# IA - Clase 1A

## Introducción
## Repaso de Matemáticas
## Conceptos de Aprendizaje de Máquina
## (ML – Machine Learning)

# Repaso de Matemáticas

- Función vectorial: $f(x, y) = x^\top Ay + x^\top Bx - Cy + D$

con:

- $x \in \mathbb{R}^M$ (vector columna de dimensión $M$)
- $y \in \mathbb{R}^N$ (vector columna de dimensión $N$)
- $f : \mathbb{R}^M \times \mathbb{R}^N \to \mathbb{R}$ (escalar).

- Análisis de cada término

$x^\top Ay$

$x^\top$ es de tamaño $1 \times M$.

$y$ es de tamaño $N \times 1$.

Para que el producto sea válido, $A$ debe "conectar" $M$ con $N$.

Es decir: $A \in \mathbb{R}^{M \times N}$.

Entonces:

$$(1 \times M)(M \times N)(N \times 1) = (1 \times 1) \quad (\text{escalar}).$$

# Repaso de Matemáticas

- Análisis de cada término

$x^\top Bx$

$x^\top$ es $1 \times M$.

$x$ es $M \times 1$.

Para que funcione, $B \in \mathbb{R}^{M \times M}$.

Entonces:

$$(1 \times M)(M \times M)(M \times 1) = (1 \times 1).$$

$-Cy$

$y$ es $N \times 1$.

Para que dé un escalar, $C$ debe ser $1 \times N$.

Entonces:

$$(1 \times N)(N \times 1) = (1 \times 1).$$

$D$

Como es una constante sumada a escalares, necesariamente debe ser un **escalar**.

$$D \in \mathbb{R}.$$

# Repaso de Matemáticas

- Ejemplo numérico con M=2 N=3
  - Vectores:

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

  - Matrices y escalar:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & -1 \\ 0 & 3 \end{bmatrix}, \quad C = \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}, \quad D = 10$$

**Cálculos paso a paso**

1. $x^\top Ay = 150$
2. $x^\top Bx = 12$
3. $-Cy = -98$
4. $D = 10$

**Resultado final**

$$f(x, y) = 150 + 12 - 98 + 10 = 74$$

# Repaso de Matemáticas

- Ejemplo numérico con M=2 N=3

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

**Paso 1.1:** $Ay$

$$Ay = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \cdot 3 + 2 \cdot 4 + 3 \cdot 5 \\ 4 \cdot 3 + 5 \cdot 4 + 6 \cdot 5 \end{bmatrix} = \begin{bmatrix} 26 \\ 62 \end{bmatrix}$$

