimensión que quiero que tenga mi embedding, vocab_size es el largo del vocabulario total

Entrenamiento

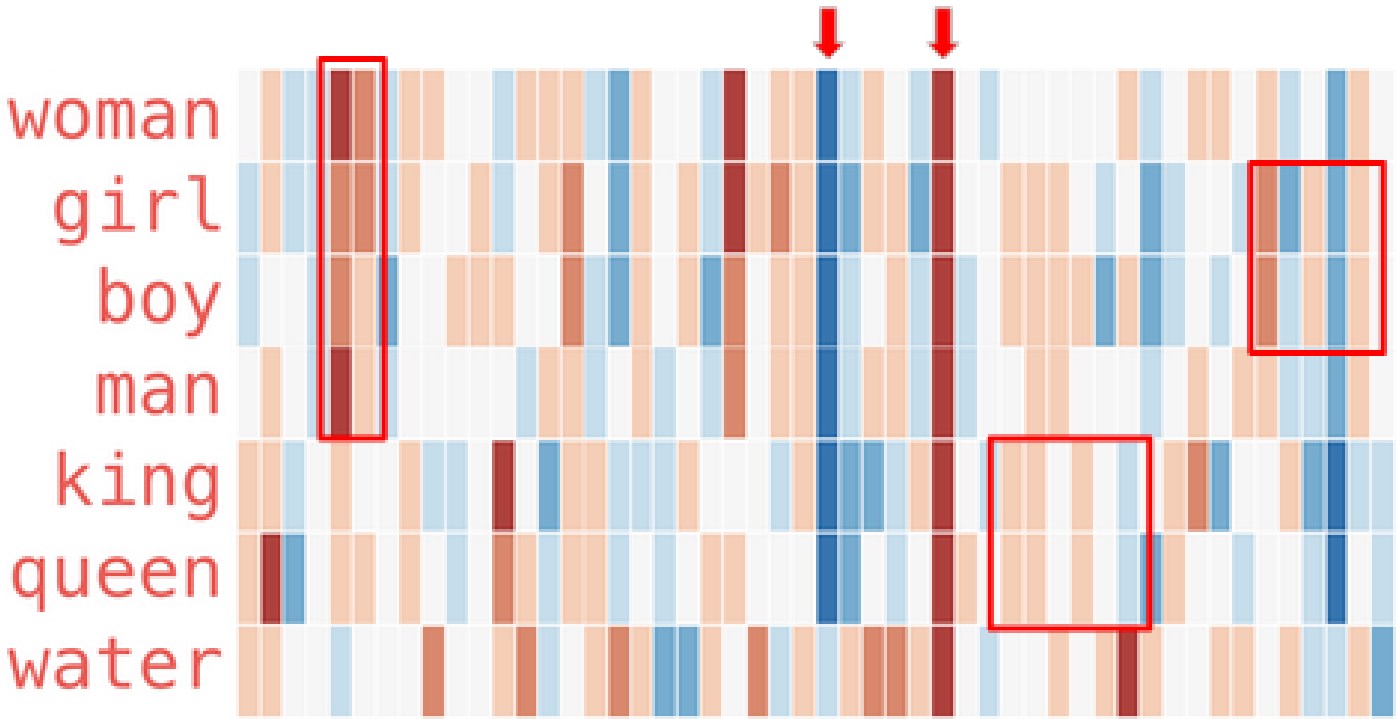
input word	output word	target	input • output	sigmoid()	Error
not	thou	1	0.2	0.55	0.45
not	aaron	0	-1.11	0.25	-0.25
not	taco	0	0.74	0.68	-0.68



La red toma los vectores de la palabra central y de las contextuales y realiza operaciones para intentar reducir el error al mínimo

Word2Vec

La idea de este tipo de entrenamientos es quedarnos con esos vectores que, luego del entrenamiento, han condesado en sus componentes algunos aspectos relevantes de cada palabra y su distribución.



