Natural Language Processing (NLP)

El problema de computar o codificar el lenguaje natural

1. Motivación inicial: ¿Cómo harías que una máquina entendiera lenguaje natural?

- **Planteamiento del problema**:
  - Dada una frase, ¿cómo puede una máquina entender su contenido, por ejemplo, el sentimiento?
- **Enfoque intuitivo histórico**:
  - Primeras aproximaciones: codificar reglas gramaticales manualmente.
  - Dificultades: enorme complejidad, ambigüedades, y necesidad constante de actualización.

---

2. Dos grandes paradigmas en NLP

- **a. Enfoque basado en reglas (Rule-based NLP):**
  - Codificación manual de reglas gramaticales y sintácticas.
  - Ejemplo: Sistemas expertos, parsers clásicos.
  - Ventajas y desventajas:
    - Ventaja: Control total, interpretabilidad.
    - Desventaja: No escalable, costoso de mantener, poco flexible a cambios en el lenguaje.
- **b. Enfoque basado en aprendizaje automático (Machine Learning):**
  - Utilización de grandes corpus de texto para aprender reglas automáticamente.
  - Algoritmos que aprenden patrones lingüísticos a partir de datos.

- Avance reciente: predominancia del aprendizaje automático y Deep Learning.

---

3. El reto de procesar texto: necesidad de representación numérica

- **Las máquinas entienden números, no texto**
  - Las redes neuronales y la mayoría de algoritmos ML trabajan con números.
  - Ejemplo: Imagen como matriz de píxeles, audio como señales digitalizadas, tablas con variables numéricas.
  - El texto debe convertirse en una representación numérica antes de ser procesado.

---

4. Estrategias de representación de texto

- **a. Codificación por caracteres:**
  - Cada carácter representado por un código numérico (por ejemplo, ASCII).
  - Ejemplo: “A” = 65, “B” = 66, etc.
  - Ventajas y desventajas:
    - Ventaja: Simple, universal, permite generar palabras nuevas.
    - Desventaja: Difícil generar textos coherentes y realistas.
- **b. Codificación por palabras (tokenización):**
  - El texto se divide en palabras ("tokens").
  - A cada palabra se le asigna una etiqueta numérica.
  - Necesidad de que la asignación sea siempre la misma.
  - Problema: Asignar números secuenciales puede inducir relaciones numéricas erróneas entre palabras.
- **c. Codificación por subpalabras:**
  - División del texto en fragmentos más pequeños que la palabra, pero más grandes que el carácter.

- Ejemplo: Byte-Pair Encoding (BPE), usado en modelos modernos como GPT-2 y GPT-3.
- Ventaja: Equilibrio entre flexibilidad y coherencia léxica.

---

5. Tokenización

- **Definición de token**
  - Unidades mínimas del texto sobre las que operamos (palabras, caracteres, subpalabras).
- **Proceso de tokenización**
  - Ejemplo de tokenización por palabras: “La tostadora es nueva” → [“La”, “tostadora”, “es”, “nueva”].
- **Importancia de la tokenización para NLP y Deep Learning**

---

6. Codificación numérica de tokens: Limitaciones

- **Asignar una sola etiqueta numérica es insuficiente**
  - Problema: El modelo puede interpretar relaciones inexistentes entre palabras (por ejemplo, “tostadora” = 15, “teléfono” = 20, “pangolín” = 30; la distancia numérica carece de significado lingüístico).
  - Analogía con variables categóricas: el valor numérico no debe tener interpretación ordinal o métrica.

---

7. Representaciones One-Hot

- **Solución: Representar cada palabra como un vector de ceros y un uno**
  - Vocabulario de tamaño N → cada palabra se representa como un vector de N posiciones, todas a 0 excepto la correspondiente a la palabra, que es 1.
- **Ejemplo geométrico:**
  - Tres palabras (“teléfono”, “tostadora”, “pangolín”):
    - “teléfono” = [1, 0, 0]
    - “tostadora” = [0, 1, 0]
    - “pangolín” = [0, 0, 1]

- **Ventajas:**
  - No hay relación numérica artificial entre palabras.
  - Todas las distancias entre palabras son iguales (ortogonalidad).
- **Desventajas:**
  - Gran consumo de memoria para vocabularios grandes (matrices muy dispersas).
  - No refleja similitud semántica (la distancia entre “tostadora” y “teléfono” es igual a la de “pangolín” y “teléfono”).

