

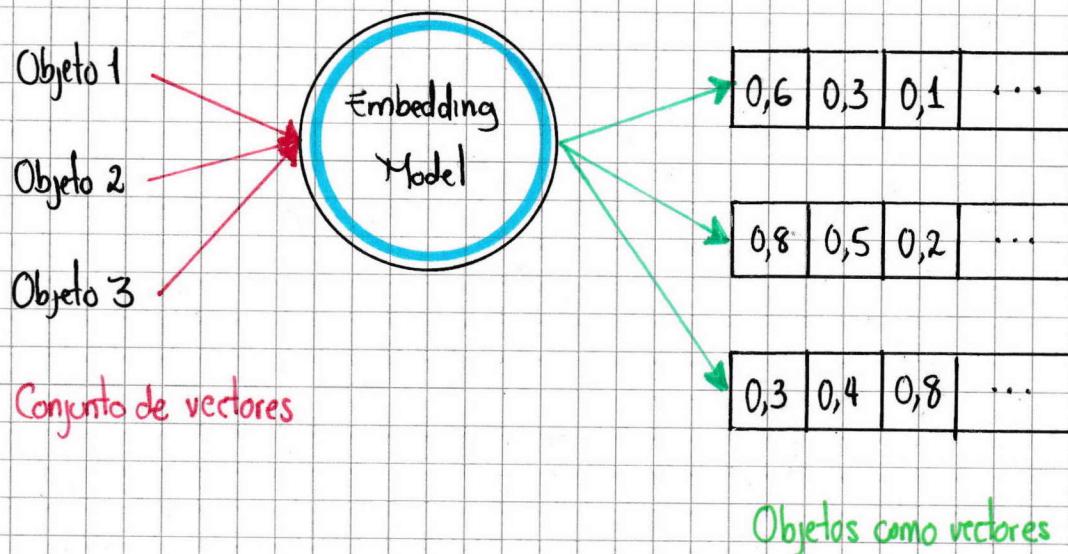
Módulo 1

Procesamiento de texto en el PC

Importancia de los vectores en el aprendizaje automático (ML) y el análisis de datos:

Los vectores son representaciones numéricas de datos, en nuestro caso, texto.

La vectorización, es la forma en que traducimos el texto a un formato que un modelo de aprendizaje automático ~~Machine~~ Learning, puede entender y aprender.



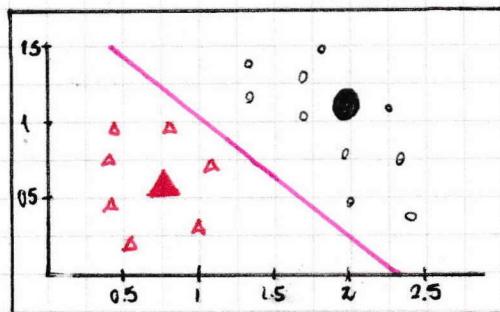
Definición de un vector

- Cantidad que tiene magnitud y dirección.
- Es una matriz de escalares



Utilidades específicas de los vectores en NLP

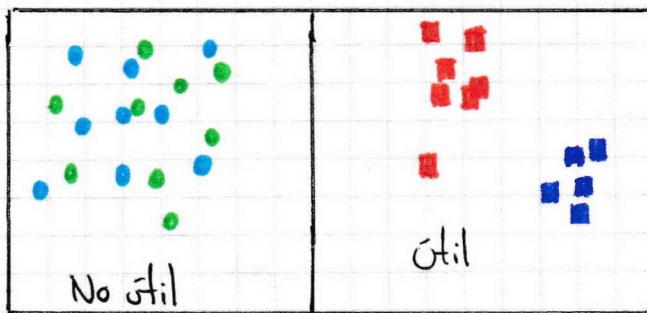
- Detección de spam utilizando vectores



- Organización de documentos mediante el uso de vectores, através de clustering

Desafíos y consideraciones al convertir texto en vectores

- No todas las representaciones vectoriales son útiles
- El objetivo es obtener representaciones útiles.



Bolsa de palabras (BoW - Bag of Words)

Conversion de palabras en números específicos guardados en orden dentro de un paquete de datos.

Maria corre en la mañana
corre la en Maria mañana



Enfoque del NLP

La prevalencia de los enfoques de NLP que no consideran el orden de las palabras.

- La propuesta de una representación numérica del texto que no considera el orden de las palabras: BoW

Importa la presencia o no de las palabras.

Representación de las palabras

- Pérdida de información con el uso de BoW
- Aclaración de los diferentes significados basados en el orden de las palabras.

Juego de salón

Ajedrez

Salón de juegos

Casino

Aplicaciones de BoW

- Uso de BoW en modelos vectoriales y aprendizaje automático clásico.
- Comparación con otros enfoques, como modelos probabilísticos y el aprendizaje profundo.

Análisis de sentimientos
Procesamiento de datos

Valor y eficacia de BoW

- Reflexiones sobre las limitaciones y la precisión del BoW
- Reconocimiento y utilidad en la BoW para el NLP.

ChatGPT no puede usar BoW
SPAM si usa BoW

Conteo de palabras

- Descripción del método de conversión de texto a vectores
- La técnica de conteo como una instancia del enfoque de BoW

Documento

- Definición de documento
- Tipos de documentos

texto, sentimientos en comentarios, palabras spam, etc...

Método de conteo

Documentos

- 1 Me encanta el café. El café me mantiene despierto,
- 2 Me encanta la música. La música me alegra,
- 3 Odio el calor. El calor me molesta

Lista de palabras

Me
encanta
el
café
mantiene
despierto
la
música
alegra
odio
calor
molesta

Modelo de Vector

	1	2	3
me	2	2	1
encanta	1	1	0
el	2	0	2
café	2	0	0
mantiene	1	0	0
despierto	1	0	0
la	0	2	0
música	0	2	0
alegra	0	1	0
odio	0	0	1
calor	0	0	2
molesta	0	0	1

Permite inferir de qué trata cada documento

Tokenerización

Dividir palabras de textos para obtener las individuales. (Tokens)

Mapeo

Posición en el vector de cada palabra para observar la correlación para entregar conclusiones correctas.

Tokenerización

- Dividir el texto en tokens (palabras, caracteres, símbolos)
- Unidades individuales de texto.
- Implementación de los split en python

Ejemplo 1:

```
texto = "Hola, ¿Cómo estás?"  
tokens = texto.split()  
print(tokens)
```

Salida: ['Hola', '¿Cómo', 'estás?']

Diferentes perspectivas para abordar el tema de la tokenización:

Basada en palabras (puerta, ventana, etc.)

Basada en caracteres (a, h, 5, %, etc.)

Basada en subpalabras (automóvil \Rightarrow auto, móvil)

Consideraciones

- Diferentes casos de letras (fildadas)
- Tokenizar por(palabras, caracteres o subpalabras)
- Puntuación (parte de la palabra o token separado)

Volumen de datos

Es crucial para el aprendizaje de los modelos

Manejo de casos (mayus y minus)

La palabra "Hola" no es la misma que "hola" para la máquina.

Ejemplo 2:

```
texto = "Hola, ¿Cómo estás?"  
texto = texto.lower()  
print(texto)  
tokens = texto.split()  
print(tokens)
```

Salida: hola, ¿cómo estas?
['hola', '¿cómo', 'estás?']

Stop Words

- Búsqueda de palabras que carecen de relevancia y no aportan al análisis del texto.
- Ayuda a reducir las dimensiones de los vectores al eliminar las stop words.

Ejemplo 3:

```

→ texto = "El gato es negro y el perro es blanco"
→ import nltk
      nltk.download('stopwords') !pip install nltk
→ from nltk.corpus import stopwords
      from nltk.tokenize import word_tokenize
→ stopWords = set(stopwords.words('spanish'))
      print(stopWords)
→ tokens = word_tokenize(texto)
      print(tokens)
Salida: ['El', 'gato', 'es', 'negro', 'y', 'el', 'perro', 'es', 'blanco']
→ textoFiltrado = [word for word in tokens if not word in stopWords]
      print(textoFiltrado)
Salida: ['El', 'gato', 'negro', 'perro', 'blanco']
→ texto = texto.lower()
      tokens = word_tokenize(texto)
      textoFiltrado = [word for word in tokens if not word in stopWords]
      print(textoFiltrado)
Salida: ['gato', 'negro', 'perro', 'blanco']

```

Existen palabras que se suelen usar en el ámbito académico o empresarial y que no están en la lista de stop_words, así que podemos agregarlas de la siguiente manera:

Ejemplo 4:

```
→ newStopWords = {"ejemplo", "prueba", "hola"}  

stop_words.update(newStopWords)  

print(stop_words)
```

Stemming y Lemmatization

Existen algunos problemas de la tokenización básica y recuento

- Las palabras similares se tratan como entidades separadas
- Las grandes dimensiones de los vectores resultantes
- Aplicaciones prácticas con sus desventajas mencionadas

Caminar, caminando, camina, ...

Para solucionar, usamos **stemming** y **lemmatization**

Stemming: Es una técnica más simple que elimina los sufijos de las palabras

caminando → camina

Lemmatization: Es una técnica más avanzada que utiliza las reglas del lenguaje para obtener la raíz de una palabra.

caminando
caminamos
caminan } → caminar

La lemmatización puede ser más efectiva que el stemming, pero su costo computacional es más alto.