Word2Vec

En este sentido los investigadores llegaron a dos resultados:

- **Esperable**: Palabras similares tienen embeddings similares
- **Inesperado**: Estos embeddings eran capaces de capturar información semántica

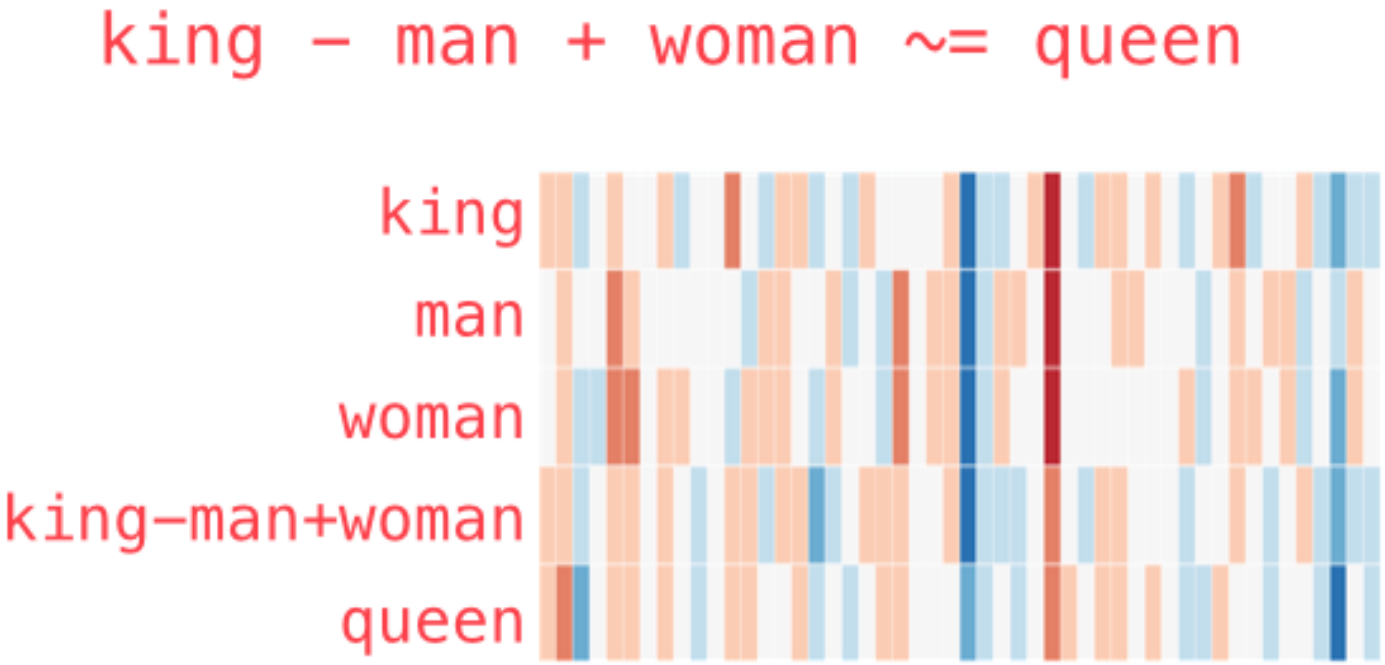
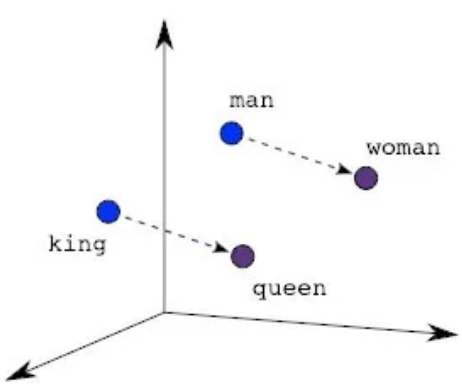
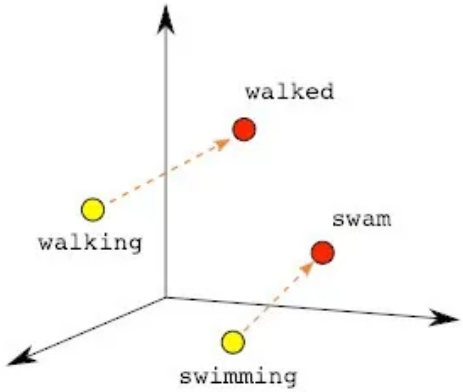