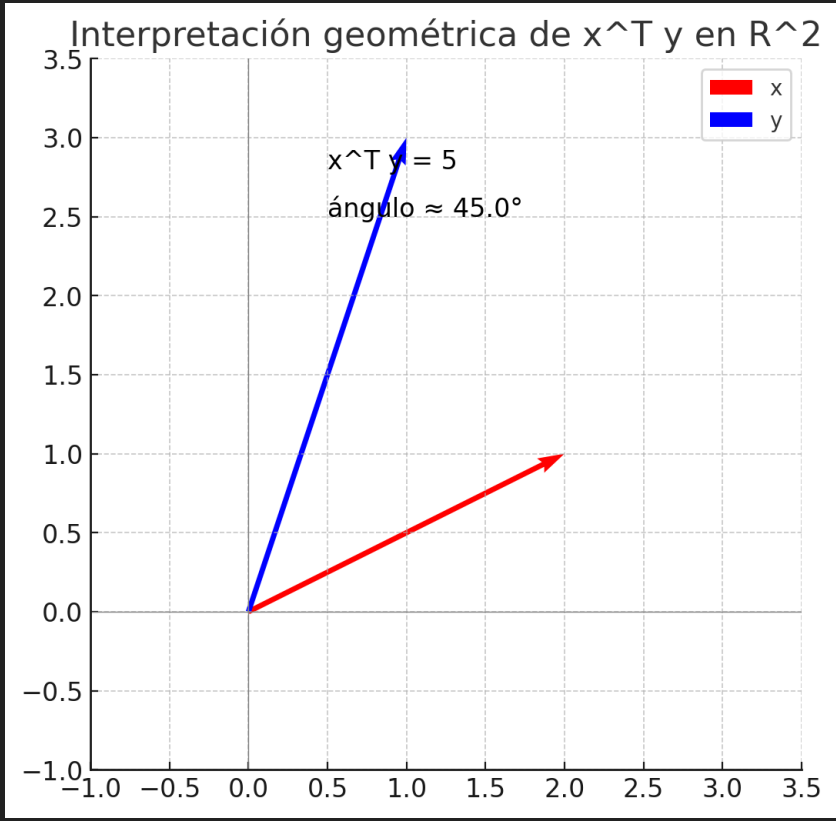
**Paso 1.2:** $x^\top (Ay)$

$$x^\top (Ay) = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 26 \\ 62 \end{bmatrix} = 1 \cdot 26 + 2 \cdot 62 = 150$$

# Repaso de Matemáticas

- Análisis de cada término
  - $x^T$
    - La transpuesta de un vector/matriz consiste en cambiar filas ↔ columnas.
    - Si x es un vector columna (M×1), entonces:
    - $x^T=[x1\ x2\cdots xM] \in R1\times M$
  - Ahora $x^T$ es un vector fila.
  - Si multiplicáramos x·y directamente, no siempre tendría sentido porque ambos son vectores columna.
  - Pero al usar $x^T$ y tenemos:
    - $(1\times M)(M\times 1)=(1\times 1)$,
    - es decir, un escalar: justamente el producto interno o producto punto de dos vectores.
  - $x^Ty$ es el producto escalar entre x e y, mide qué tanto apuntan en la misma dirección. De hecho:
  - $x^Ty=\|x\|\|y\|\cos(\theta)$, donde θ es el ángulo entre x e y.

# Repaso de Matemáticas




**Cálculo exacto con** $x = (2, 1), y = (1, 3)$

1. Producto escalar:

$$x^\top y = 2 \cdot 1 + 1 \cdot 3 = 5$$

2. Normas:

$$\|x\| = \sqrt{2^2 + 1^2} = \sqrt{5}, \quad \|y\| = \sqrt{1^2 + 3^2} = \sqrt{10}$$

3. Coseno del ángulo:

$$\cos(\theta) = \frac{5}{\sqrt{5}\sqrt{10}} = \frac{5}{\sqrt{50}} = \frac{5}{7.071} \approx 0.707$$

4. Ángulo:

$$\theta = \arccos(0.707) \approx 45^\circ$$

En 2D y 3D podemos visualizar los vectores y el ángulo.

En 4D o más, no podemos dibujar, pero el producto escalar y la fórmula del coseno nos siguen diciendo si los vectores son "cercanos" (ángulo pequeño), "ortogonales" (θ=90°) u "opuestos" (θ>90°).

# Repaso de Matemáticas

- El vector es uno de los conceptos matemáticos más importantes en Inteligencia Artificial (IA) y en particular en Machine Learning (ML) y Deep Learning (DL).
- Representación de datos
  - En IA, todo se representa como vectores: una imagen en escala de grises 28×28 (como MNIST) se convierte en un vector de dimensión 784
  - Una palabra (ej. "gato") se representa como un vector en un espacio semántico (Word Embeddings como Word2Vec, GloVe, o BERT).
  - Una fila de una base de datos con variables (edad, peso, altura, ingresos) también se guarda como un vector de características (feature vector).
  - El vector es la forma estándar de pasar la información al modelo.