---

8. Limitaciones de la representación one-hot

- **Escalabilidad**
  - Problemas de espacio y eficiencia con vocabularios grandes.
- **Ausencia de relaciones semánticas**
  - La representación no capta que ciertas palabras son conceptualmente próximas.
  - Ejemplo: “tostadora” y “teléfono” pueden estar más relacionadas que “pangolín”, pero la representación one-hot no refleja esto.

---

9. Motivación para usar Embeddings

- **Introducción al concepto de embeddings**
  - Necesidad de una representación que capture relaciones semánticas y sintácticas entre palabras.
  - Los embeddings permiten que palabras similares tengan representaciones numéricas próximas.
  - Conexión directa entre NLP y Deep Learning.
- **Anticipo**
  - El tema de embeddings y sus técnicas se desarrollará en el próximo vídeo.

Resumen y Cierre

- El campo del NLP ha evolucionado desde sistemas basados en reglas hacia modelos que aprenden representaciones y relaciones a partir de grandes cantidades de datos.
- Representar el texto de forma adecuada es un paso fundamental para que los algoritmos puedan aprender y procesar lenguaje natural.
- El próximo paso lógico es el uso de embeddings, que será explorado en la siguiente unidad/vídeo.

Embeddings y reducción de dimensionalidad en NLP

1. Introducción

- **Contexto:**

  El punto de partida en procesamiento de lenguaje natural (NLP) es que los ordenadores solo procesan números, pero el lenguaje humano es texto. Para que un modelo basado en redes neuronales pueda trabajar con texto, debemos transformar las palabras en números de una forma adecuada.

2. Representación vectorial básica: One-hot encoding

- **¿Qué es?**

  Es una codificación donde cada palabra se representa como un vector de la misma longitud que el vocabulario; todos sus elementos son 0, excepto uno en la posición que corresponde a la palabra.

- **Ventajas:**
  - Sencillez.
  - Representación geométrica clara: cada palabra ocupa su propia dimensión, siendo todas independientes.
- **Desventajas:**
  - **Equidistancia:** Todas las palabras están a la misma distancia unas de otras; no hay noción de similitud semántica.

- **Alta dimensionalidad:** Si el vocabulario es de 10,000 palabras, cada vector tiene 10,000 dimensiones, lo que es poco eficiente y genera problemas de escalabilidad.

3. Limitaciones cognitivas de one-hot encoding

- **Carencia de relaciones semánticas:**

  Como humanos, entendemos que palabras como “planeta”, “galaxia” y “universo” están conceptualmente más cerca entre sí que de “pangolín”, pero one-hot encoding no captura esa información.

- **Necesidad de incorporar similitud:**

  El objetivo es conseguir representaciones vectoriales donde la distancia y la proximidad entre palabras refleje similitud conceptual.

4. Reducción de dimensionalidad: motivación y ejemplo visual

- **Analogía visual:**

  Se plantea una tarea donde se ordenan imágenes (caras) en función de dos atributos (ejemplo: color de piel, edad). De miles de píxeles (dimensiones) se pasa a solo dos dimensiones, comprimiendo la información y facilitando la interpretación.

- **Paralelismo en redes neuronales:**

  En redes convolucionales, las imágenes pasan de alta a baja dimensionalidad conforme avanzan por las capas, hasta quedar representadas por un vector pequeño (por ejemplo, de tamaño 10 o 20).

5. Reducción de dimensionalidad en texto: embeddings

- **Transición de one-hot a embeddings:**

  En vez de trabajar con vectores de 10,000 dimensiones (vocabulario), la red neuronal aprende a comprimir cada palabra a un vector de, por ejemplo, 300 dimensiones.

- **¿Cómo ocurre esto?**

La primera capa de la red neuronal actúa como una “capa de embeddings”, encargada de transformar el vector one-hot en un vector denso más pequeño (embedding), que será optimizado según la tarea (por ejemplo, análisis de sentimiento o reviews de electrónica).

