

# Inteligencia Artificial Aplicada para la Economía

 Profesores

**Profesor Magistral**

Camilo Vega Barbosa

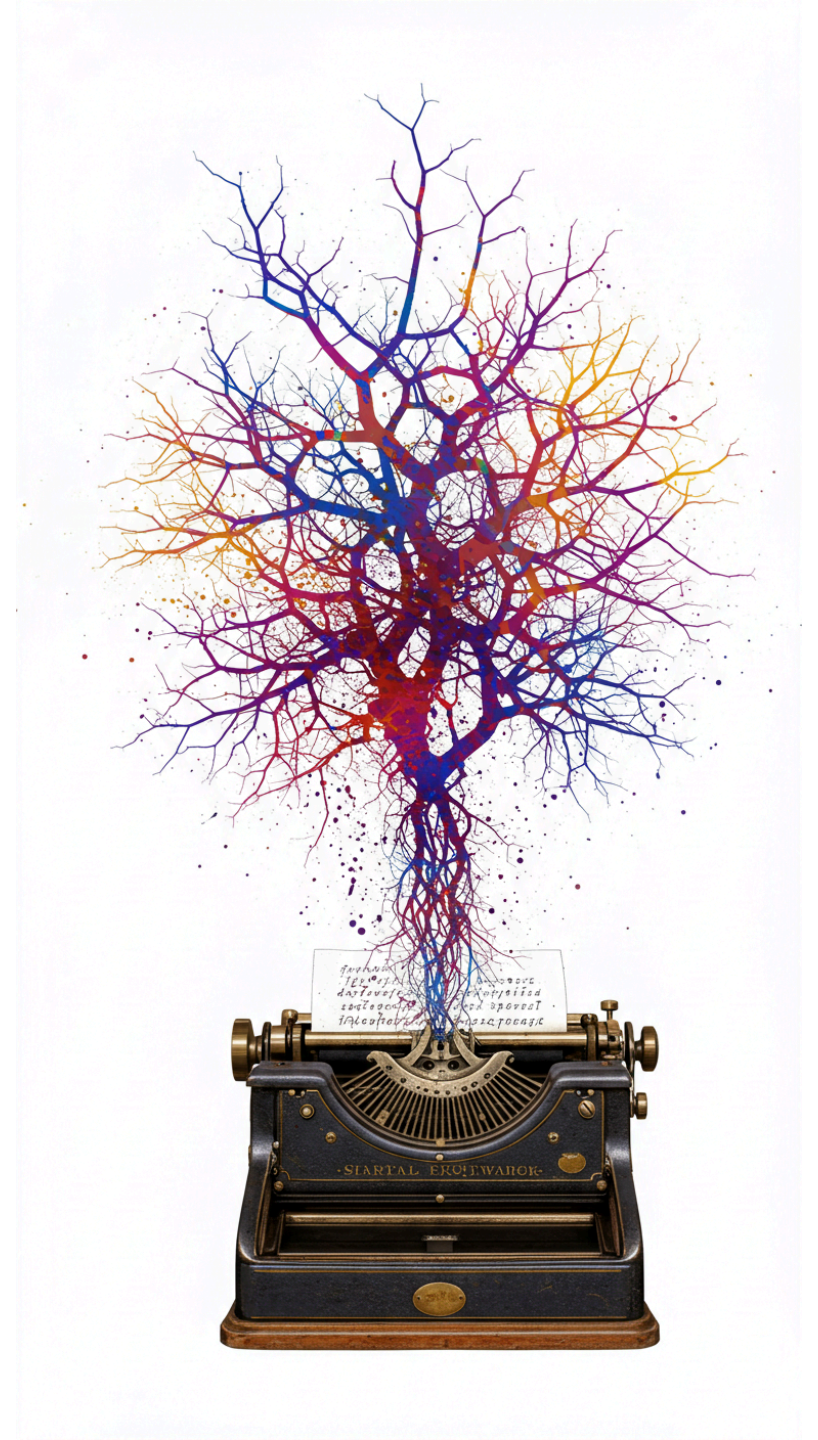
**Asistente de Docencia**

Sergio Julian Zona Moreno



# Fundamentos de Procesamiento de Lenguaje Natural (NLP)



Hablando con la máquina  













# Introducción al Procesamiento de Lenguaje Natural

El Procesamiento de Lenguaje Natural (NLP) es una rama de la inteligencia artificial que permite a las máquinas entender 📖, interpretar 🔍 y generar ✍️ lenguaje humano. Su objetivo principal es cerrar la brecha comunicativa entre humanos y computadoras.