El uso de lematización puede requerir el etiquetado previo.

Aplicación de Stemming y Lemmatization en situaciones reales

Asistentes virtuales y Chatbots

Amazon-Alexa Samsung-Bixby
Chatbot de servicio al cliente
FedEx ⇒ ¿Dónde está mi paquete?

- Análisis de sentimiento

Monitorear X, facebook etc

- Sentimientos en búsquedas (Plato delicioso)
- Sistemas de recomendación (Netflix)

- Motores de búsqueda

Recomendación relacional de búsquedas

- Publicidad Online y etiquetas de redes

Comparación producto.

Ejemplo 5:

La lemmatization no está disponible en español con `nltk`, entonces usaremos `spacy`.

```
+ ! pip install spacy
+ ! python -m spacy download es_core_news_sm
+ import spacy
+ # Cargamos el modelo en español
nlp = spacy.load('es_core_news_sm')
# Creamos el documento
doc = nlp("caminando caminó caminar")
# Imprimimos el texto y el lemma del token
for token in doc:
    print(token.text, "→", token.lemma_)
```

Salida:

caminando	→	caminar
caminó	→	caminar
caminar	→	caminar

Ahora, trataremos de implementar sobre el modelo `spacy` una lemmatization más completa a través de `nltk`.

Ejemplo 6:

```

import spacy
from nltk.tokenize import word_tokenize
import nltk

# Descargamos el modelo en español de spacy
nlp = spacy.load('es_core_news_sm')

# Documento
texto = 'Los niños están corriendo en el parque mientras los perros jugaban.'

# Tokenización con nltk
tokens = word_tokenize(texto)

# Lemmatization con spacy
lemmas = [token.lemma_ for token in nlp(' '.join(tokens))]

print('Texto original:', texto)
print('Texto lematizado:', ' '.join(lemmas))

```

Salida: Texto original: Los niños están corriendo en el parque mientras los perros jugaban.

Texto lematizado: el niño estar correr en el parque mientras el perro jugar .

Ejemplo Práctico: Proyecto 1.pdf Anexo 1

Para desarrollar este ejercicio, necesitamos ingresar a [Kaggle.com](#) y en el dashboard seleccionar [NPL](#) y agregar el filtro [spanish](#) para encontrar la siguiente base de datos

[Spanish News Classification](#)

O simplemente buscar el enlace:

[kaggle.com/datasets/kevinnmorgado/spanish-news-classification](https://www.kaggle.com/datasets/kevinnmorgado/spanish-news-classification)

Descargamos y guardamos en la carpeta del proyecto [Guía_IA_2025](#) de la ubicación raíz de los proyectos [anaconda/pypyter](#).

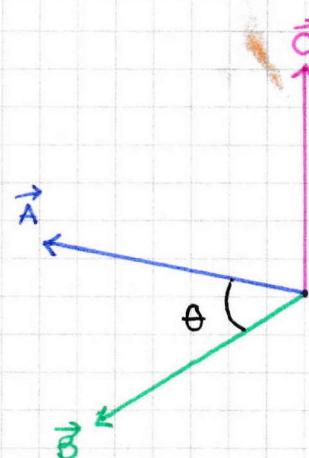
Similitud de vectores

¿Qué es la similitud de vectores?

Similitud de documentos

Clasificar documentos similares por palabras

Calorías → doc de ejercicio



Spinning de artículos y SEO

Posicionamiento en Google, este detecta copias de otros artículos

Recomendaciones

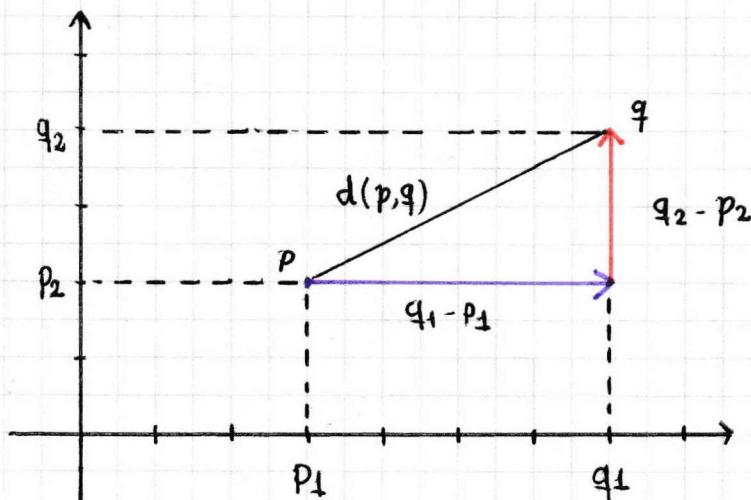
Reviews de Netflix para recomendar

Chatbots

Bosca el vector similar para dar una respuesta asociada a otras.

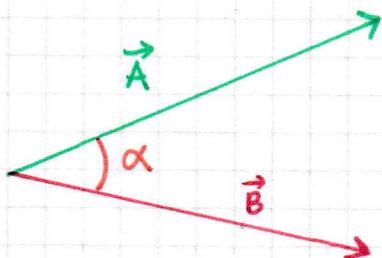
Cálculo de similitud de vectores

Distancia Euclídea



$$d(p,q)^2 = (q_1 - p_1)^2 + (q_2 - p_2)^2$$

Angulo entre vectores



$$\cos \alpha \Rightarrow 1$$

Si $\cos \alpha = 1$ máxima similitud

Si $\cos \alpha = -1$ No hay ninguna similitud

Para similitud exacta $\alpha =$

Comparación entre el ángulo entre vectores y la distancia euclídea

En el NLP, a menudo se prefiere la similitud del coseno porque toma en cuenta el ángulo entre los vectores, que puede ser más relevante en contextos donde los vectores representan texto o palabras.

Entre más similitud, mejor agrupamiento de textos, entre menos similitud, serán más los grupos para clasificar.

Método TF-IDF

Term frequency (TF) e inverse document frequency (IDF)

Consiste en una técnica utilizada en NLP para saber qué tan importante una palabra en un documento de una colección, y a cada palabra se le asigna un puntaje de acuerdo a su frecuencia.

Reduce el peso de las palabras comunes, como las stopwords y aumenta el peso de las que son poco comunes.

Ambigüedad y especificidad de las stopwords dependiendo de la aplicación

No todas las palabras comunes son stopwords en todas las situaciones.

Por ejemplo, no podría ser una de ellas en muchos contextos, pero en el análisis de sentimientos, podría ser una palabra clave.

Función del TF-IDF

Funciona al asignar un score a cada palabra en un documento en función de su frecuencia en ese documento (TF) y su frecuencia en todos los documentos (IDF)

Cuanto más a menudo aparece una palabra en un solo documento, pero menos a menudo en todos los documentos, mayor es su puntaje TF-IDF

Las palabras que aparecen con más frecuencia en un documento pero raramente en otros documentos, son más importantes.

token aparece en muchos documentos de NLP, pero en documentos de python, no será común, entonces le da más peso para clasificar, de qué está hablando cierto documento.

Fórmulas de TF-IDF

1. TF-IDF se calcula multiplicando dos componentes: TF e IDF
2. TF se calcula como el número de veces que aparece una palabra en un documento, dividido por el total de palabras del documento
3. IDF se calcula como el logaritmo del total de documentos, dividido por el número de documentos que contienen la palabra.

$$\textcircled{1} \quad \text{TF-IDF} = \text{TF}(t,d) * \text{IDF}(t,D)$$

dónde:

$$\textcircled{2} \quad \text{TF}(t,d) = \frac{t}{D}$$

donde, t es el número de veces que un término aparece en un documento y D es el total de palabras del documento.

y

$$\text{IDF}(t,D) = \ln\left(\frac{D}{t}\right)$$

dónde, D es el total de documentos, y t es el número de documentos donde aparece la palabra

Por ejemplo, la palabra **token** aparece 5 veces en un documento de 1000 palabras. Si tenemos 5 documentos y **token** solo aparece en 1 de ellos, tenemos que:

$$\text{TF}(5, 1000) = 5/1000 = 0,005$$

$$\text{IDF}(5, 1) = \ln(5/1) = 1,609$$

$$- \text{TF-IDF} = 0,005 * 1,609 = 0,008 \uparrow$$

Aplicación del método TF-IDF

Ejemplo Práctico: Proyecto2.pdf Anexo 2

Para desarrollar este ejercicio necesitamos ingresar a [kaggle.com](https://www.kaggle.com) y en el dashboard buscar **5000 movies** para encontrar la base de datos

IMDB 5000 Movie Dataset

o simplemente buscar el enlace

[Kaggle.com/dataset/carolzhangdc/imdb-5000-movie-dataset](https://www.kaggle.com/dataset/carolzhangdc/imdb-5000-movie-dataset)

Descargamos y guardamos en la carpeta del proyecto **Guia_IA_2025** de la ubicación raíz de los proyectos anaconda jupyter.

Neural Word Embeddings

Incrustación de palabras neuroticas, es una técnica avanzada del **método de conteo** visto en la página 1.