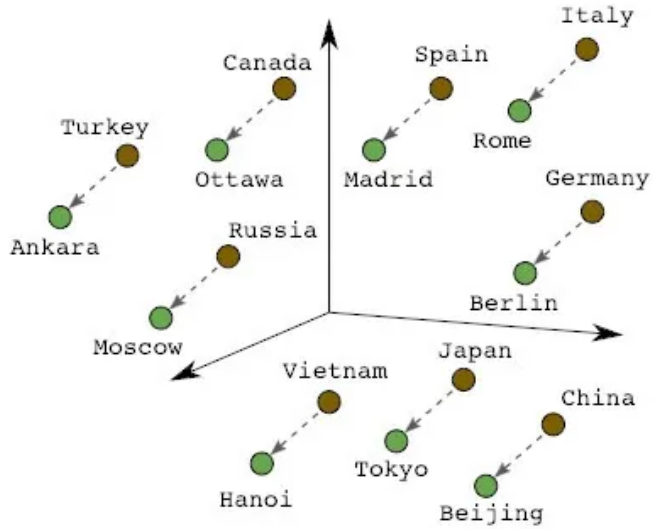
Figura: Así, se tomó el vector de la palabra 'Rey', se le restó el vector de la palabra 'Hombre', se le sumó el vector de la palabra 'Mujer' y se obtuvo un vector parecido al de la palabra 'reina'



Male-Female



Verb Tense



Country-Capital

Figura: Además, vemos como estas representaciones permiten establecer relaciones tareas de analogía

Word2Vec - Visualización

- Veamos ahora cómo se ve un embedding: TensorFlow Embedding Projector

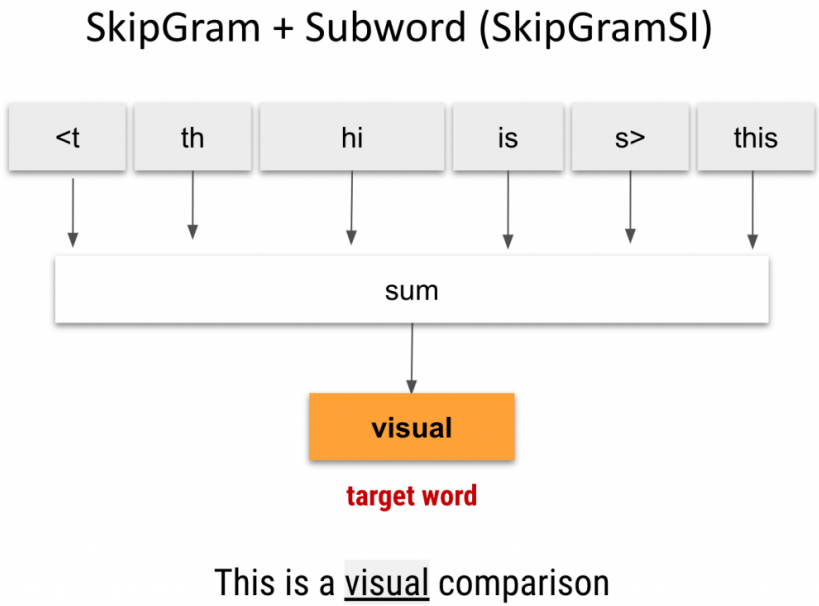
W2V - Limitaciones

- Si durante el entrenamiento no se ha encontrado un término, W2V no puede crear un vector para él y, en su lugar, asignará un vector aleatorio, lo cual no es óptimo.
- No tiene representaciones compartidas a nivel de subpalabras
- Difícil de escalar a nuevos idiomas

FastText

- Propuesto por Bojanowski *et al.* (2016)
- A diferencia de W2V, incorpora información de subpalabras
- Captura información de la estructura morfologica
- Cada palabra es representada como un conjunto de n-gramas a nivel del caracter

FastText: Subwords



- Un **n-gram de caracteres** es un conjunto de caracteres co-ocurrentes dentro de una ventana.
- Una **bolsa de n-gramas** representa una palabra como la suma de sus n-gramas.
- Asume implícitamente que cada n-gram es igualmente importante independientemente del contexto, pero, en realidad, ese no es el caso porque no todos los n-gramas son un morfema.

Recapitulemos:

- Con lo que vimos hasta acá sabemos que existen al menos dos objetivos al entrenar un modelo de embeddings
  - Predecir una palabra dado su contexto
  - Predecir un contexto dado una palabra central
- Tanto W2V como FastText generan Embeddings Estáticos, es decir, más allá de dónde aparezca la palabra, el vector será el mismo
- Ambos utilizan una arquitectura similar que contiene una capa de entrada, una capa intermedia y una capa de salida.
- Entrenan en un gran corpus, ajustando pesos para que las palabras con contextos similares tengan vectores similares.