# Repaso de Matemáticas

- Operaciones básicas en modelos
- Los algoritmos de IA dependen de operaciones con vectores:
  - Producto escalar ($x^T y$): mide similitud entre dos vectores (coseno del ángulo → usado en búsqueda semántica, recomendación).
  - Norma (tamaño, longitud o magnitud $\|x\|$): mide la magnitud de un vector → usado en regularización, distancias.
  - Distancia Euclidiana ($\|x-y\|$): mide cuán distintos son dos vectores → en KNN, clustering, embeddings.
- Redes neuronales
  - En redes neuronales profundas una capa densa hace :
    - $z=Wx+b$
      - x es el vector de entrada,
      - W es la matriz de pesos,
      - b es el vector de sesgo,
      - z es el nuevo vector de salida.
  - Cada neurona es un producto escalar entre el vector de pesos y el vector de entrada.

# Repaso de Matemáticas

- Geometría de la IA
  - En espacios de alta dimensión, cada vector representa un punto.
  - Clasificar, agrupar o predecir significa separar y organizar vectores en ese espacio.
    - Ejemplo: un clasificador lineal busca un hiperplano (definido por un vector normal) que divida los vectores de dos clases.

- Interpretación semántica
  - En IA moderna (NLP *Procesamiento de Lenguaje Natural*, visión, audio):
    - Dos vectores cercanos → significan cosas similares (ej. "rey" y "reina").
    - Dos vectores ortogonales → no relacionados.
    - Dos vectores opuestos → significados contrarios.
  - Esto se usa en embeddings para búsquedas, traducción automática, chatbots, etc.

# Repaso de Matemáticas
# Analicemos un ejemplo

- NLP = IA aplicada al lenguaje humano.
  - Representa palabras/oraciones como vectores en espacios de alta dimensión.
  - Se aplica en traducción, chatbots, resumen, análisis de sentimiento, búsqueda semántica.
  - Los Transformers revolucionaron NLP → base de los LLM como ChatGPT.
- Ver ejemplo en Python en:
  - *IA_TP1A_ML_Aprendizaje _Máquina_R1.ipynb*

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - Representaciones vectoriales de palabras, frases o documentos en un espacio de dimensión finita ($R^d$).
  - Intuición: convierten texto en números que capturan similitud semántica.
    - Ejemplo: “gato” → [0.2,−0.5,0.8,0.1]
    - “perro” → cercano a “gato” en el espacio.
  - Uso: entradas para modelos de NLP, búsqueda semántica, clasificación.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - Por qué el vector “gato” → [0.2,−0.5,0.8,0.1] tiene 4 valores en vez de uno ?
  - Supongamos que representamos cada palabra con un único valor:
    - "gato" → 0.2
    - "perro" → 0.5
    - "avión" → 0.9
  - Problema: con 1 dimensión solo tenemos una recta, y no podemos capturar similitudes complejas.
  - Ejemplo: ¿dónde pongo "león"? ¿Más cerca de “perro” o de “avión”?
  - Con un solo número es imposible reflejar múltiples relaciones.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - Con varias dimensiones (ejemplo 4D)
  - Al usar un vector en $R^d$ cada coordenada puede capturar un aspecto diferente (aunque no interpretable directamente):
    - "gato" → [0.2,−0.5,0.8,0.1]
    - "perro" → [0.1,−0.6,0.7,0.0]
    - "avión" → [−0.9,0.2,−0.1,0.5]
  - “gato” y “perro” quedan cerca en el espacio (vectores similares), pero “avión” queda lejos.
  - No es que la primera coordenada sea “animal” y la segunda “tamaño”.
  - Más bien, el conjunto completo del vector es lo que captura el significado.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - Dimensiones típicas
    - Word2Vec → 50 a 300 dimensiones.
    - GloVe → 100 a 300.
    - BERT / GPT → 768, 1024 o más.
  - Cuantas más dimensiones → más capacidad para representar relaciones semánticas finas, aunque también más costo de cómputo.
  - Imaginar que cada palabra es un punto en un espacio de alta dimensión.
    - Distancia pequeña → significados cercanos ("gato", "perro").
    - Distancia grande → significados lejanos ("gato", "avión").
    - Direcciones → relaciones semánticas ("rey - hombre + mujer ≈ reina").