Crea un vector por cada palabra, ya no por cantidad de palabras en un documento.

palabras = { 'avion' : [0.1, 0.9],
 'auto' : [0.15, 0.85] }

No son iguales,
pero tienen similitud

En resumen, tenemos 3 métodos de vectorización, vistos previamente:

BoW

TF-IDF

Embeddings

: Representamos el documento como una serie de vectores, uno para cada palabra y luego su combinación en una unidad del texto

Modelos para secuencias de Deep learning

- Modelos construidos para secuencias (CNN, RNN, Transformers)
- Relevancia del orden de las palabras en una sentencia
- Aplicaciones en traducciones de idiomas, respuestas a preguntas, chat bots, entre otras.

[CNN] Convolutional Neural Networks / [RNN] Recurrent Neural Networks

Word Embeddings (Incrustación de palabras)

- Introducción a Word2Vec y GloVe

Word2Vec usa redes neuronales con continuos Bag of Words CBOW y Skip-Gram

CBOW: Se toman palabras del contexto y se intenta predecir la clasificación de determinada palabra.

Texto predictivo del móvil

Skip-Gram: Toma una palabra e intenta predecir el contexto

Jesús → Religión / Nombres / Biblia

Glove (Global for word vector representation), algoritmo que aprende las representaciones vectoriales a partir de su matriz de coocurrencia

Librería Online → Me gustan libros de ciencia ficción

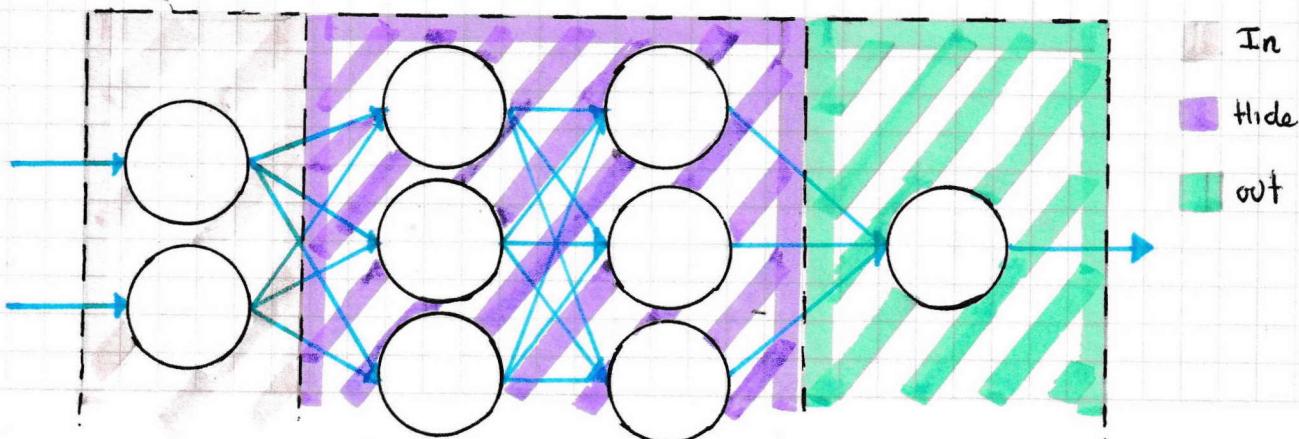
Dos palabras → Astronauta - Espacio



Similares porque tienden a aparecer en los gustos previos (Coocurrencia)

Ambas funcionan con redes neuronales (NN), que funcionan simulando el comportamiento del cerebro, uniéndo información para un todo a partir de 'neuronas'

Funciona ajustando los pesos de las neuronas en cada una de sus capas, en sus enlaces entre sí; mientras más se acerquen a la realidad se detiene el entrenamiento.



Uso práctico de Word Embeddings

Uso de analogías con las palabras, por ejemplo

España → Madrid

Alemania → Berlin

Si España está cerca de Alemania, entonces Madrid está cerca de Berlin.

Aplicación del Word Embeddings - Analogías

Ejemplo Práctico: Proyecto 3.pdf Anexo 3

Para desarrollar este ejercicio necesitamos ingresar a [kaggle.com](https://www.kaggle.com) y en el dashboard buscar [word vectors for spanish](#) para encontrar la base de datos

[Pre-trained Word Vectors for Spanish](#)

o simplemente buscar el enlace

[kaggle.com/datasets/rstatman/pretrained-word-vectors-for-spanish](https://www.kaggle.com/datasets/rstatman/pretrained-word-vectors-for-spanish)

Descargamos y guardamos en la carpeta del proyecto [Guia_1A_2025](#) de la ubicación raíz de los proyectos anaconda jupyter.

La base de datos tiene 1'000.000 de palabras en español y 300 dimensiones por cada una. La base de datos pesa aproximadamente 2 G.

Creación de un Embedding con Word2Vec

Ahora, vamos a crear tres entrenamientos, desde un texto sencillo, hasta un libro de gran tamaño, lo que permitirá tener nuestra base de datos de palabras embeddadas, para un uso más personalizado.

De esta manera, podemos crear [words embeddings](#) sin tener que usar bases de datos externas.

Antes de revisar las prácticas de entrenamiento, debemos identificar la disponibilidad de nuestro equipo en cuestión de rendimiento.

Ejemplo 7: CPU disponibles en mi PC

```
import os  
  
def numero_de_cpus():  
    return os.cpu_count()  
  
print(f'Mi equipo tiene {numero_de_cpus()} CPU's')
```

Salida: Mi equipo tiene 8 CPU's

Ahora, implementaremos los proyectos:

Ejemplos prácticos:

Proyecto4.pdf Anexo 4

Para desarrollar este ejercicio, usaremos un texto corto que tiene alrededor de 50.000 caracteres, llamado mundiales.txt.

Proyecto5.pdf Anexo 5

Para desarrollar este ejercicio, usaremos un texto más extenso llamado Cien años de soledad del escritor colombiano Gabriel García Márquez, que tiene alrededor de 139.318 palabras y 823.735 caracteres.

Proyecto6.pdf Anexo 6

Este ejercicio, permitirá combinar el nombre de forma masiva a diferentes archivos para ejecutarlos de manera rápida en un análisis de vectorización para NLP, evitando errores por los nombres que traen archivos a la hora de descargarlos o de generarllos.

Proyecto7.pdf Anexo 7

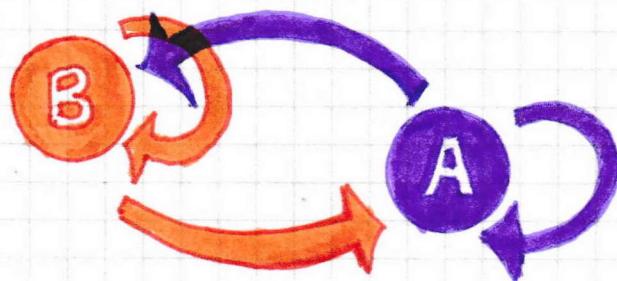
Para desarrollar este último ejercicio del módulo, usaremos 100 textos aleatorios en pdf, desde 20.000 caracteres hasta más de 1.500.000 caracteres, los cuales suman un total de 240.745.048 caracteres.

Módulo 2

Visión probabilística en AI

Introducción a los modelos de Markov en el NLP

Andrew Markov, diseñó una teoría que indicaba que los sistemas cambiaban con el tiempo; por ejemplo, el clima que con el pasar el tiempo va de un punto A a un punto B con cierta probabilidad y viceversa.



Apliabilidad universal de los modelos de Markov

Finanzas: Subida y bajada de precios a través de probabilidad

Aprendizaje por Refuerzo: Ajedrez

Modelos ocultos de Markov: Procesamiento de respuestas de Alexa, Bixby,...

Cadena de Markov Monte Carlo (MCMC): Análisis genético

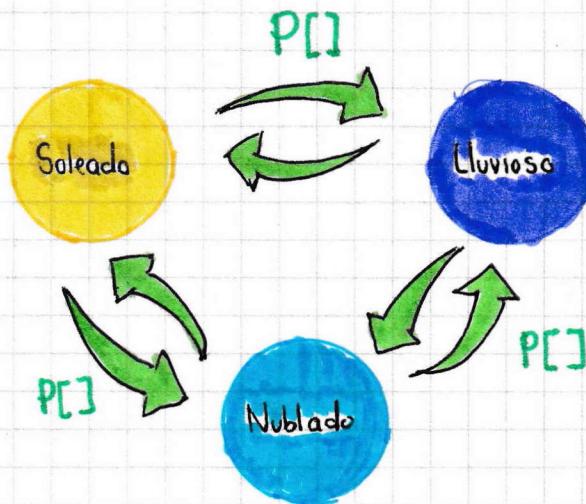
Propiedad fundamental de los modelos de Markov

Considerar solo el estado actual de un evento, para predecir el futuro, ignorando las variables pasadas de dicho evento.

Estructura y entrenamiento de un modelo de Markov

- Definición y componentes del modelo
- Matriz de transición de estados
- Entrenamiento del modelo

Por ejemplo, en el caso del clima, puede cambiar de un estado a otro transitando, o permanecer, de acuerdo a su probabilidad [0-1]



Aplicaciones en NLP

Clasificación de textos *spam*

Generación de texto original *poemas*

Diferencia entre aprendizaje *supervisado* y *No supervisado*.