Embeddings Contextuales

- ELMo: Embeddings contextuales usando LSTM bidireccional
- Procesan una sequencia de derecha a izquierda y de izquierda a derecha
- Capaz de extraer un embedding para cada palabra dependiendo de la posicion en la secuencia
- Transformer y Attention: Permitieron procesar secuencias en paralelo y atender a palaras alejadas sin tener que pasar secuencialmente por cada token

Modelos basados en Transformers

- Generan embeddings contextuales.
- La misma palabra puede tener diferentes representaciones según el contexto.
- Tienen una comprension mayor del concepto de ambiguedad y polisemia, ya que generan representaciones en funcion del contexto.
- Son más potentes, aunque se alejan del concepto tradicional de embeddings fijos.
- Modelos como **BERT** generan vectores a nivel del token, de la oración o del segmento y de la **posición**.

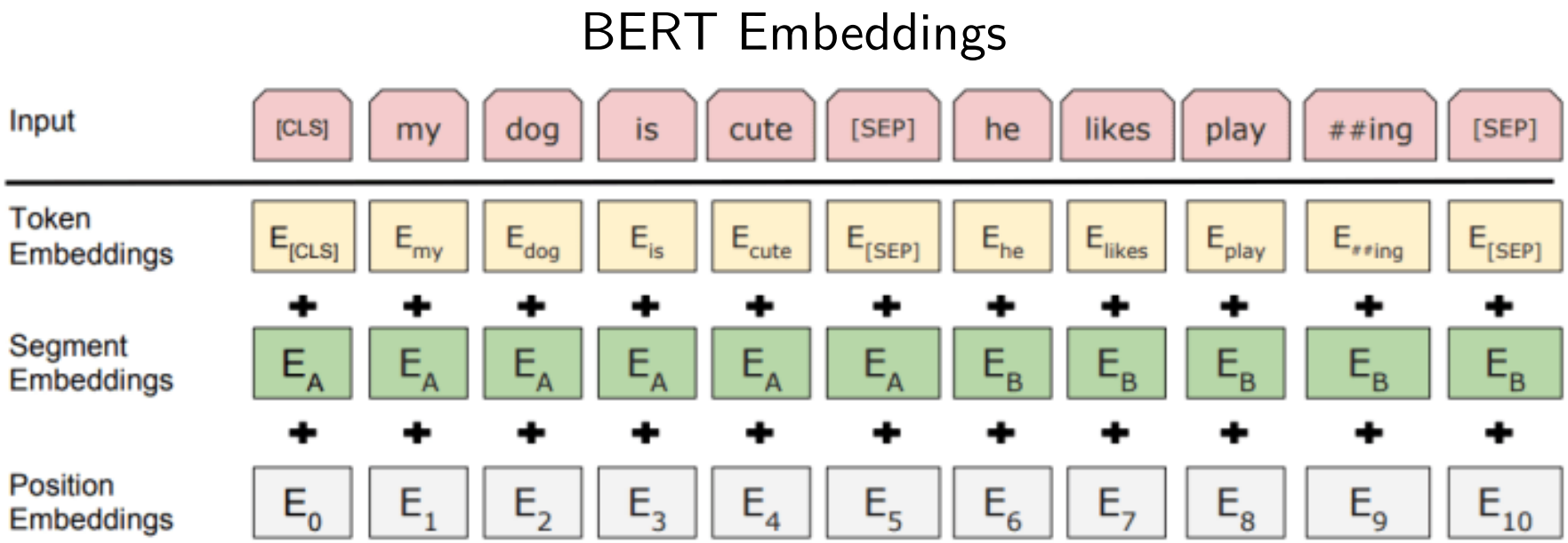


Figura: Aquí podemos ver como esta serie de arquitecturas generan embeddings para diferentes niveles de representación

Bibliografía I

Bojanowski, P., Grave, E., Joulin, A., y Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Firth, J. (1957). *A synopsis of linguistic theory, 1930-1955*, pp. 1930–1955. Philological Society.

Mikolov, T. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 3781.