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - Una palabra no se puede describir con un solo número porque el lenguaje tiene muchos matices.
  - Un embedding de varias dimensiones permite capturar múltiples aspectos de significado.
  - No miramos cada número por separado → lo importante es la posición relativa de los vectores en el espacio.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - ¿Cada elemento del vector representa una caracteristica (cercana o lejana)?
  - Cada elemento del vector es una coordenada.
  - En un embedding, la palabra se representa como un punto en un espacio de muchas dimensiones.
  - Cada número es una coordenada en ese espacio (igual que un punto en 2D tiene (x,y) pero aquí hay cientos de coordenadas).
  - Ejemplo en 2D (fácil de visualizar): (en 4D o más ya es imposible visualizarlos)
    - "gato" → (2, 1)
    - "perro" → (2.2, 1.1)
    - "avión" → (-3, 5)
    - “Gato” y “perro” quedan cerca en el plano, “avión” queda lejos.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - ¿Cada coordenada = representa una característica?
  - No necesariamente.
    - No es que el primer número sea “animal”, el segundo “pelaje” o el tercero “doméstico”.
  - Los modelos no aprenden significados humanos directos, sino patrones estadísticos del lenguaje.
  - Lo que buscamos es la combinación de todas las coordenadas que ubica cada palabra en un lugar donde la distancia y la dirección reflejan similitud semántica.
  - Si solo tenemos 1 dimensión (una línea), solo podemos decir “más a la izquierda” o “más a la derecha”.
  - Con 2 dimensiones (un plano), ya podemos distinguir norte-sur y este-oeste.
  - Con 300 dimensiones, cada objeto (palabra) se ubica de forma que refleja muchos matices de similitud al mismo tiempo.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Transformers
  - Arquitectura de red neuronal (2017, “Attention is All You Need”) que revolucionó NLP.
  - Usa mecanismo de atención para procesar texto en paralelo y capturar relaciones de largo alcance.
  - Ejemplo: en “El gato duerme en la cama”, el modelo entiende que “cama” se relaciona más con “duerme” que con “El”.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- BERT (Bidirectional Encoder Representations from Transformers)
  - Modelo de lenguaje basado en la parte encoder del Transformer (Google, 2018).
  - Bidireccional: mira a la izquierda y a la derecha del token al mismo tiempo.
  - Ejemplo:
    - Frase: “El ___ duerme en la cama”
    - BERT puede predecir que la palabra faltante es “gato”.
    - Uso: clasificación, extracción de entidades, búsqueda semántica, Q&A.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Loss (Función de pérdida)
  - Medida de qué tan mal lo está haciendo el modelo.
  - Matemáticamente: diferencia entre predicción y valor real.
  - Ejemplo:
    - Predicción del modelo: “positivo” con 0.7 de confianza.
    - Etiqueta real: “positivo” (=1).
    - Loss (ej. cross-entropy) = número positivo → mientras más chico, mejor.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Backpropagation
  - Algoritmo para ajustar los pesos de la red.
  - Proceso:
    - Calcular la pérdida (loss).
    - Propagar hacia atrás el gradiente $(\frac{\partial \text{loss}}{\partial w})$.
    - Ajustar cada peso en dirección de menor error.
  - Ejemplo intuitivo:
    - como aprender a tirar flechas → cada vez que se falla al blanco se corrige el lanzamiento en la dirección contraria al error.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Gradiente
  - Pieza importante del aprendizaje en redes neuronales (incluyendo embeddings, Transformers, BERT, etc.).
  - Matemáticamente: el gradiente de una función f(w) respecto a sus parámetros w es un vector de derivadas parciales.

$$\nabla_{\mathbf{w}} f = \left[ \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \ldots, \frac{\partial f}{\partial w_n} \right]$$