Supervisado: Con base en una clasificación previa, buscamos clasificaciones entre nuestros textos nuevos, es decir, se conoce el resultado final.

No supervisado: No se conoce el resultado final, se entrena a partir de textos variados y a partir de modelos matemáticos se obtiene una clasificación futura por similaridad cercanía.

Procesos de Markov

Los procesos de Markov son esencialmente sistemas que pasan de un estado a otro con ciertas probabilidades.

Estas probabilidades, no dependen de cómo llegó al estado actual, sino solo del estado actual en sí.

Representación Matemática

Si tenemos tres estados: A, B y C. La probabilidad de pasar de A a B se representa de la siguiente manera:

$$P(B|A)$$

donde '|' se lee: ...dado a...
...dado que...

A representa que el día está soleado y B que el día está nublado,

$P(B|A)$ es la probabilidad de que se vuelva nublado dado que hoy el día está soleado.

Uso de modelos de Markov para Secuencias

Podemos usar un modelo de Markov para predecir la siguiente palabra basándose en la actual.

Por ejemplo, supongamos que después de la palabra cielo, la palabra azul aparece un 80% de las veces. Entonces,

$$P(\text{azul}' | \text{cielo}') = 0,8$$

Estados en modelos de Markov

- Definición de Estados y Símbolos Categóricos

En un modelo de Markov, por ejemplo, para prever el clima, los estados podrían ser Soleado, Lluvioso y Nublado.

Los estados los declaramos con s (de State)

- Notación y Representación

Usando la definición anterior, podemos tener la notación si para soleado, sj para día lluvioso y sk para día nublado. La probabilidad de que después de un día soleado venga un día lluvioso es

$$P(sj|si)$$

- Convención de numeración de los estados

Soleado = 1; Lluvioso = 2; Nublado = 3

- Distribución de probabilidad de los estados

(la suma de las probabilidades de transición desde Soleado a todos los otros estados, incluyendo quedarse en Soleado es 1.

Transición de Estados y Matrices de Transición

- Dependencia de los estados anteriores

Después de un día soleado, la probabilidad de que el siguiente sea lluvioso no depende de si los días anteriores fueron soleados o lluviosos.

- Distribuciones Condicionales y Valores de Probabilidad

$$P(\text{Lluvioso} | \text{Soleado}) = 0.4 \quad P(\text{Nublado} | \text{Soleado}) = 0.6$$

- Matriz de Transición de Estado

	Soleado	Lluvioso	Nublado
Soleado	0.5	0.3	0.2
Lluvioso	0.3	0.4	0.3
Nublado	0.2	0.5	0.3

- Homogeneidad en el tiempo

- Distribución del estado inicial (P_i)

Al inicio de una secuencia, no hay un estado anterior del que depender. Por eso, necesitamos una distribución de probabilidad inicial para determinar el primer estado.

$$P_i = [0.6 (\text{Soleado}), 0.3 (\text{Nublado}), 0.1 (\text{Lluvioso})]$$

- Representación en Python

transiciones = {

$$\begin{aligned} 'A': \{ 'A': 0.5, 'B': 0.3, 'C': 0.2 \}, \\ 'B': \{ 'A': 0.3, 'B': 0.4, 'C': 0.3 \}, \\ 'C': \{ 'A': 0.2, 'B': 0.5, 'C': 0.3 \} \end{aligned}$$

}

Implementación Computacional y Entrenamiento

Podemos usar bibliotecas de Python como NumPy para ayudar a representar y manipular matrices y vectores.

Con la matriz de transición y Π , podemos calcular la probabilidad de una secuencia específica usando un simple bucle.

Usando datos históricos, podemos "entrenar" nuestro modelo ajustando las probabilidades hasta que representen mejor nuestros datos.

Si observamos que después de 100 días soleados, 70 resultaron ser nublados al día siguiente, la probabilidad de transición sería 0.7.

Procesos de Markov - Navegación de Probabilidades

Estimaciones de Máxima Verosimilitud

$$P(\text{word}_2 | \text{word}_1) = \frac{\text{count}(\text{word}_1, \text{word}_2)}{\text{count}(\text{word}_1)}$$

Si la palabra luna aparece 100 veces en el dataset y la secuencia luna llena aparece 20 veces, entonces

$$P(\text{llena} | \text{luna}) = \frac{20}{100} = 0.2$$

Problemas de valores CERO

Si intentamos calcular la probabilidad de una secuencia y alguno de los valores resulta ser **cero** (porque ciertos pares de palabras no aparecieron en el conjunto de entrenamiento), toda la expresión se vuelve **cero**.

Si la secuencia luna llena nunca aparece, entonces,

$$P(\text{llena} | \text{luna}) = 0$$

Esto no es deseable, ya que una frase no debería ser considerada imposible solo porque contiene un par de palabras que no se encontraron en el conjunto de entrenamiento.

Suavizado de probabilidad

El suavizado es la solución en los casos donde la probabilidad tiene problemas de valores cero.

Suavizado +1 (Laplace)

Si en nuestro dataset hay 1000 palabras únicas y la secuencia luna llena nunca aparece, entonces

$$P_{+1}(\text{word}_2 | \text{word}_1) = \frac{\text{count}(\text{word}_1, \text{word}_2) + 1}{\text{count}(\text{word}_1) + V}$$

donde V es el número total de estados (palabras del entrenamiento del dataset)

$$P_{+1}(\text{llena} | \text{luna}) = \frac{0 + 1}{\text{count}(\text{luna}) + 1000}$$

Suavizado Epsilon

Si tenemos el mismo ejemplo del Suavizado+1 o de Laplace, usamos un ϵ muy pequeño, en este caso $\epsilon=0.1$, tenemos que:

$$P_\epsilon(\text{word}_2 | \text{word}_1) = \frac{\text{count}(\text{word}_1, \text{word}_2) + \epsilon}{\text{count}(\text{word}_1) + \epsilon \times V}$$

donde ϵ es un valor positivo muy menor a 1 ($\epsilon \ll 1$)

$$P_\epsilon(\text{llena} | \text{luna}) = \frac{0 + 0.1}{\text{count}(\text{luna}) + 0.1 \times 1000}$$

Probabilidad de una secuencia

$$P(\text{sequence}) = P(\text{word}_1) \times P(\text{word}_2 | \text{word}_1) \times \dots \times P(\text{word}_n | \text{word}_{n-1})$$

donde, $P(\text{word}_1)$ es la probabilidad de que aparezca la primera palabra.
 $P(\text{word}_2 | \text{word}_1)$ es la probabilidad de que la segunda palabra aparezca después de la primera y así sucesivamente.

Por ejemplo, vamos a utilizar la secuencia el gato duerme:

1. La probabilidad de que la palabra el aparezca al inicio $P(\text{el})$.
2. La probabilidad de que la palabra gato siga a el es $P(\text{gato}|\text{el})$.
3. La probabilidad de que la palabra duerme siga a gato es $P(\text{duerme}|\text{gato})$.

Si cada una de estas probabilidades es 0,01 (es decir, 1% de probabilidad):

$$P(\text{sequence}) = P(\text{el}) \times P(\text{gato}|\text{el}) \times P(\text{duerme}|\text{gato})$$

$$P(\text{sequence}) = 0.01 \times 0.01 \times 0.01$$

$$P(\text{sequence}) = 0.000001$$

Esto significa que, basado en nuestro modelo hipotético, la probabilidad de que la frase el gato duerme aparezca en ese orden específico es de 0.000001, es decir, el 0.0001%.

Al intentar calcular la probabilidad conjunta de una secuencia (especialmente en lenguaje), se termina multiplicando muchos números pequeños, lo que puede llevar a errores numéricos o desbordamiento por debajo (underflow).

Esta problemática ocurre porque la multiplicación de valores pequeños, resulta en números aún más pequeños. Una PC podría redondear estos valores extremadamente pequeños a cero, lo que genera errores en los cálculos.

Espacio logarítmico

Es la solución a los problemas de underflow y aproximaciones a cero.

En lugar de trabajar directamente con las probabilidades, se propone trabajar con logaritmos de probabilidades.

La ventaja es que la suma de logaritmos es más numéricamente estable que la multiplicación de probabilidades. Además, computacionalmente, sumar es más eficiente que multiplicar.

Es importante tener en cuenta que el logaritmo es una función monótonamente creciente, lo que significa que si un número es mayor que otro, su logaritmo también será mayor; así, trabajar en el espacio logarítmico no altera las relaciones de orden entre las probabilidades.