6. Características de los embeddings

- **Adaptabilidad:**

  La representación que aprende la red depende del objetivo: palabras con carga emocional parecida estarán cerca si la tarea es análisis de sentimientos, o palabras técnicas de un dominio concreto si es clasificación de reviews técnicas.

- **Mejora de la representación:**

  Los embeddings permiten a la red capturar relaciones semánticas y estructurales entre palabras, algo imposible para los one-hot.

7. Embeddings pre-entrenados y transferencia de aprendizaje

- **Analogía con visión por computadora:**

  Se compara con usar redes pre-entrenadas en imágenes (transfer learning). Es mejor partir de una red que ya ha aprendido representaciones generales.

- **Embeddings pre-entrenados:**

  Existen modelos de embeddings (como Word2Vec) ya entrenados sobre grandes corpus, que pueden ser usados en tareas nuevas, mejorando el rendimiento y acelerando el aprendizaje.

- **Ventaja:**

  Palabras similares quedan próximas en el espacio vectorial, y esta representación se puede reutilizar en muchos contextos.

8. Word2Vec y la universalidad de los embeddings

- **¿Cómo se entrena un embedding “universal”?**

  1. Muchísimo texto.

2. Una tarea genérica (como predecir la palabra siguiente en una frase) que no condicione excesivamente la estructura del embedding.

- **Resultado:**

  Las palabras conceptualmente parecidas quedan cerca; podemos visualizar estos agrupamientos usando técnicas de reducción de dimensionalidad (ejemplo: clusters de números, meses, palabras de la misma raíz, palabras del mismo campo semántico...).

- **Ejemplo de agrupamientos:**
  - Verbos de movimiento (“go”, “run”, “wake”).
  - Números (“zero”, “five”, “seven”).
  - Meses del año.
  - Palabras derivadas de la misma raíz.
  - Palabras del mismo campo semántico (“cats”, “elephants”, “rat”, “box”...).

9. Visualización y análisis de embeddings

- **Técnicas de visualización:**

  Principal Component Analysis

  Algoritmos de reducción de dimensionalidad como PCA o t-SNE permiten ver en 2D o 3D cómo se agrupan los embeddings.

- **Interpretación:**

  Clusters semánticos, agrupación de palabras con relación sintáctica o de raíz, y ejemplos de palabras con múltiples significados (“python” puede estar cerca de informática y de humor, por Monty Python).

10. Embeddings más allá del texto

- **Generalización:**

  Los embeddings no son exclusivos del texto. Cualquier variable categórica (genes, jugadas de ajedrez, etc.) se puede representar mediante embeddings, reemplazando vectores dispersos (one-hot) por vectores densos que capturan relaciones relevantes.

11. Propiedades algebraicas y cliffhanger

- **Espacio matemático con operaciones:**

  En el espacio de embeddings se pueden hacer operaciones algebraicas (por ejemplo: “rey” - “hombre” + “mujer” ≈ “reina”), aunque en la práctica esto tiene matices.

Resumen y conclusiones

- El paso de one-hot a embeddings es un salto fundamental en NLP moderno.
- Los embeddings permiten comprimir y capturar la similitud semántica entre palabras.
- Usar embeddings pre-entrenados es una de las claves para mejorar el rendimiento y la generalización en tareas de procesamiento del lenguaje.
- Las ideas de embeddings son transferibles a otros dominios más allá del texto.

Introducción intuitiva a los Transformes y la evolución de las Redes Neuronales en NLP

1. Introducción y Contexto

- **Breve historia del Deep Learning en NLP**
  - Rápida evolución en la última década: de modelos simples a arquitecturas avanzadas.
  - Uso de redes neuronales multicapa en los primeros modelos.
  - Adaptación de las arquitecturas a la naturaleza de los datos:
    - Imágenes → Redes Convolucionales (CNN).
    - Texto → Redes Recurrentes (RNN).
  - Avances recientes: modelos generativos y de análisis con resultados espectaculares en IA.
- **Revolución de 2017: el surgimiento de los Transformers**

- Aparece el paper “Attention is All You Need”.
- Cambia radicalmente el paradigma: nuevos récords en tareas de NLP, visión por computador y más.
- Ejemplos de aplicación: AlphaFold 2 (genómica), Tesla Autopilot (visión), GPT-3 (texto), VQGAN (arte).

