Procesamiento del Lenguaje Natural (NLP). Word Embeddings y Tokenization

Inteligencia Artificial e Ingeniería del Conocimiento

Constantino Antonio García Martínez

Universidad San Pablo Ceu

Bibliografía

- Russell, Stuart J., and Peter Norvig. Artificial intelligence: a modern approach. Pearson, 2016.

¿Qué es NLP?

¿Qué es el Procesamiento del Lenguaje Natural?

Objetivo: Permitir que las máquinas comprendan, interpreten y generen lenguaje humano

Tareas típicas:

- Traducción automática
- Análisis de sentimientos
- Chatbots y asistentes virtuales
- Resumen automático
- Reconocimiento de entidades

Los Desafíos del Lenguaje Natural

1. Ambigüedad
"Vi al hombre con el telescopio"

¿Quién tiene el telescopio?

Los Desafíos del Lenguaje Natural

1. Ambigüedad
"Vi al hombre con el telescopio"

¿Quién tiene el telescopio?

2. Dependencia del Contexto
"banco"

¿Institución financiera o asiento?

Los Desafíos del Lenguaje Natural

1. Ambigüedad
"Vi al hombre con el telescopio"

¿Quién tiene el telescopio?

2. Dependencia del Contexto
"banco"

¿Institución financiera o asiento?

3. Variabilidad

- Mismo significado, diferentes formas: *"no está mal"* == *"está bien"*
- Idiomas, dialectos, registros formales/informales: "Está en el colo"
- Evolución constante: neologismos, argot: "el profe de IA da lache"

Características del Lenguaje Natural

Observación Clave

El lenguaje humano es fundamentalmente diferente de:

- Datos numéricos estructurados
- Imágenes (pixeles en cuadrícula)
- Datos tabulares

Consecuencia: Necesitamos representaciones y algoritmos especializados

Representación de Palabras

Representación Naive: One-Hot Encoding

Idea: Representar cada palabra como un vector con un 1 y el resto 0s

Example: Vocabulario pequeño

V = {gato, gata, perro, rey, reina, hombre, mujer}

$$\text{gato} = [1, 0, 0, 0, 0, 0, 0]$$

$$\text{gata} = [0, 1, 0, 0, 0, 0, 0]$$

$$\text{perro} = [0, 0, 1, 0, 0, 0, 0]$$

$$\text{rey} = [0, 0, 0, 1, 0, 0, 0]$$

$$\ldots$$

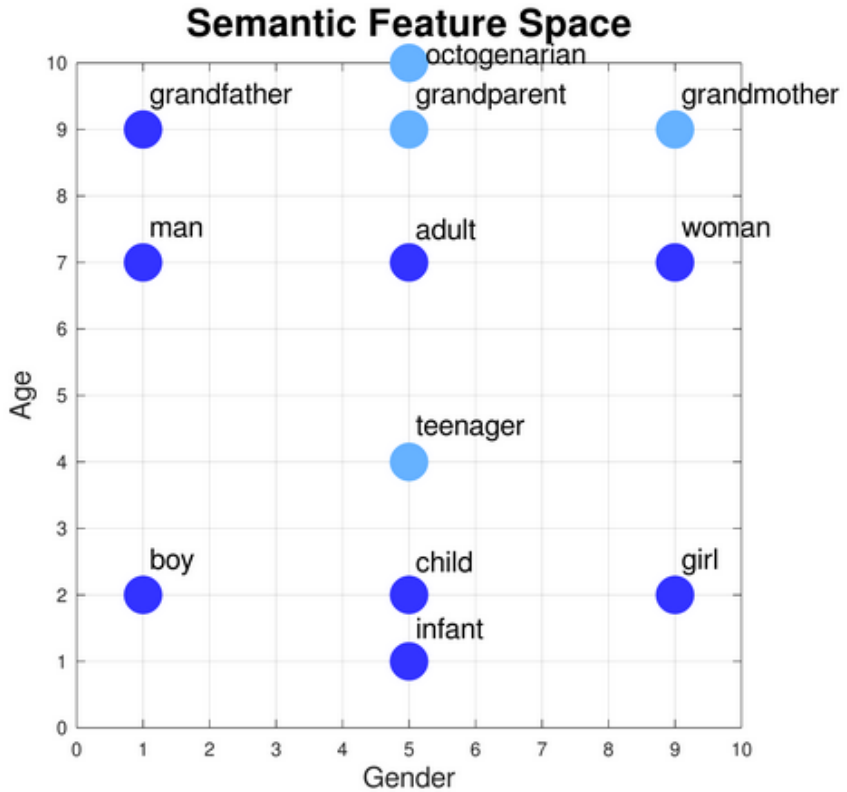
Problemas

1. **Dimensionalidad:** Tamaño del vector = $|V|$ (típicamente 50k-1M palabras)
2. **No captura similitud:** "gato" y "perro" son tan diferentes como "gato" y "democracia"
3. **No captura relaciones semánticas**

Word Embeddings: Representación Densa

Idea: Representar palabras en espacio continuo de dimensión baja (típicamente 50-300 (GloVe) hasta 1024-4096+ en arquitecturas modernas)