El NLP combina lingüística computacional , aprendizaje automático 🤖 y análisis estadístico  para procesar texto y voz de forma significativa. Esto permite aplicaciones como asistentes virtuales, traducción automática, análisis de sentimiento y extracción de información.




A diferencia de los lenguajes de programación tradicionales, donde los humanos debían adaptarse a la máquina, el NLP permite que las computadoras comprendan nuestro lenguaje natural con sus ambigüedades, contexto y matices semánticos. 🌐

## ¿Para qué sirve el NLP?








-  Asistentes virtuales y chatbots inteligentes
-  Análisis de sentimiento en texto
-  Resumen automático de documentos
-  Traducción automática entre idiomas
-  Sistemas de búsqueda semántica
-  Extracción de información estructurada
-  Publicidad y marketing personalizado
-  Interfaces conversacionales

# Relevancia del NLP en la Era Digital

## Explosión de datos textuales no estructurados

- 90% de datos empresariales son texto 
- Crecimiento exponencial de contenidos digitales 
- Imposibilidad de procesamiento manual 

## Transformación industrial





- Automatización documental 
- Asistentes inteligentes 
- Análisis de sentimiento a gran escala   
- Búsqueda semántica avanzada 
- Personalización de la experiencia del usuario 

## El Reto del Preprocesamiento de Textos

Para que las máquinas entiendan texto, primero debemos transformarlo en algo procesable .




*El preprocesamiento determina cómo el modelo "ve" e interpreta el lenguaje, condicionando todo el funcionamiento posterior.*

Es como enseñar a leer a un niño .

1. Primero aprende letras 
2. Luego forma palabras 
3. Después comprende oraciones 
4. Finalmente interpreta significados 

# El Reto del Preprocesamiento de Textos


## Procesos fundamentales


- Tokenización : Segmentar el texto en unidades básicas
- Normalización : Estandarizar texto para análisis
- Vectorización : Convertir texto en representaciones numéricas


## Impacto en el rendimiento

- Mejora la precisión del modelo
- Reduce ruido y ambigüedades
- Optimiza recursos computacionales
- Facilita generalización a nuevos casos

## Tokenización: Fundamentos

La tokenización es el proceso de dividir texto en unidades básicas llamadas tokens  que pueden ser procesadas por modelos computacionales. Este paso fundamental determina cómo los sistemas de NLP interpretarán el lenguaje.

El proceso varía según idioma y aplicación . En lenguas con separación clara como el español o inglés, los espacios sirven como delimitadores naturales. Sin embargo, en idiomas como el chino o japonés sin espacios, o con morfología compleja como el finlandés, la tokenización requiere enfoques especializados.

Los métodos modernos de tokenización por subpalabras (como BPE, WordPiece y SentencePiece) equilibran la eficiencia de vocabularios compactos con la capacidad de capturar componentes morfológicos significativos, mejorando el rendimiento en palabras desconocidas y raras. 



# Tokenización: Del Texto a Unidades Procesables

El proceso completo :

1. Texto bruto :

"El Banco Central reduce tasas"

2. Tokenización :

["El", "Banco", "Central", "reduce", "tasas"]

3. Asignación de IDs :




[143, 2580, 1863, 507, 3298]


4. Embedding :

Cada ID se convierte en un vector multidimensional

# Tokenización: Del Texto a Unidades Procesables

## Técnicas principales

- **Por palabras** : División por espacios y signos de puntuación
  - *"El gato come"* → [*"El"*, *"gato"*, *"come"*]
- **Por caracteres** : Cada carácter como unidad básica
  - *"El gato"* → [*"E"*, *"l"*, *" "*, *"g"*, *"a"*, *"t"*, *"o"*]
- **Subword** : Unidades entre caracteres y palabras
  - *"inversiones"* → [*"inver"*, *"###siones"*]

La tokenización subword permite balance entre eficiencia (vocabulario compacto) y efectividad (captura morfología). Los modelos modernos usan vocabularios de 30,000-50,000 tokens. 