---

2. Procesamiento del Lenguaje Natural antes de los Transformers

2.1 El reto de procesar texto: naturaleza secuencial

- El texto como secuencia: cada palabra tiene una posición y depende del contexto anterior.
- Estrategia tradicional:
  - Input de la red = primera palabra.
  - Salida procesada capa a capa.
  - El output de la red se concatena como input de la siguiente palabra, y así sucesivamente.

2.2 Redes Neuronales Recurrentes (RNN)

- **Funcionamiento básico**
  - Output de cada paso alimenta el input del siguiente (“efecto ciempiés”).
  - Permiten analizar secuencias, manteniendo información del contexto anterior.
- **Analogía humana**:
  - Cuando leemos, procesamos palabra a palabra, pero apoyándonos en el contexto de las anteriores.
- **Usos típicos**:
  - Generadores de texto, traductores automáticos, análisis de secuencias (por ejemplo, acciones en videojuegos como Dota 2).

2.3 Limitaciones de las RNN

- **Falta de memoria a largo plazo**

- Problema: Las primeras palabras “se olvidan” a medida que avanza la secuencia.
- Consecuencia: No se detectan relaciones importantes entre palabras distantes (ejemplo: “el pangolín dormía... usando su cola” → la relación “su” ↔ “pangolín” se pierde).

- **Pregunta crítica**:
  - ¿Cuál fue la primera palabra del vídeo? Si no lo recuerdas, lo mismo le pasa a las RNN.

---

3. Solución: El Mecanismo de Atención

Paper: "All you need is attention"

3.1 Origen y necesidad

- Los mecanismos de atención surgen como solución al “olvido” en las RNN.
- Objetivo: Que cada palabra pueda relacionarse con cualquier otra, sin importar la distancia en la secuencia.

3.2 Representación de palabras en NLP

- **Palabras → vectores numéricos**
  - Cada palabra es representada como un vector multidimensional (embeddings).
  - Vectores similares representan palabras semánticamente relacionadas.
  - Ejemplo: “rey” y “reina” estarán cerca en el espacio de embeddings.

3.3 ¿Cómo automatizar la búsqueda de relaciones?

- **Entrenamiento dual: Queries y Keys**
  - Se entrenan dos redes para cada palabra:
    - Una genera un **vector query** (lo que busca la palabra).
    - Otra genera un **vector key** (lo que ofrece la palabra).
  - Analogía:
    - Tinder (perfil y lo que buscas) o
    - Llave y cerradura (cada llave puede abrir varias cerraduras con mayor o menor compatibilidad).

- **Compatibilidad entre palabras**
  - Se calcula el producto escalar (dot product) entre los vectores query y key.
  - Si dos vectores están alineados → mayor compatibilidad (atención).
  - Esto genera un **vector de atención** (a qué palabras presta atención cada palabra).
- **Visualización: Matriz de atención**
  - Se obtiene una matriz que muestra la importancia relativa entre todas las palabras de la frase.
  - Muy útil para analizar cómo el modelo distribuye el contexto.

---

4. El Mecanismo de Atención en Acción

4.1 ¿Por qué calcular la atención?

- Para dar el contexto adecuado a cada palabra según las relaciones más relevantes en la frase.
- A cada palabra se le asigna un **vector valor** (“value”), que es procesado junto con los factores de atención.
- Se realiza una **suma ponderada**:
  - El output de cada palabra es la mezcla de los valores del resto, ponderados por su atención.

4.2 Ventajas frente a las RNN

- Las palabras pueden contextualizarse con cualquier otra, sin importar la distancia.
- **El problema de la “memoria a largo plazo” está resuelto.**
- Permite relaciones complejas y contextos flexibles.