De esta manera, tenemos que

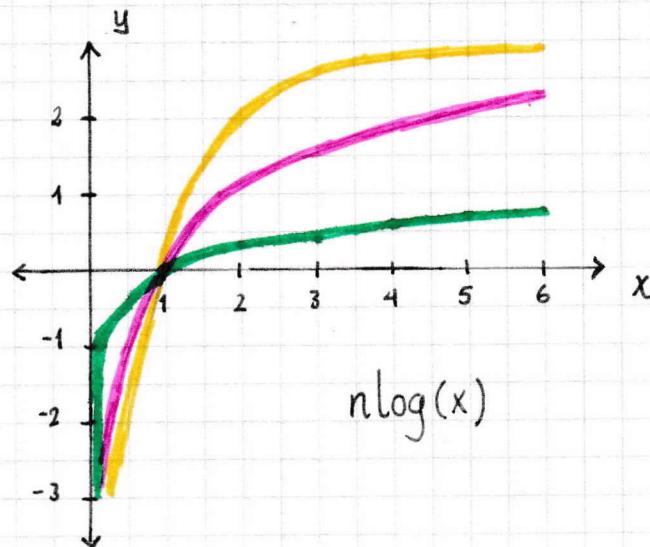
$$\log(P(\text{sequence})) = \log(P(\text{word}_1)) + \log(P(\text{word}_2|\text{word}_1)) + \dots + \log(P(\text{word}_n|\text{word}_{n-1}))$$

Por ejemplo, vamos a utilizar la secuencia del ejemplo anterior:

$$\log(P(\text{sequence})) = \log(0.01) + \log(0.01) + \log(0.01)$$

Si utilizamos logaritmo en base 10, entonces $\log(0.01) = -2$, entonces

$$\log(P(\text{el gato duerme})) = -2 - 2 - 2 = -6$$

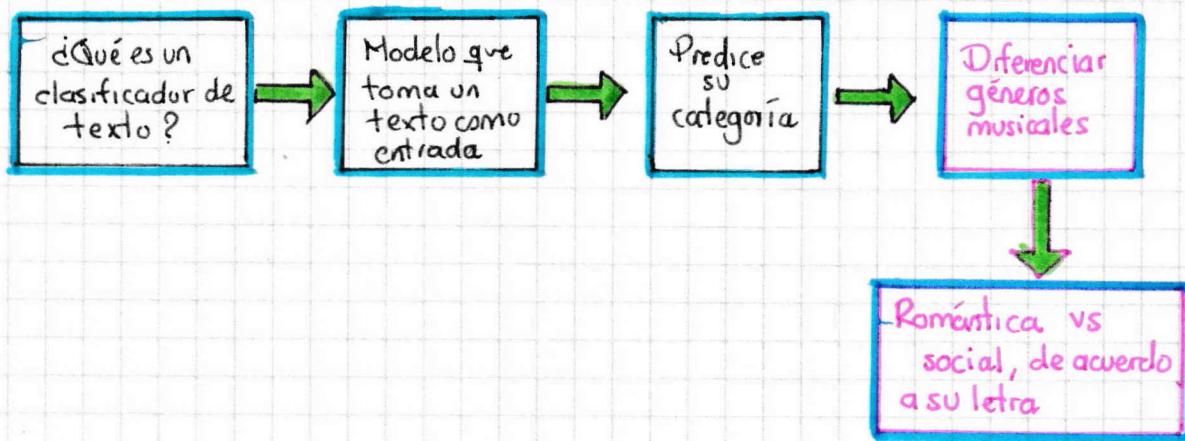


Construcción de un generador de texto

Aplicaremos los modelos de Markov para construir un clasificador de texto, pasando de la teoría a una aplicación real y funcional.

Estos modelos pueden aprender patrones de texto, para que luego de su entrenamiento, pueda clasificar nuevos textos.

Observemos la siguiente estructura de ejemplo; en este caso de la ruta a ejecutar en una situación real.



Otras aplicaciones de un clasificador de texto

- Predecir si un correo es o no, **spam**
- Determinar si una crítica de una película es positiva o negativa.

Aplicación del modelo de Markov y clasificación de textos

El objetivo de este ejercicio, no es crear el modelo más preciso, ya que estos analizan miles de millones de datos.

Clasificación de textos es **aprendizaje supervisado**.

Modelos de Markov es **aprendizaje no supervisado**

Regla de Bayes permite combinar ambos aprendizajes.

Regla de Bayes

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

por ejemplo,

$$P(\text{spam}|\text{premio}) = \frac{P(\text{premio}|\text{spam}) * P(\text{spam})}{P(\text{premio})}$$

Aplicación del clasificador de textos

Ejemplo Práctico: Proyecto8.pdf Anexo 8

Para desarrollar este ejercicio, utilizaremos dos archivos pdf, con diversos poemas de **Jorge Luis Borges** y **Mario Benedetti**.

A partir de estos textos, crearemos un **Clasificador de textos** que permitirá, a través de un texto nuevo ingresado por el usuario final, predecir si su forma de escritura pertenece a la clase 0 (**Jorge Luis Borges**) o a la clase 1 (**Mario Benedetti**).

Generador de texto

Usaremos modelos de **Markov** para generar textos.

Corresponde a la línea de **Aprendizaje No supervisado** ya que no necesitamos **etiquetas**.

Ja que se busca en este caso es predecir si una palabra se acomoda coherenteamente a una frase, ya que no tenemos etiquetas o valores conocidos para entrenar el modelo, solo el estado actual según la cadena de **Markov**

Ampliación de los modelos de Markov

Problemas en los modelos de Markov



Esta palabra, solo depende de la anterior, más no del contexto de la frase.

Para dar solución a este problema, necesitamos indagar no solo el estado anterior, sino, sus antecesores, por lo menos uno de ellos.

Modelo de Markov de segundo orden

Es una nueva forma de almacenar y representar las probabilidades de transición

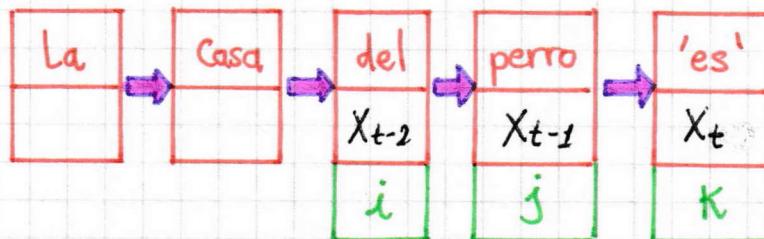
La matriz tridimensional A se representa de la siguiente manera:

$$a_{ijk} = P(X_t = k \mid X_{t-1} = j \mid X_{t-2} = i)$$

dónde,

- X_t , representa el estado en el tiempo.
- a_{ijk} , es la probabilidad de que, dado que el sistema estaba en el estado i en el tiempo $t-2$ y en el estado j en el tiempo $t-1$, estará en el estado k en el tiempo t .

Para el diagrama anterior,



Implicaciones

A medida que aumenta el número de estados anteriores que consideremos, el tamaño de la matriz A (**tensor**) crece exponencialmente.

Este problema que vamos a mencionar, se debe al crecimiento de la matriz, el cual conlleva a una indeseada eficiencia computacional, y tener que usar recursos significativos para su implementación y los cálculos de ésta.

Creación de un generador de texto con python

Ejemplo Práctico: Proyecto9.pdf Anexo 9

Para desarrollar este ejercicio, utilizaremos el archivo pdf de poemas de **Federico Beneditti**, lo convertimos en archivo de texto, para depurar su trabajo.

A partir de este texto, crearemos un **Generador de textos**, el cual, utilizando los modelos de **Marcov** de primer y segundo orden, entrenará un modelo que sea capaz de generar un texto al estilo de este gran poeta.

Text Spinning

El text spinning o Article Spinning, es una técnica de tomar un artículo ya escrito y modificarlo (reemplazando palabras, frases, reorganizando estructuras) para que aparezca un nuevo contenido, pero manteniendo el mensaje original.

Importancia de los motores de búsqueda

los factores que influyen en el ranking, como **google**.

El desafío de crear contenido original y abundante para que los motores lo tengan en cuenta

Requiere un tiempo y esfuerzo para escribir estos contenidos

Formas de hacer Spinning de contenido

- Anteriormente se hacía de forma manual, haciendo uso del recurso humano para editar los contenidos haciendo uso del lenguaje.
- Actualmente, se hace de forma automática con NPL.

Evolución de las técnicas de generación de contenido

- Uso de software de sugerencia de palabras
 - Limitadas técnicas previas al Machine Learning.
 - Modelos de Marcov como técnica básica.

Avances técnicos en NPL

Mejoras en los procesos de **Markov**, implementando modelos de segundo y tercer orden.

Técnicas avanzadas como **RNN's** y **Transformers**.

Los modelos de **Markov** tienen una falencia con la memoria a largo plazo, ya que en segundo grado, solo toma las dos palabras anteriores, lo que significa que va perdiendo el contexto.

N-GRAM

Los **N-Grams** y los modelos de **Markov** están estrechamente relacionados en el modelado de lenguaje. Como sabemos, los modelos de **Markov** son una clase de modelos probabilísticos en los que el estado futuro depende solo del estado actual (o de un número limitado de estados anteriores), lo que encaja perfectamente con la idea de **N-Grams** en **NPL**.

Un modelo de **Markov** de orden **N-1** asume que la probabilidad de la siguiente palabra en una secuencia depende solo de las **N-1** palabras anteriores. Esto es exactamente lo que hace un modelo basado en **N-Grams**.