# Algoritmos de Tokenización Avanzada

## SentencePiece

- Tokenización sin separación previa por espacios
- Adecuado para idiomas sin espacios (japonés, chino)
- Tratamiento unificado multi-idioma
- Aumenta portabilidad entre idiomas

## Byte-Pair Encoding (BPE)

- Comienza con caracteres individuales
- Fusiona iterativamente los pares más frecuentes
- Construye vocabulario de forma incremental
- Usado en GPT-2, GPT-3, RoBERTa

# Algoritmos de Tokenización Avanzada

## WordPiece



- Similar a BPE con criterio diferente
- Prioriza pares que maximizan probabilidad
- Usado en BERT y sus variantes
- Marca subpalabras con ##



## Normalización de Texto

*La normalización es el proceso de estandarizar el texto para reducir variabilidad no significativa, permitiendo que el modelo se enfoque en patrones importantes.*

### Técnicas comunes

- Case folding : Convertir a minúsculas
  - "Economía" → "economía"
- Eliminación de stopwords :
  - Remover palabras muy frecuentes con poco valor semántico
  - "el", "la", "y", "que", "en", "a"



















## Normalización de Texto

- Eliminación de puntuación y caracteres especiales ✂
  - *"economía!"* → *"economía"*
- Corrección ortográfica ✓
  - *"ecnomia"* → *"economía"*
- Expansión de contracciones ↺
  - *"del"* → *"de el"*
- Estandarización de formato 📅
  - *"23/05/2023"* → *"2023-05-23"*

## Stemming: Reducción algorítmica

- **Definición** : Reducción algorítmica a la raíz
- **Algoritmo Porter** : Elimina sufijos mediante reglas
  - "*inversiones*" → "*invers*"
  - "*inversionista*" → "*invers*"
- **Características** :
  - Rápido pero impreciso 
  - No siempre genera palabras reales 
  - Puede sobrerreducir o subreducir 
  - Muy útil para recuperación de información 

## Lemmatization: Reducción lingüística

- **Definición** : Reducción a forma canónica (lema)
- **WordNet Lemmatizer** : Usa diccionario y análisis morfológico
  - "*inversiones*" → "*inversión*"
  - "*mejores*" → "*bueno*"
- **Características** :
  - Más preciso pero más costoso computacionalmente 
  - Requiere conocimiento del idioma 
  - Genera palabras válidas del diccionario 
  - Preserva el significado semántico 

# Comparación: Stemming vs Lemmatization

## Ejemplo práctico:

- Stemming: "corrieron", "correrá", "corredores" → "corr" ✂️
- Lemmatization: "corrieron", "correrá", "corredores" → "correr" 📖


## Stemming 🗑️


- Proceso más rápido ⚡
- Basado en reglas heurísticas 📏
- Mayor tasa de error ⚠️
- Ideal para grandes volúmenes 📊



## Lemmatization 🔍

- Proceso más lento 🐢
- Basado en diccionarios 📚
- Mayor precisión ✅
- Ideal para análisis lingüístico 🧠

# Word Embeddings: Semántica en el Espacio Vectorial

Los word embeddings son representaciones vectoriales de palabras en un espacio multidimensional  donde la posición y distancia entre vectores captura relaciones semánticas. Esta técnica revolucionó el NLP al permitir que los modelos comprendan significados y similitudes entre palabras.

A diferencia de las representaciones tradicionales (como one-hot encoding), los embeddings capturan relaciones semánticas: palabras con significados similares tienen vectores cercanos en el espacio . Esto emerge naturalmente del análisis estadístico de cómo las palabras aparecen en contextos similares.






La propiedad más notable es la posibilidad de realizar operaciones algebraicas con significados : "Rey - Hombre + Mujer  $\approx$  Reina". Estas representaciones, típicamente de 100-300 dimensiones, forman la base de muchos modelos avanzados de NLP. 