  - En ML: mide cómo cambia la pérdida (loss) si ajusto un poquito cada peso.
  - El gradiente apunta en la dirección de máxima subida de la función de pérdida.
  - Para minimizar la pérdida, caminamos en la dirección opuesta al gradiente.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Gradiente
  - Ejemplo con embeddings. Supongamos un embedding simple en 2D para la palabra "gato":

$$\text{vec}(\text{"}gato\text{"}) = (0.2, 0.5)$$

  - Si el modelo predice mal, la pérdida $\mathcal{L}$ depende de esos valores.
  - El gradiente puede resultar en algo como:

$$\nabla \mathcal{L} = \left( \frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2} \right) = (-0.3, 0.1)$$

  - Si aumento $w_1$, la pérdida sube (porque el gradiente es negativo, conviene restar y hacerlo crecer).
  - Si aumento $w_2$, la pérdida baja (gradiente positivo, conviene restar y hacerlo más chico).

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Gradiente
  - Regla básica de descenso por gradiente: $w_{\text{nuevo}} = w_{\text{viejo}} - \eta \cdot \nabla \mathcal{L}$
  - Ejemplo numérico:
    - w=(0.2,0.5)
    - Gradiente = (−0.3,0.1)
    - Learning rate = η=0.1
    - $W_{\text{nuevo}}$=(0.2,0.5)−0.1·(−0.3,0.1)=(0.23,0.49)
  - El embedding de “gato” se mueve en el espacio vectorial para reducir la pérdida.
  - El gradiente dice cómo modificar cada peso para que el modelo se equivoque menos.
  - Como un “mapa de pendientes” que guía al entrenamiento.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Optimizador
  - Regla que usan los gradientes para actualizar los pesos.
  - Ejemplo:
    - Regla básica (SGD): $w_{\text{nuevo}} = w_{\text{viejo}} - \eta \cdot \frac{\partial \mathcal{L}}{\partial w}$
    - η es la tasa de aprend
    - η es un número positivo que controla la velocidad de aprendizaje del modelo.
    - Es un hiperparámetro → valores que decidimos antes del entrenamiento y que controlan cómo aprende el modelo.
    - Se define por diseño. En trabajos de NLP con Adam, se suele usar $\eta = 10^{-3}$(0.001) como valor inicial recomendado.
      - Puede ajustarse experimentalmente según el comportamiento del entrenamiento.
      - O usar usan técnicas como grid search, random search o Bayesian optimization para elegirlo automáticamente.

η Muy baja → lento.
η Muy alta → inestable.
η Óptima → balance entre velocidad y estabilidad.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Adam (Adaptive Moment Estimation)
  - Un optimizador avanzado muy usado en Deep Learning.
  - Adapta automáticamente la tasa de aprendizaje para cada peso usando promedios de gradientes (momentum).
  - Intuición: en lugar de dar pasos iguales, da pasos “inteligentes” → más grandes en direcciones seguras, más chicos donde hay oscilaciones.
  - Entrena redes neuronales más rápido y estable que SGD puro.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Paso1 - Entrada: Texto en lenguaje humano
  - "El gato duerme en la cama"
- Paso 2 - Tokenización
  - El texto se convierte en tokens (piezas de palabras).
  - Con tokenizador de sub-palabras (BPE, WordPiece).
  - Cada token se asocia a un índice en un vocabulario.
  - ["El", "gato", "duer", "me", "en", "la", "cama"]
- Paso 3 - Embeddings
  - Cada token se transforma en un vector en $R^{d}$.. Ejemplo (d=4)
  - vec("*gato*")=[0.23,−0.11,0.87,0.45]
  - La oración es ahora 1 matriz: cada fila es un token, cada columna una dimensión.