Example: Embeddings en dimensión 2



Word Coordinates

	Gender	Age
grandmother	9	9
grandparent	5	9
octogenarian	5	10
teenager	5	4

[1]

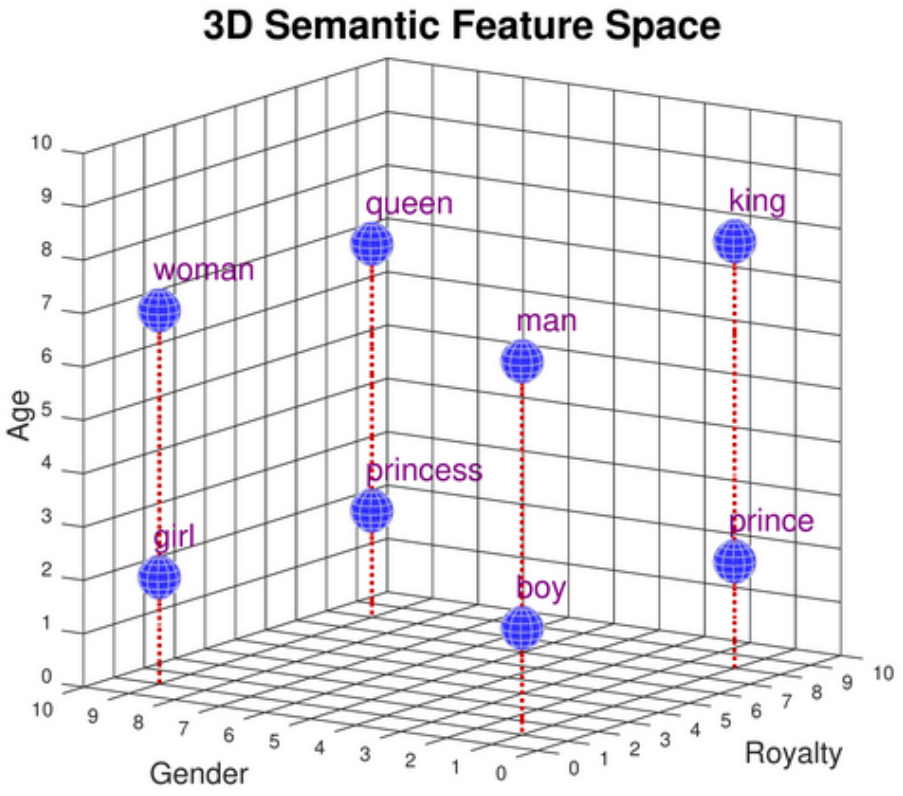
[1] Fuente: https://www.cs.cmu.edu/ dst/WordEmbeddingDemo/tutorial.html

Word Embeddings: Representación Densa

Ventajas

1. **Dimensión fija y pequeña** (no crece con vocabulario)
2. **Captura similitud:** palabras similares $\rightarrow$ vectores cercanos
3. **Captura relaciones:** rey - hombre + mujer $\approx$ reina

Example: Embeddings en dimensión 3



Word Coordinates

	Gender	Age	Royalty
man	[1,	7,	1]
woman	[9,	7,	1]
boy	[1,	2,	1]
girl	[9,	2,	1]
king	[1,	8,	8]
queen	[9,	7,	8]
prince	[1,	2,	8]
princess	[9,	2,	8]

[2]

[2] Fuente: https://www.cs.cmu.edu/ dst/WordEmbeddingDemo/tutorial.html

La Hipótesis Distribucional

Firth (1957)
"Conocerás una palabra por la compañía que tiene"

Idea: Palabras que aparecen en contextos similares tienen significados similares

La Hipótesis Distribucional

Firth (1957)
"Conocerás una palabra por la compañía que tiene"

Idea: Palabras que aparecen en contextos similares tienen significados similares

Example: Mismo contexto, palabras similares

- "El _____ doméstico es un animal muy independiente"
- Puede ser: "gato", "perro", "conejo"
- NO puede ser: "teorema", "democracia", "azul"

Si "gato" y "perro" pueden aparecer en los mismos contextos
$\rightarrow$ Sus embeddings deberían ser similares

La Hipótesis Distribucional

Firth (1957)
"Conocerás una palabra por la compañía que tiene"

Idea: Palabras que aparecen en contextos similares tienen significados similares

Example: Mismo contexto, palabras similares

- "El _____ doméstico es un animal muy independiente"
- Puede ser: "gato", "perro", "conejo"
- NO puede ser: "teorema", "democracia", "azul"

Si "gato" y "perro" pueden aparecer en los mismos contextos
$\rightarrow$ Sus embeddings deberían ser similares

Fundamento de: Word2Vec, GloVe, FastText, etc.