## Word Embeddings: Semántica en el Espacio Vectorial

Los *word embeddings* convierten palabras en vectores que preservan relaciones semánticas y permiten razonamiento analógico.  $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow$  

**Ventajas clave:**

- Capturan similitud semántica y contextual  - Permiten operaciones algebraicas con significados 
- Reducen dimensionalidad del problema 
- Mejoran generalización del modelo 
- Facilitan transferencia de aprendizaje 

# Word Embeddings: Propiedades Fundamentales

Los word embeddings transforman palabras en vectores multidimensionales con propiedades semánticas útiles:

- La **posición** en el espacio refleja el significado 📍
- La **distancia** entre vectores cuantifica similitud semántica 📏
- Las **direcciones** capturan relaciones conceptuales 🧭

Esto permite operaciones algebraicas con significados:

Rey - Hombre + Mujer  $\approx$  Reina 👑

## Operaciones semánticas 📊

- Similitud:  $\cos(\text{economía}, \text{finanzas}) \approx 0.8$
- Analogías: París:Francia :: Madrid:España
- Agrupamiento: formación de clusters

## Aplicaciones principales 🚀




- Búsqueda semántica 🔍
- Sistemas de recomendación 👍
- Análisis de sentimiento 😊


# Similitud de Coseno: Matemática de la Semántica

La similitud de coseno es la métrica fundamental para medir qué tan "cercanas" semánticamente están dos palabras en el espacio de embeddings.

$$\text{similitud\_coseno}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}}$$

Donde:

- $\vec{a}$  y  $\vec{b}$  son vectores de embedding 
- $\vec{a} \cdot \vec{b}$  es el producto escalar ✖
- $|\vec{a}|$  y  $|\vec{b}|$  son las magnitudes (normas) 
- El resultado varía entre -1 (direcciones opuestas) y 1 (misma dirección) 

La similitud de coseno mide el **ángulo** entre vectores, ignorando sus magnitudes. Valores cercanos a 1 indican gran similitud semántica. 

## Ejemplo Práctico: Similitud de Coseno (1/2)

Vectores de embedding simplificados (dimensión reducida):

- $\vec{economía} = [0.2, 0.5, -0.3, 0.8]$
- $\vec{finanzas} = [0.3, 0.6, -0.2, 0.7]$
- $\vec{literatura} = [-0.4, 0.1, 0.7, 0.2]$

Cálculo paso a paso para  $\text{sim}(\text{economía}, \text{finanzas})$ :

1. Producto punto:

$$\begin{aligned}\vec{economía} \cdot \vec{finanzas} &= (0.2 \times 0.3) + (0.5 \times 0.6) + (-0.3 \times -0.2) + (0.8 \times 0.7) \\ &= 0.06 + 0.30 + 0.06 + 0.56 = 0.98\end{aligned}$$

2. Magnitud de economía:

$$\begin{aligned}|\vec{economía}| &= \sqrt{0.2^2 + 0.5^2 + (-0.3)^2 + 0.8^2} \\ &= \sqrt{0.04 + 0.25 + 0.09 + 0.64} = \sqrt{1.02} = 1.01\end{aligned}$$

## Ejemplo Práctico: Similitud de Coseno (2/2)



3. Magnitud de finanzas:

$$\begin{aligned} |\vec{finanzas}| &= \sqrt{0.3^2 + 0.6^2 + (-0.2)^2 + 0.7^2} \\ &= \sqrt{0.09 + 0.36 + 0.04 + 0.49} = \sqrt{0.98} = 0.99 \end{aligned}$$

4. Similitud de coseno:

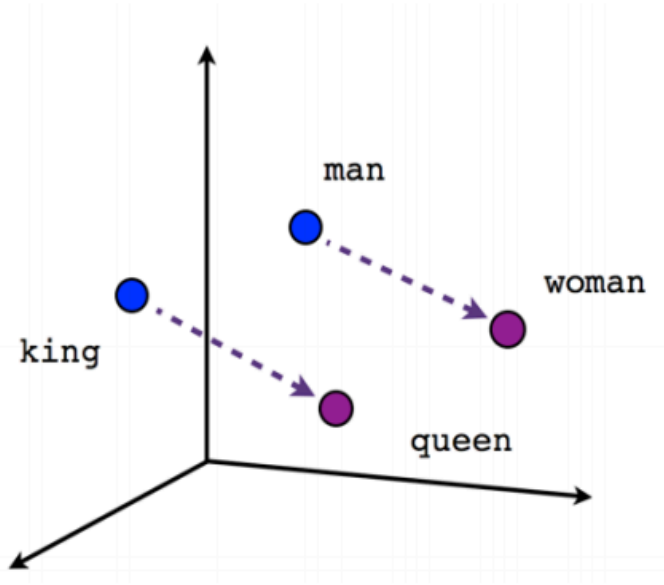
$$\frac{\vec{economía} \cdot \vec{finanzas}}{|\vec{economía}| |\vec{finanzas}|} = \frac{0.98}{1.01 \times 0.99} = \frac{0.98}{1.00} = 0.98$$