Por ejemplo, en un modelo de **bigrams (N-2)**, la probabilidad de la palabra **Wn** depende solo de la palabra **Wn-1**:

$$P(W_n | W_1, W_2, \dots, W_{n-1}) \approx P(W_n | W_{n-1})$$

Para el uso de **Spelling** debemos implementar un modelo de distribución para una palabra dada, basada en la palabra anterior y la palabra siguiente

$$P(W_n | W_{n-1}, W_{n+1})$$



Propuesta de modelo para Spinning

$$P(W_t | W_{t-1}, W_{t+1}) = \frac{\text{count}(W_{t-1} \rightarrow W_t \rightarrow W_{t+1})}{\text{count}(W_{t-1} \rightarrow \text{ANY} \rightarrow W_{t+1})}$$

por ejemplo:

<u>t-1</u>	<u>t</u>	<u>t+1</u>
el	sol	brilla
el	foco	brilla
el	sol	brilla
el	sol	brilla

$$P(\text{sol}(t) | \text{el}(t-1), \text{brilla}(t+1)) = \frac{3}{4}$$

≈ 0,75

Creación de Spinning de Texto con python

Ejemplo práctico: [Proyecto 10.pdf](#) Anexo 10

Para desarrollar este ejercicio, necesitamos ingresar a [Kaggle.com](#) y en el dashboard buscar [Spanish news](#) para encontrar la siguiente base de datos:

Spanish news (La Razón - Público)

O simplemente, buscar el enlace

[Kaggle.com/datasets/josemamuiz/noticias-larazonpublico?](https://www.kaggle.com/datasets/josemamuiz/noticias-larazonpublico)

Descargamos y guardamos en la ruta [Guia_IA_2025/Modulo2/data](#) de la ubicación raíz de los proyectos anaconda-jupyter.

Módulo 3

Métodos de Machine Learning para NLP

Introducción al aprendizaje automático

El aprendizaje automático (**Machine learning ML**) es una rama de la **Inteligencia Artificial (AI)** que se enfoca en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender patrones a partir de datos y tomar decisiones sin estar explícitamente programadas.

Como definimos en capítulos anteriores, los **ML** se clasifican en tres tipos, **aprendizaje supervisado (SL)**, **aprendizaje no supervisado (UL)** y **aprendizaje por refuerzo (RL)**.

Ejemplos de modelos de aprendizaje

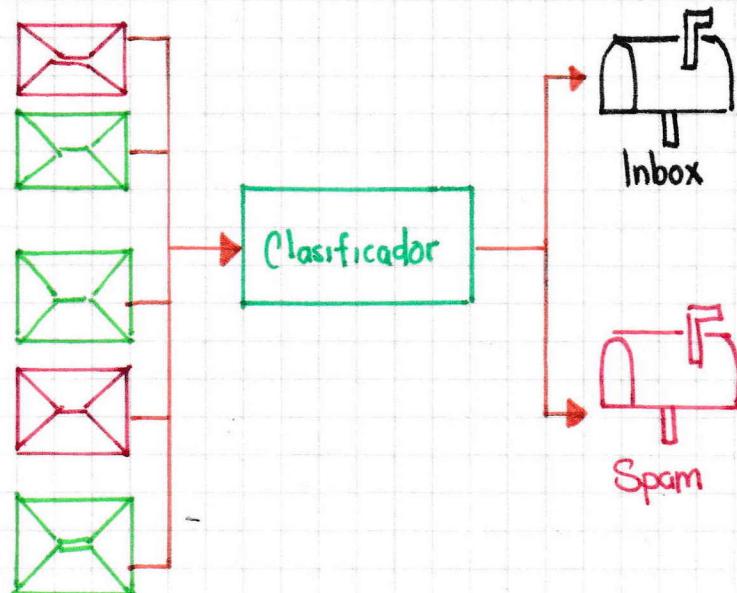
- Detección de spam: **Naive Bayes**.
- Análisis de sentimientos: **Regresión logística**.
- Indexación semántica latente (SEO): **PCA** y **SVD**.
- Modelado de temas: **Asignación latente directa (LDA)**.

Detección de SPAM

Proceso mediante el cual se identifican y se filtran correos electrónicos o mensajes no deseados

Por ejemplo:

Ventas
Malware
Robo de credenciales
Estafas



Importancia de la detección de Spam

- Experiencia de usuario
- Seguridad
- Eficiencia

Automatización mediante ML para filtrar spam

- Adoptabilidad
- Precisión
- Automatización

Descripción del proceso

Función objetivo: detectar_spam

Input: la entrada para esta función es el texto del correo electrónico, que podría ser cualquier mensaje, ya sea un correo electrónico, un SMS, o cualquier otro tipo de mensaje.

Output: Una vez que el documento pasa por la función, esta devolverá un valor binario, es decir, uno de dos posibles resultados:

retorna 1 si el correo es spam.

retorna 0 si el correo no es spam

Regla de Naïve Bayes

La regla de Bayes es un proceso de probabilidad condicional que describe cómo actualizar las probabilidades de hipótesis cuando se dispone de nueva evidencia.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

donde,

$P(A|B)$ → es la probabilidad posterior de A dado B.

$P(B|A)$ → es la probabilidad de B dado A.

$P(A)$ → es la probabilidad previa de A.

$P(B)$ → es la probabilidad total de B.

Por ejemplo:

Supongamos que hay una enfermedad rara en una población, y sólo el 1% de la población la tiene. Hay una prueba para detectar dicha enfermedad, la cual tiene las siguientes características.

- Si tienes la enfermedad, la prueba tiene un 99% de probabilidad de darte un resultado positivo.
- Si no tienes la enfermedad, la prueba tiene un 95% de probabilidad de darte un resultado negativo, pero un 5% de probabilidad de darte un resultado falso-positivo.
- Ahora, supongamos que te haces la prueba y el resultado es positivo. ¿Cuál es la probabilidad de que realmente tengas la enfermedad?

$$P(\text{enfermedad} | \text{positivo}) = \frac{P(\text{positivo} | \text{enfermedad}) \times P(\text{enfermedad})}{P(\text{positivo})}$$

dónde:

$P(\text{enfermedad} | \text{positivo})$ es lo que queremos encontrar

$P(\text{positivo} | \text{enfermedad})$ es 0,99

$P(\text{enfermedad})$ es 0,01

$P(\text{positivo})$ es (usando la ley total de probabilidad)

$$P(\text{positivo}) = P(\text{positivo} | \text{enfermedad}) \times P(\text{enfermedad}) + \\ P(\text{positivo} | \text{No enfermedad}) \times P(\text{No enfermedad})$$

$$P(\text{positivo}) = 0,99 \times 0,01 + 0,05 \times 0,99 = 0,0594$$

reemplazamos en bayes:

$$P(\text{enfermedad} | \text{positivo}) = \frac{0,99 \times 0,01}{0,0594} \approx 0,16$$

Aplicación de la regla de Bayes en ML.

X variables de entrada
Y variable objetivo

Por ejemplo: Clasificación de correos electrónicos (spam o No spam)

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)}$$

donde,

$P(Y=\text{spam} | X)$ probabilidad de que un correo sea spam dado su contenido.

$P(X | Y=\text{spam})$ probabilidad de observar cierto contenido en un correo electrónico sabiendo que es spam.

$P(Y=\text{spam})$ probabilidad general de que cualquier correo sea spam.

$P(X)$ probabilidad de observar un correo con cualquier contenido.

Aplicando Naive Bayes en ML

En este caso, suponemos adicionalmente que hay más de un evento independiente, por ejemplo, si el correo spam trae las palabras ganar y gratis entonces, implementamos la regla de bayes y aplicamos el algoritmo naive

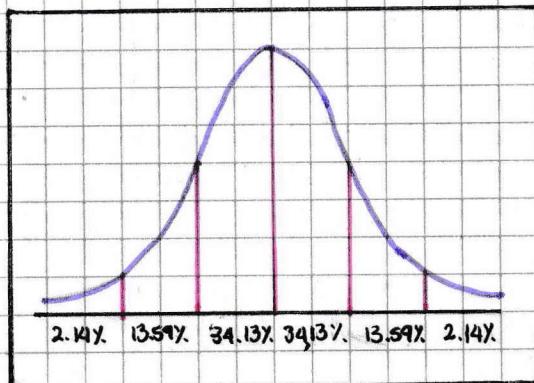
$$P(\text{ganar, gratis} | \text{spam}) = P(\text{ganar} | \text{spam}) \times P(\text{gratis} | \text{spam})$$

Elección del modelo

El modelo se elige de acuerdo a la distribución de los datos, como veremos a continuación.

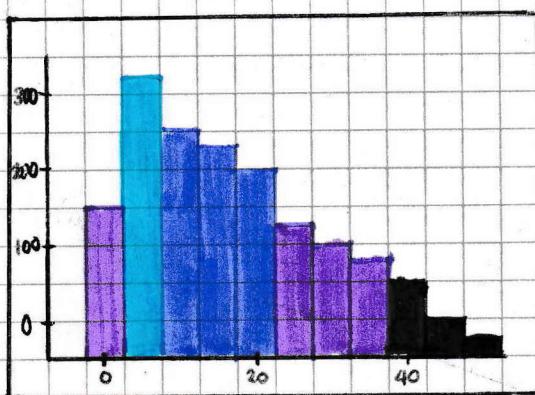
Gaussiano:

Si los datos tienen distribución normal (gaussiana), el naïve Bayes Gaussiano es la mejor opción. Por ejemplo, medidas de altura, masa, peso, etc.