Aprendiendo Embeddings: GloVe

Global Vectors (GloVe): Aprende embeddings basándose en co-ocurrencias

Loss Function de GloVe (simplificada)

$$L = \sum_{i,j} f(X_{ij}) \left(w_i^T w_j + b_i + b_j - \log X_{ij} \right)^2$$

- X_{ij}: número de veces que palabra i y j co-ocurren
- w_i, w_j: embeddings de palabras
- $f(X_{ij})$: función de peso

Evolución de las Representaciones de Palabras

1. **Antes (2013-2018): Word2Vec, GloVe, FastText**
   - Entrenados por separado del modelo principal
   - El modelo principal se entrenaba sobre estos embeddings
   - Usualmente, los embeddings se mantenían congelados o se aplicaba fine-tuning

Evolución de las Representaciones de Palabras

1. **Antes (2013-2018): Word2Vec, GloVe, FastText**
   - Entrenados por separado del modelo principal
   - El modelo principal se entrenaba sobre estos embeddings
   - Usualmente, los embeddings se mantenían congelados o se aplicaba fine-tuning

2. **Ahora (2018+): BERT, GPT, LLaMA, etc.**
   - Los embeddings se aprenden de forma conjunta con el resto de la red neuronal

Evolución de las Representaciones de Palabras

1. **Antes (2013-2018): Word2Vec, GloVe, FastText**
   - Entrenados por separado del modelo principal
   - El modelo principal se entrenaba sobre estos embeddings
   - Usualmente, los embeddings se mantenían congelados o se aplicaba fine-tuning

2. **Ahora (2018+): BERT, GPT, LLaMA, etc.**
   - Los embeddings se aprenden de forma conjunta con el resto de la red neuronal

Los embeddings iniciales en ambos casos son estáticos (1 palabra = 1 vector). Sin embargo, estudiaremos que tras pasar por varias capas Transformer, **los embeddings se vuelven contextuales**

Example: Embedding contextual
"Me senté en el **banco**" $\neq$ "Fui al **banco**" (diferentes vectores)

Tokenización: De Palabras a Subpalabras

Problema: Vocabulario de Palabras

Con word embeddings, cada palabra única necesita su propio vector

Problemas

1. **Palabras fuera de vocabulario (OOV):**
   - Vocabulario de entrenamiento: {gato, gata, perro, ...}
   - Palabra nueva: "gatito" $\rightarrow$ ¿cómo representarla?
   - Solución común: token especial <UNK> (pierde información)

Problema: Vocabulario de Palabras

Con word embeddings, cada palabra única necesita su propio vector

Problemas

1. **Palabras fuera de vocabulario (OOV):**
   - Vocabulario de entrenamiento: {gato, gata, perro, ...}
   - Palabra nueva: "gatito" $\rightarrow$ ¿cómo representarla?
   - Solución común: token especial <UNK> (pierde información)
2. **Palabras relacionadas tratadas como independientes:**
   - "jugar", "jugando", "jugó", "jugador" $\rightarrow$ 4 vectores diferentes
   - No capturan que comparten raíz "jug-"

Problema: Vocabulario de Palabras

Con word embeddings, cada palabra única necesita su propio vector

Problemas

1. **Palabras fuera de vocabulario (OOV):**
   - Vocabulario de entrenamiento: {gato, gata, perro, ...}
   - Palabra nueva: "gatito" $\rightarrow$ ¿cómo representarla?
   - Solución común: token especial <UNK> (pierde información)
2. **Palabras relacionadas tratadas como independientes:**
   - "jugar", "jugando", "jugó", "jugador" $\rightarrow$ 4 vectores diferentes
   - No capturan que comparten raíz "jug-"
3. **Vocabulario enorme:**
   - Español: 100k palabras comunes
   - Cada palabra flexionada = entrada diferente
   - "hablar": hablo, hablas, habla, hablamos, habláis, hablan, hablaba, ...

Solución: Tokenización por Subpalabras (BPE)

Byte Pair Encoding (BPE): Dividir palabras en unidades más pequeñas (subpalabras)

Example: Tokenización BPE
Vocabulario de tokens: {gat, o, a, ito, perr, jugar, jug, ando, ador}

- "gato" → ["gat", "o"]
- "gata" → ["gat", "a"]
- "gatito" → ["gat", "ito"] (¡funciona aunque no esté en vocabulario!; ¡muy útil para erratas!)
- "jugando" → ["jug", "ando"]
- "jugador" → ["jug", "ador"]

Solución: Tokenización por Subpalabras (BPE)

Byte Pair Encoding (BPE): Dividir palabras en unidades más pequeñas (subpalabras)

Example: Tokenización BPE
Vocabulario de tokens: {gat, o, a, ito, perr, jugar, jug, ando, ador}

- "gato" → ["gat", "o"]
- "gata" → ["gat", "a"]
- "gatito" → ["gat", "ito"] (¡funciona aunque no esté en vocabulario!; ¡muy útil para erratas!)
- "jugando" → ["jug", "ando"]
- "jugador" → ["jug", "ador"]

Ventajas de BPE

1. **Sin palabras OOV:** Cualquier palabra se puede representar
2. **Comparte información:** "jug-" tiene mismo embedding en las variantes
3. **Vocabulario más pequeño:** 50k tokens vs 100k+ palabras
4. **Eficiente para múltiples idiomas**

BPE con Transformers

Example: tokenization.py