## Resultados comparativos

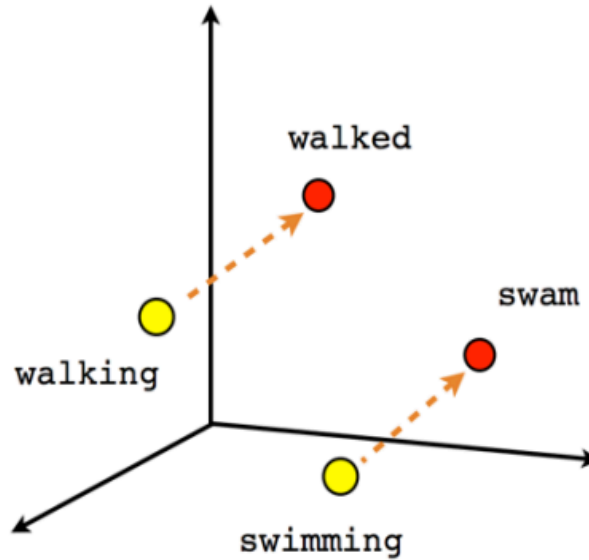
- $\text{sim}(\text{economía}, \text{finanzas}) = 0.98 \rightarrow$  ¡Muy similares! 
- $\text{sim}(\text{economía}, \text{literatura}) = -0.22 \rightarrow$  Divergentes 
- $\text{sim}(\text{finanzas}, \text{literatura}) = -0.05 \rightarrow$  Casi ortogonales  $\perp$



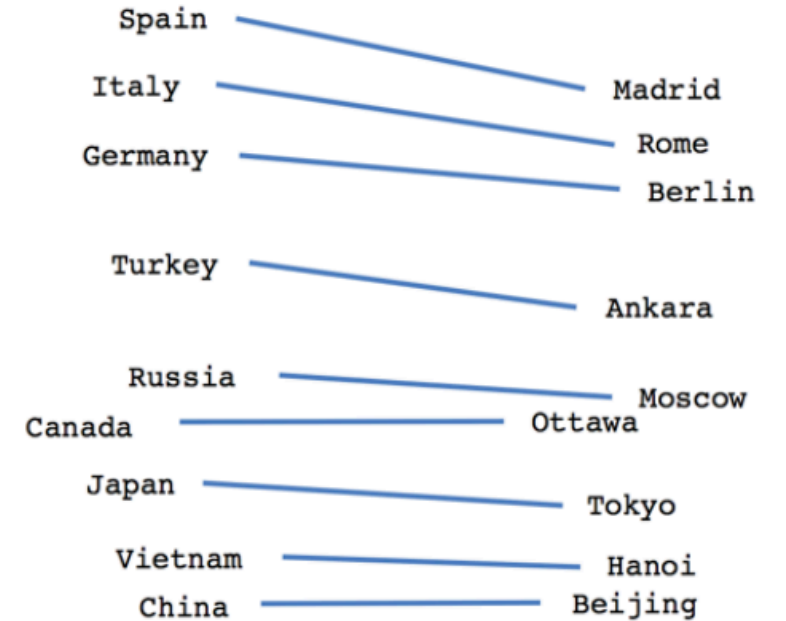
# 🧠 Visualización de Word Embeddings



Male-Female



Verb tense



Country-Capital


Tomado de: <https://cdn.analyticsvidhya.com/wp-content/uploads/2017/06/06062705/Word-Vectors.png> ✖