# Repaso de Matemáticas
# Analicemos el Paso 3
# ¿Primero el "Huevo o la Gallina" ?

- Esto va al corazón de cómo un modelo de NLP aprende representaciones de palabras (embeddings).
- Embeddings (vectores de palabras)
  - Cuando transformamos un token en un vector en $R^d$, cada palabra queda representada como: vec("gato")=$[w_1,w_2,\ldots,w_d]$
  - Los números $w_i$ son los pesos que definen la posición de la palabra en el espacio vectorial.
- Entonces: ¿Cómo se definen esos pesos?
  - 2 grandes enfoques:
    - Embeddings preentrenados (no se entrena desde cero)
    - Embeddings entrenados junto con el modelo

# Repaso de Matemáticas
# Analicemos el Paso 3
# ¿Primero el “Huevo o la Gallina” ?

- Embeddings preentrenados (no se entrena desde cero)
  - Se entrenan sobre grandes corpus de texto (Wikipedia, noticias, libros).
    - Ejemplos: Word2Vec, GloVe, FastText, BERT embeddings.
  - Cada palabra obtiene un vector que refleja similitud semántica.
    - “gato” y “perro” tendrán vectores cercanos.
    - “gato” y “avión” estarán lejos.
  - Estos pesos vienen “listos” y se usan en el modelo como tablas de búsqueda.
  - Los pesos ya están definidos por otro entrenamiento.(¿?)

# Repaso de Matemáticas
# Analicemos el Paso 3
# ¿Primero el “Huevo o la Gallina” ?

- Embeddings entrenados junto con el modelo
  - En redes neuronales (ej. Transformer, LSTM), los embeddings son parámetros inicializados aleatoriamente.
  - Al entrenar la red, los pesos se ajustan mediante backpropagation.
  - Se aprende automáticamente qué valores de los vectores hacen que el modelo minimice la pérdida (loss).
  - Aquí el modelo define los pesos a medida que aprende la tarea.
- Cómo se almacenan?

$$E \in \mathbb{R}^{|V| \times d}$$

  - Se usa una matriz de embeddings:
    - $|V|$ = tamaño del vocabulario (cantidad de palabras/tokens).
    - $d$ = dimensión de cada vector.
    - Cada fila $E_i$ corresponde al embedding del token i.
  - Cuando se encuentra un token en el texto, se reemplaza por su fila correspondiente en *E.*

# Repaso de Matemáticas
# Analicemos el Paso 3
# ¿Primero el “Huevo o la Gallina” ?

- Ejemplo: *(|V|=5,d=3)*

$$E = \begin{bmatrix} 0.2 & -0.1 & 0.9 \\ \text{(token 1: ”el”)} & & \\ -0.3 & 0.5 & 0.7 \\ \text{(token 2: ”gato”)} & & \\ 0.8 & 0.1 & -0.4 \\ \text{(token 3: ”duerme”)} & & \\ \cdots & & \end{bmatrix}$$

  - Los pesos no tienen un significado individual.
    - El conjunto completo de coordenadas es lo que captura relaciones semánticas y sintácticas.
    - Ejemplo: *vec("rey")−vec("hombre")+vec("mujer")≈vec("reina")*
  - Esto se debe a la forma en que el entrenamiento ajusta esos pesos en miles de dimensiones.
  - Los “pesos” de cada palabra = sus coordenadas en el espacio vectorial. Se obtienen de una matriz de embeddings.
  - Pueden venir preentrenados (Word2Vec, GloVe, BERT) o se aprenden durante el entrenamiento.
  - Capturan significados relativos (similitud, analogía), no un “atributo humano” en cada coordenada.

# Repaso de Matemáticas
# Analicemos el Paso 3
# ¿Primero el “Huevo o la Gallina” ?

- ¿ Quién define los pesos del entrenamiento ?
- Aquí tenemos el dilema del huevo y la gallina ...
  - ¿De dónde salen los pesos iniciales de los vectores si todavía no entrenamos?
  - ¿Y cómo sabe el modelo cómo ajustarlos?
- Inicialización de los pesos
  - Al crear una matriz de embeddings $E \in \mathbb{R}^{|V| \times d}$
  - Cada fila (embedding de una palabra) se inicializa con valores aleatorios pequeños (normalmente distribuciones gaussiana o uniforme).
  - Al principio los vectores no tienen ningún significado.
  - Ejemplo (para vocabulario de 5 palabras, dimensión 3):

$$E = \begin{bmatrix} 0.02 & -0.01 & 0.03 \\ -0.04 & 0.05 & -0.02 \\ 0.01 & 0.07 & -0.06 \\ \cdots & & \end{bmatrix}$$