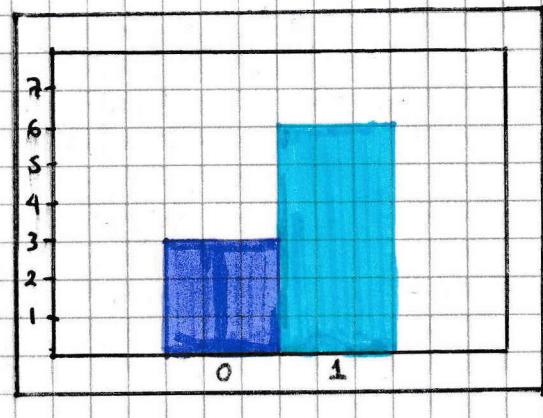
Multinomial:

Si los datos representan conteos, como la frecuencia de palabras en documentos, naïve Bayes Multinomial es la mejor opción. NPL



Bernoulli:

Si los datos son binarios o representan la presencia o ausencia de características, el naïve Bayes Bernoulli es la mejor opción.



Aplicación de detector de Spam

Ejemplo práctico: Proyecto11.pdf Anexo 11

Para desarrollar este ejercicio, necesitamos ingresar a [Kaggle.com](#) y buscar los siguientes datasets:

[Spam or Ham in Spanish 15.000](#)

[Email spam Detection 98% \(EDA + ML\)](#)

O simplemente, buscar los enlaces:

[kaggle.com/datasets/wistestatter2025/spam-or-ham-in-spanish](https://www.kaggle.com/datasets/wistestatter2025/spam-or-ham-in-spanish)

[kaggle.com/code/sachinpatil1280/email-spam-detection-98-eda-ml](https://www.kaggle.com/code/sachinpatil1280/email-spam-detection-98-eda-ml)

Descargamos y guardamos en la ruta [Guia_1A_2025/Modulo3/data](#) de la ubicación raíz de los proyectos [anaconda-jupyter](#).

Análisis de Sentimientos

Como introducción, tenemos:



- Presentación del tema en el contexto de NLP
- Exploración del análisis de sentimientos
- Interpretación, y categorización de las opiniones y emociones humanas expresadas en forma de texto.
- Técnicas, algoritmos y herramientas utilizadas comúnmente para el análisis de sentimientos.

Por ejemplo, tenemos las siguientes frases:

1. Wow, esa fue una película realmente genial
2. No puedo creer que haya desperdiciado dos horas de mi vida en esta película

- Concepto general de sentimiento; medida de positividad o negatividad.
- Categorización típica = **positivo, negativo y neutral**.

Clasificación vs. Regresión en el análisis de sentimientos

• El análisis de sentimiento es una tarea de clasificación

• En la regresión se consideran:

- Relación con sistemas de recomendación

- Diferencia en el tratamiento de sentimientos y categorías.

Clasificación \Rightarrow Etiqueta (Bueno, Regular, Malo)
 Regresión \Rightarrow Puntaje (mayor rango)

Descripción de la tarea del Análisis de Sentimiento

Definición de la tarea.

Contextualización con otros métodos de aprendizaje supervisado

Comparación con detección de spam (Tarea de clasificación)

Aplicabilidad y beneficios del Análisis de Sentimiento

- Beneficios en términos financieros
- Gestión de reputación
- Monitoreo de redes sociales
- Análisis programático de comentarios
- Análisis competitivo
- Análisis en soporte al cliente
- Influencia de artículos y noticias en las acciones de las personas.

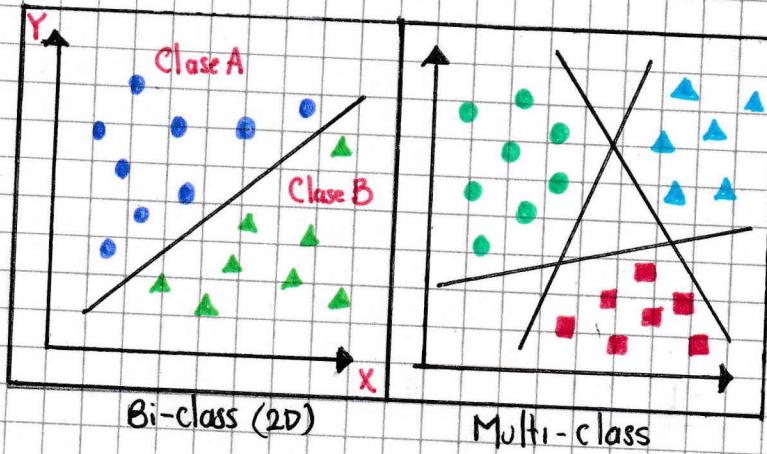
Regresión logística

Hasta el momento, hemos visto modelos basados en vectores y basados en probabilidades.

La regresión logística, pertenece a los modelos basados en vectores.

Perspectiva vectorial sobre la tarea de clasificación

- Trajar vectores en dimensiones (Simplificación 2D)
- Encontrar una recta o curva que pueda separar puntos de diferentes colores.
- Los colores representan diferentes clases.



Representación lineal

La regresión logística es un modelo lineal:

$$X_1 = m X_2 + i \quad \Rightarrow \quad y = mx + b$$

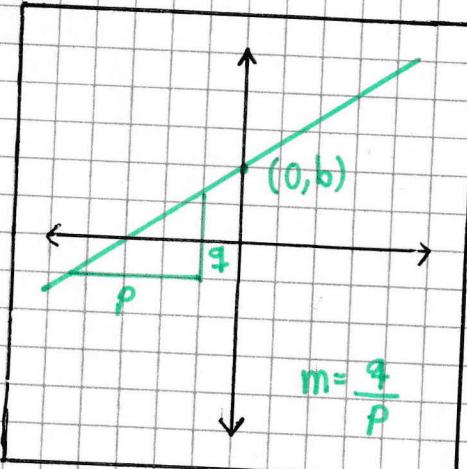
$$X_1 - m X_2 - i = 0$$

$$W_1 X_1 + W_2 X_2 + b = 0$$

$$W_1 = 1$$

$$W_2 = -m$$

$$b = -i$$



$$X_1 = Y$$

$$X_2 = X$$

$$i = b$$

donde,

W \Rightarrow pesos
 b \Rightarrow sesgos

Por ejemplo, si tenemos

$$x_1 + x_2 - 1 = 0$$

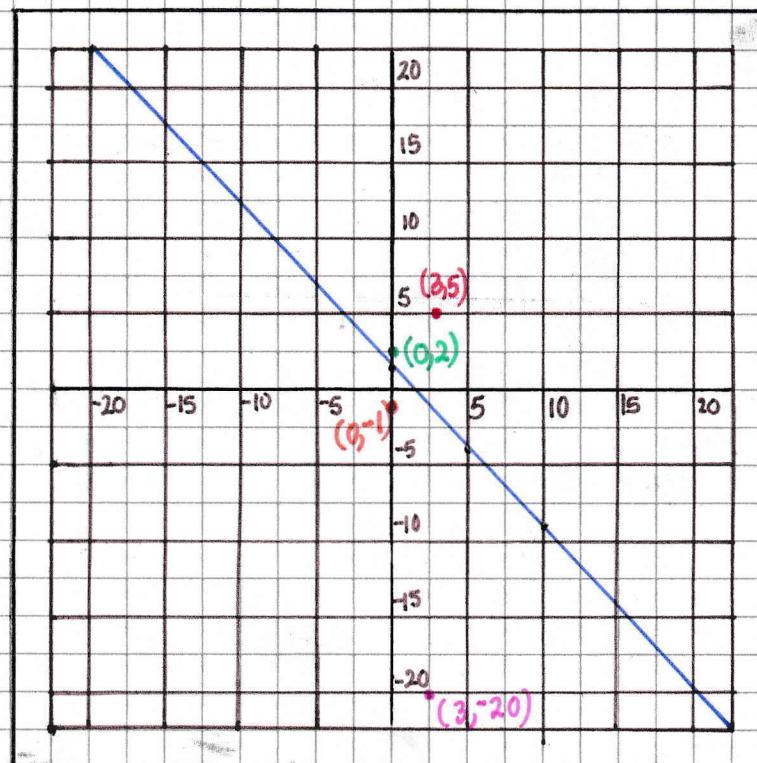
$$w_1 = 1$$

$$w_2 = 1$$

$$b = -1$$

gráficamente, tendríamos

$$x_1 + x_2 - 1 = 0$$



Observemos los siguientes puntos:

$$(0, 2)$$

$$(0, -1)$$

$$\{3, 5\}$$

$$(3, -20)$$

los puntos $(0, 2)$ y $(3, 5)$ pertenecen a una clase, y los puntos $(0, -1)$ y $(3, -20)$ pertenecen a otra clase.

Por lo tanto, cualquier punto que cumpla con la igualdad estará en la línea. Cuando esta expresión es menor que cero, entonces el punto que conectamos cae a la izquierda de la línea. Cuando esta expresión es mayor que cero, entonces el punto que conectamos cae a la derecha de la línea.