# Dimensionalidad en Word Embeddings

Los embeddings de alta dimensionalidad permiten capturar relaciones semánticas más complejas entre palabras 🧠. Como un "vocabulario" enriquecido para el modelo, estos espacios vectoriales representan mejor los matices contextuales, reducen colisiones conceptuales y modelan jerarquías más profundas entre palabras relacionadas. ✨

## Servicios de Embeddings 🚀

- **Word2Vec/GloVe**: 50-300 dimensiones (estándar)
- **FastText**: 300 dimensiones (común)
- **BERT**: 768-1024 dimensiones (capas profundas)
- **Ada (OpenAI)**: 1536 dimensiones (estado del arte)

El aumento de dimensiones tiene un límite práctico donde el beneficio marginal se reduce frente al costo computacional y riesgo de sobreajuste. 

# 🧩 El Pipeline Completo de NLP: Un Ejemplo Práctico

## Texto original

"La Economía Española creció un 2.5% en el último trimestre, superando las expectativas de los analistas."

## 1 Tokenización

["La", "Economía", "Española", "creció", "un", "2.5", "%", "en", "el", "último", "trimestre", ",", "superando", "las", "expectativas", "de", "los", "analistas", "."]

## 2 Normalización

- Minúsculas: ["la", "economía", ...]
- Sin stopwords: ["economía", "española", "creció", "2.5", "%", "último", "trimestre", "superando", "expectativas", "analistas"]
- Lemmatización: ["economía", "español", "crecer", "2.5", "%", "último", "trimestre", "superar", "expectativa", "analista"]

## 3 Vectorización

Cada token se convierte en un vector de embedding

## Consideraciones en el Orden del Procesamiento

**Regla general:** La secuencia tokenización → normalización → vectorización es la más común, pero algunos ajustes pueden ser necesarios según el caso de uso.

### Variaciones importantes ⚠

- Algunas normalizaciones (como eliminar URLs o corregir ortografía) pueden aplicarse **antes de tokenizar**
- Ciertos algoritmos de tokenización incluyen normalizaciones implícitas
- Los modelos modernos (transformers) realizan tokenización y embedding simultáneamente
- La vectorización puede ocurrir a nivel de palabra, documento o frase según la aplicación

# Frameworks y Modelos Populares para NLP

## spaCy

- Biblioteca de NLP industrial y de código abierto
- Optimizado para producción (rápido y eficiente)
- Pipeline completo: tokenización, POS, NER, parsing
- Fácil integración con deep learning
- Soporte multilingüe
- Ideal para: extracción de información, procesamiento a gran escala

## NLTK

- Plataforma clásica para NLP en Python
- Amplia colección de recursos lingüísticos
- Gran cantidad de algoritmos implementados
- Enfoque educativo y de investigación
- Ideal para: prototipado, análisis lingüístico, enseñanza



# Modelos Transformers para NLP

## BERT (Bidirectional Encoder Representations from Transformers)

- Desarrollado por Google (2018)
- Comprensión bidireccional del contexto
- Pre-entrenado en grandes corpus de texto
- Revolucionó el estado del arte en múltiples tareas
- Variantes: RoBERTa, DistilBERT, ALBERT

## GPT (Generative Pre-trained Transformer)

- Desarrollado por OpenAI, pero ya hay una gran oferta de modelos equivalentes.
- Enfoque autoregresivo (predice el siguiente token de palabras con base a los anteriores)
- Excelente en generación de texto

## Ventajas de los Modelos Transformer

**Características clave:** - Atención simultánea a todas las palabras del contexto 👁️ - Capturan dependencias a larga distancia 🔗 - Pre-entrenamiento + fine-tuning (transfer learning) 🔄 - Representaciones contextuales dinámicas ✨ - Estado del arte en prácticamente todas las tareas de NLP 🏆

## Aplicaciones prácticas 🚀

- Asistentes virtuales avanzados (ChatGPT, Claude)
- Búsqueda semántica empresarial
- Análisis de documentos a gran escala
- Traducción de alta calidad
- Generación de contenido personalizado

## Recursos del Curso

## Plataformas y Enlaces Principales

### GitHub del curso

 [github.com/CamiloVga/IA\\_Aplicada](https://github.com/CamiloVga/IA_Aplicada)

### Asistente IA para el curso

 [Google Notebook LLM](#)