---

5. El Salto de Calidad: Transformers

5.1 El paper “Attention Is All You Need”

- Propone eliminar la parte recurrente y trabajar **sólo** con mecanismos de atención.
- Nace la arquitectura Transformer:
  - No es ni una red recurrente, ni una convolucional.
  - Solo utiliza atención para construir el contexto.

5.2 Ventajas de los Transformers

- Superan a las RNN y LSTM en tareas de NLP.
- Permiten paralelizar el procesamiento (más rápido, más eficiente).
- Se adaptan a tareas de visión (Visual Transformers), generación de texto (GPT), traducción, etc.

La magia de los Transformes y la codificación posicional

1. Introducción a los Transformers y su impacto

- **Evolución de los modelos de lenguaje**:
  - Del procesamiento secuencial (RNN, LSTM) a modelos paralelizables (Transformers).
- **Modelos destacados**:
  - GPT-3 vs Megatron Turing: volumen de parámetros, capacidades y potencial.
- **Aplicaciones modernas basadas en Transformers**:
  - NLP (GPT-3, Megatron), visión (DALL·E, CLIP), ciencia (AlphaFold).
- **Pregunta guía**: ¿Qué hace a los Transformers tan eficaces?

---

2. Elementos clave de un Transformer

- **Revisión del paper "Attention is All You Need"**:
  - Arquitectura encoder-decoder.
  - Introducción al diagrama estructural clásico.

- **Atención como primer componente fundamental**:
  - ¿Qué es y cómo ayuda a centrar el foco en partes relevantes de la secuencia?
- **Planteamiento crítico**: *¿Y si la atención no fuera el ingrediente más importante?*

---

3. El problema del orden en Transformers

- **Diferencia con redes recurrentes**:
  - Procesamiento secuencial vs. procesamiento en paralelo.
- **Consecuencia de la paralelización**:
  - Pérdida de información sobre el orden de los tokens.
- **Ejemplo ilustrativo**:
  - Frases con las mismas palabras en diferente orden ⇒ distinto significado.

---

4. Soluciones propuestas para la codificación posicional

- **Primera idea: Posicionamiento absoluto**:
  - Añadir un vector numérico con el índice de la posición (1, 2, 3...).
  - Problema: distorsión del vector original por valores grandes.
- **Segunda idea: Normalización entre 0 y 1**:
  - Vector con fracción de posición (i/n).
  - Problema: ambigüedad entre secuencias de diferentes longitudes.
- **Tercera idea: Codificación binaria discreta**:
  - Representar las posiciones en binario.
  - Observación de patrones: frecuencia en los bits.

---

5. De lo discreto a lo continuo: Senos y cosenos

- **Motivación**:

- Los Transformers son redes neuronales *continuas*, no discretas.

- **Codificación posicional sinusoidal**:
  - Ondas a diferentes frecuencias para representar patrones de posición.
  - Analogía con alternancia de bits binarios.
- **Fórmula general (sin entrar en derivaciones)**:
  - Se usan funciones seno y coseno de distinta frecuencia para cada componente del vector.
- **Ventajas**:
  - Captura el orden de manera continua, eficiente y sin distorsionar los embeddings.
  - Permite extrapolar a secuencias más largas (generalización).

---

6. Implicaciones prácticas y técnicas

- **Paralelización del entrenamiento**:
  - Mayor rendimiento en GPUs/TPUs.
  - Entrenamiento de modelos masivos como GPT-3 o Megatron-Turing.
- **Ventaja frente a RNNs**:
  - RNNs tienen dependencia secuencial, impiden procesamiento paralelo.
  - Transformers procesan toda la secuencia simultáneamente gracias a la codificación posicional.

---

7. Reflexión final y conclusión

- **La codificación posicional no es solo un detalle técnico**:
  - Es el componente que permite a los Transformers “entender” secuencias en contexto.
- **Importancia dentro del Transformer**:
  - Junto con la atención, hace posible la revolución en NLP y deep learning.
- **Cierre con visión de futuro**:

- Nuevas aplicaciones como AIs que programan (tema del siguiente vídeo).
- Evolución de arquitecturas basadas en codificación posicional aprendida.