# Repaso de Matemáticas
# Analicemos el Paso 3
# ¿Primero el “Huevo o la Gallina” ?

- Proceso de entrenamiento
  - Aquí entra la “gallina”: el algoritmo de aprendizaje.
  - Se define una función de pérdida (loss).
  - Ejemplo: en un clasificador de sentimiento, penaliza si el modelo predice negativo cuando el texto era positivo.
  - Con cada ejemplo de entrenamiento, el modelo calcula:

$$\text{pérdida} = \mathcal{L}(predicción, etiqueta\ real)$$

  - Se aplica backpropagation:
  - La red calcula las derivadas *de la pérdida respecto a cada peso (incluidos los embeddings).*
  - El optimizador (ej. SGD, Adam) actualiza los pesos un poco en la dirección que reduce la pérdida:

$$w_{\text{nuevo}} = w_{\text{viejo}} - \eta \cdot \frac{\partial \mathcal{L}}{\partial w}$$

  - los vectores empiezan a moverse en el espacio hasta organizarse según la tarea.

# Repaso de Matemáticas
# Analicemos el Paso 3
# ¿Primero el “Huevo o la Gallina” ?

- Resultado del aprendizaje, Después de muchas iteraciones:
  - Palabras que aparecen en contextos similares terminan con vectores cercanos.
  - Palabras opuestas (positivo ↔ negativo) se separan.
  - Relaciones semánticas aparecen “solas” porque ayudan a minimizar la pérdida.
- ¿Entonces, quién define los pesos ?
  - Al principio: nadie → son aleatorios.
  - Durante el entrenamiento: los ajusta el algoritmo de optimización guiado por los datos y la función de pérdida.
  - Al final: los pesos reflejan la estructura estadística del lenguaje en los datos que vio el modelo.
  - El “huevo” son los pesos aleatorios iniciales.
  - La “gallina” es el entrenamiento con datos que va corrigiendo y “dando forma” a esos pesos.

# Repaso de Matemáticas
# Analicemos el Paso 3
# ¿Primero el “Huevo o la Gallina” ?

- Nadie escribe los pesos de las palabras a mano.
- Los embeddings empiezan al azar.
- El entrenamiento supervisado o auto-supervisado los acomoda para capturar significados y relaciones.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Paso 4 - Positional Encoding
  - Como los Transformers no saben de orden por sí mismos, se agrega un vector de posición a cada embedding.
  - Así, el modelo distingue "gato duerme" de "duerme gato".
- Paso 5 - Mecanismo de Atención
  - Cada token genera 3 vectores:
    - Q = Query
    - K = Key
    - V = Value

$$\text{Atención}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

  - La atención mide cuánto “mira” un token a los otros
    - Si el token es "duerme", probablemente preste más atención a "gato". Si el token es "cama", prestará atención a "en la".
  - Esto permite capturar dependencias largas en la oración.

# Repaso de Matemáticas
# Analicemos un ejemplo

- Paso 6 - Capas de Transformer
  - Se aplican varias capas de atención + feed-forward (redes densas).
  - Cada capa produce una representación contextualizada de los tokens.
  - Ejemplo: el vector de "banco" será diferente si hablamos de sentarse o dinero.
- Paso 7 - Salida
  - En GPT (generación de texto):
    - El modelo predice el siguiente token con probabilidad:
      - $P(token_{t+1} | tokens\ anteriores)$
  - En BERT (comprensión de texto): el modelo puede clasificar la oración, rellenar una palabra faltante, responder preguntas, etc.
- Paso 8 - Resultado
  - El Transformer devuelve: texto generado (ej: ChatGPT).
  - Etiquetas de clasificación (ej: sentimiento).
  - Respuestas a preguntas (ej: BERT en Q&A).