Materiales complementarios recomendados

- Enlace al paper original: *"Attention is All You Need"*.
- Artículo técnico sobre codificación sinusoidal (mencionado en el vídeo).
- Ejercicios prácticos:
  - Implementar codificación posicional sinusoidal en NumPy o PyTorch.
  - Comparar distintas codificaciones en un mini Transformer.

De los primeros clasificadores a los modelos funcionales en análisis de sentimientos

1. Introducción al problema del análisis de sentimientos

Objetivo: Comprender el análisis de sentimientos como una tarea de clasificación binaria (positivo vs. negativo) o multicategórica (e.g., 1 a 5 estrellas).

- Aplicación práctica: análisis de sentimientos en tweets o reseñas.
- Enfoque clásico: diccionario de polaridades positivas y negativas.
- Limitaciones: frases irónicas, contexto, ambigüedad semántica.

2. Primeros enfoques: análisis basado en polaridad léxica

Metodología:

- Uso de listas de palabras con polaridad asignada (positiva/negativa).
- Suma de puntuaciones para determinar el sentimiento general.

Ejemplo:

- Frase irónica: “Esta compra es muy recomendable si quieres tirar el dinero.”
- Problema: análisis superficial sin comprensión del contexto.

3. Evolución hacia el aprendizaje supervisado

Cambio de paradigma:

- Uso de modelos entrenados con datos etiquetados manualmente.
- Empleo de redes neuronales recurrentes (RNN) para capturar secuencias.

Implicaciones:

- Necesidad de datasets grandes con anotaciones humanas.
- Ejemplo: clasificación de tweets con etiquetas positivas o negativas.

4. Limitaciones del aprendizaje supervisado

- Cada nueva tarea requería un nuevo dataset y entrenamiento desde cero.
- Problemas de escalabilidad y coste en tareas específicas.

5. Transfer Learning: inspiración desde la visión por computador

Idea clave:

- Utilizar modelos preentrenados para tareas nuevas con poco dato etiquetado.
- Ejemplo: reconocer coches o gatos a partir de modelos entrenados previamente.
- Analogía: aprender a leer una radiografía sin aprender a ver desde cero.

6. Transformadores (Transformers) y el aprendizaje auto-supervisado

Hitos importantes (2017 en adelante):

- Aparición de modelos como BERT y GPT.
- Técnicas de preentrenamiento auto-supervisado:

    - **Máscara de palabras** (BERT).
    - **Autocompletado de texto** (GPT).
- Ventaja: uso de texto masivo sin necesidad de etiquetado humano.
- Paso de aprendizaje supervisado a auto-supervisado.

7. Democratización del acceso a modelos

- Publicación abierta de modelos preentrenados.
- Comunidad Hugging Face y su librería `transformers`.
- Pipeline automático para tareas como análisis de sentimiento, clasificación, etc.
- Uso práctico: análisis de frases directamente con modelos como DistilBERT.

8. Fine-tuning: adaptación de modelos preentrenados

Concepto:

- Aprovechar modelos ya entrenados y ajustarlos a datos específicos.
- Menor coste computacional vs. entrenamiento desde cero.
- Mejora del rendimiento en dominios particulares.

9. La era de los modelos fundacionales

GPT-3 y el efecto escala:

- Entrenamiento con 175B parámetros.
- Capacidad emergente de realizar tareas no explícitamente entrenadas:
    - Traducción, clasificación, chatbot, generación de código.
    - Ejemplo: traducir una frase o clasificar sentimientos sin entrenamiento específico.
- Entrada en escena de tareas con **zero-shot learning** o **few-shot learning**.

10. Aplicación creativa del prompting

Prompt engineering básico:

- Dar ejemplos al modelo para que infiera la tarea:
  - “Esta frase tiene sentimiento positivo.”
  - “Detesto tener que hacer esto.” → modelo infiere 1 estrella.
- Clasificación temática: frases de Harry Potter vs. El Señor de los Anillos.

11. Reflexión final y visión hacia el futuro

- El análisis de sentimientos sirve como excusa para mostrar el cambio radical en NLP.
- Modelos generalistas capaces de adaptarse a múltiples tareas mediante escalado.
- Próximo paso: modelos aún más grandes como PaLM (Google), con nuevas propiedades emergentes.