Activación (Pertenecer a una clase u otra)

Si $a(x) > 0$ pertenece a la clase A $[(0, 2), (3, 5)]$

Si $a(x) < 0$ pertenece a la clase B $[(0, -1), (3, -20)]$

Si $a(x) = 0$ está en la linea. -

Funciones de Activación

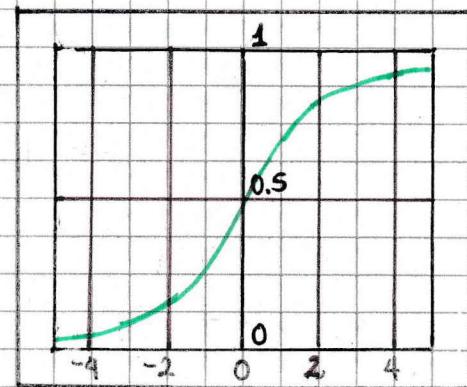
Función Sigmóide

Es una función umbral que modifica el valor resultante o impone un límite que se debe superar para poder decidir si pertenece a una clase u otra.

Acota los valores entre 0 y 1

Cuando $x=0$, $\sigma(x) = 0.5$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Regresión logística Multiclasas MLR

Es una continuación de la regresión logística en la cual intervienen múltiples clases.

Utiliza vectores **Quade** (w_1, w_2, \dots, w_k) quienes representan a través de coeficientes las relaciones entre las variables, donde k es el número de las clases.

Tiene múltiples sesgos (b_1, b_2, \dots, b_k) y k -cálculos de activaciones (a_1, a_2, \dots, a_k)

Función Softmax

Es una variación de la función **sigmoide**, donde el umbral está dado por la sumatoria de las diferentes clases, entregando como respuesta un vector de probabilidades.

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}} \rightarrow$$

0.02
0.90
0.05
0.01
0.02

donde,

\mathbf{z} es el vector de activación

K es el número total de las clases

x_i es el valor de entrada del número i en un conjunto de valores x_1, x_2, \dots, x_n

j es el iterador para recorrer cada una de las clases en la sumatoria

Salidas esperadas de la regresión logística multiclas.

Su salida está en probabilidades y se ajusta al total de eventos de clase del espacio muestral, permitiendo así predecir la ocurrencia de cada clase.

Por ejemplo,

Supongamos un modelo MLR que clasifica las opiniones de los usuarios en tres categorías: 'Positivo', 'Neutro' y 'negativo'.

Un usuario escribe: 'Me encanta este producto'. La salida del modelo podría ser una matriz de probabilidades como esta:

Positivo :	0.85
Neutral :	0.10
Negativo :	0.05

Esto indica que el modelo tiene una alta confianza (85%) en que el comentario es positivo.

Aplicación de Análisis de Sentimiento

Ejemplo práctico: Proyecto 12.pdf Anexo 12

Para desarrollar este ejercicio, necesitamos ingresar a Kaggle.com y buscar el siguiente dataset:

sentiment analysis dataset in spanish

o simplemente, buscar el enlace:

[Kaggle.com/datasets/wistestalter2025/sentiment-analysis-dataset](https://www.kaggle.com/wistestalter2025/sentiment-analysis-dataset)

Descargamos y guardamos en la ruta Guia_IA_2025/Modulo3/data de la ubicación raíz de los proyectos anaconda-jupyter.

Ejemplo Práctico: Proyecto13.pdf Anexo 13

Para desarrollar este ejercicio, necesitamos ingresar a Kaggle.com y buscar el siguiente dataset:

social media sentiment analysis dataset

O simplemente, buscar el enlace:

[Kaggle.com/datasets/lvistestatter2025/social-media-analysis-dataset](https://www.kaggle.com/datasets/lvistestatter2025/social-media-analysis-dataset)

Descargamos y guardemos en la ruta **Guia_1A_2025/Modulo3/data** de la ubicación **ra13** de los proyectos anaconda-jupyter.

Resumen de texto

Consiste en tomar un documento largo y mostrar su contenido en un formato más corto.

Permite a los lectores entender rápidamente el propósito y los temas principales de un texto sin tener que leerlo en su totalidad.

La sumarización permite avances en el análisis de artículos científicos, resúmenes ejecutivos y documentos profesionales generando una lectura eficiente.

Sumarización con Inteligencia Artificial

Se requiere una profunda comprensión para hacer sumarización, lo que conlleva un gran rango de inteligencia artificial.

Aplicación en motores de búsqueda

Complen un rol importante por la manera de realizar las búsquedas, y la comprensión que tienen para generar una anatomía en los resultados de las búsquedas.

Título	(Método fácil de búsqueda)
Enlace	(Dirección de la fuente)
Resumen	(snipet o fragmento)

Actualmente, entregan los contenidos de acuerdo a la forma en que buscamos.

Por ejemplo, busquemos en google: importancia de levantarse temprano. y observemos que el navegador despliega la siguiente estructura:

★ Visión general creada por IA

Resumen de búsqueda

→ resumen creado por IA

Algunas páginas rankeadas

→ páginas sugeridas

Más preguntas :

c?

c?



} Clasificaciones sobre su búsqueda

Categorización de la sumarización de texto

Resúmenes Extractivos

Extraen partes específicas del texto

Resúmenes Abstractivos

Generan nuevas secuencias de texto asociado al texto original.

Proceso de sumarización

- Usaremos métodos basados en vectores
- Usaremos Text Rank:

Probabilidad
Algebra lineal
Cadenas de Markov

Resumen de texto con vectores

Vamos a revisar, inicialmente, la teoría TF-IDF del Módulo 1 en la página 12.

Ventajas

- Facilidad de implementación
- Relevancia de palabras claves
- Adaptabilidad
- Reducción de ruido

Pasos básicos en la técnica TF-IDF para sumarización

Vamos a usar un ejemplo similar al de la página 5 del Módulo 1, donde tenemos el siguiente texto dividido en oraciones

1 Documentos

Me encanta el café. El café me mantiene despierto

Me encanta la música, me alegra

Odio el calor

- Tokenizamos con TF-IDF
- Verificamos stopwords
- Creamos la matriz TF-IDF

Palabras	Documento 1	Documento 2	Documento 3
me	-	-	Nulo
encanta	Media	Media	Nulo
el	-	Nulo	-
café	Alto	Nulo	Nulo
mantiene	Media	Nulo	Nulo
despierto	Media	Nulo	Nulo
la	Nulo	-	-
música	Nulo	Media	Nulo
alegra	Nulo	Media	Nulo
odio	Nulo	Nulo	Media
calor	Nulo	Nulo	Media

Nulo: Se puede interpretar como relevancia baja

- : Indica una posible stopword

Media: Indica un peso de 1

Alta: Indica un peso de 2

En donde se asigna un peso a las relevancias de las palabras.

- La puntuación de cada documento es:

$$\text{Documento 1: } \text{Media} + \text{Media} + \text{Media} + \text{Alta} = 5$$

$$\text{Documento 2: } \text{Media} + \text{Media} + \text{Media} = 3$$

$$\text{Documento 3: } \text{Media} + \text{Media} = 2$$

Selección de Sentencias para hacer el resumen

- i - Top N oraciones
 - ii - Top N palabras o caracteres
 - iii - Porcentajes
 - iv - Umbral de puntuación
 - v - Umbral de factores multiplicativos
- Documento 1 = 5, por ejemplo
 Café = 2, por ejemplo
 10% de las palabras top
 > 2 → Documento 1 y 2
 1.5 * Umbral, por ejemplo.

Consideraciones de cada método

- i - Puede excluir sentencias importantes si N es demasiado bajo, o incluir sentencias irrelevantes si N es demasiado alto.
- ii - Podría cortar sentencias a la mitad, llevando a resúmenes que no son coherentes.
- iii - La calidad del resumen puede variar según la longitud del documento original.
- iv - Si se establece un umbral demasiado alto, muchas sentencias podrían ser excluidas. Si es demasiado bajo, el resumen sería demasiado largo.
- v - Elegir un factor adecuado puede ser subjetivo y podría requerir ajustes, según el documento.

Aplicación de resumen de texto con vectores

Ejemplo práctico: Proyecto14.pdf Anexo 14

Para desarrollar este ejercicio, utilizaremos la base de datos descargada en el Proyecto 1.ipynb del Módulo 1.

Se aconseja copiar la base de datos a la ruta

Módulo 3/data

con el fin de organizar los componentes del proyecto.

Resumen de textos con Text Rank

Textos como los del ejemplo anterior se pueden mejorar en sus resúmenes con esta técnica.

Text Rank es un avanzado método de resumen automático que se inspira en el algoritmo **PageRank** de **Google**. Aunque existen bibliotecas que facilitan el uso de **Text Rank**, comprender como funciona detrás de escena es crucial para su aplicación efectiva.

Comparación de Text Rank con TF-IDF

Recordemos, en resumen, el proceso que hace el método **TF-IDF**

1. Se divide el documento en oraciones.
2. Se computa una matriz **TF-IDF** para esas oraciones.
3. Cada oración se convierte en un vector de valores **TF-IDF** para cada palabra.
4. Se puntuá cada oración tomando el **promedio** de los componentes **TF-IDF**.
5. El resumen, consiste en las oraciones con la puntuación más alta.

¿Qué hace **TextRank**?

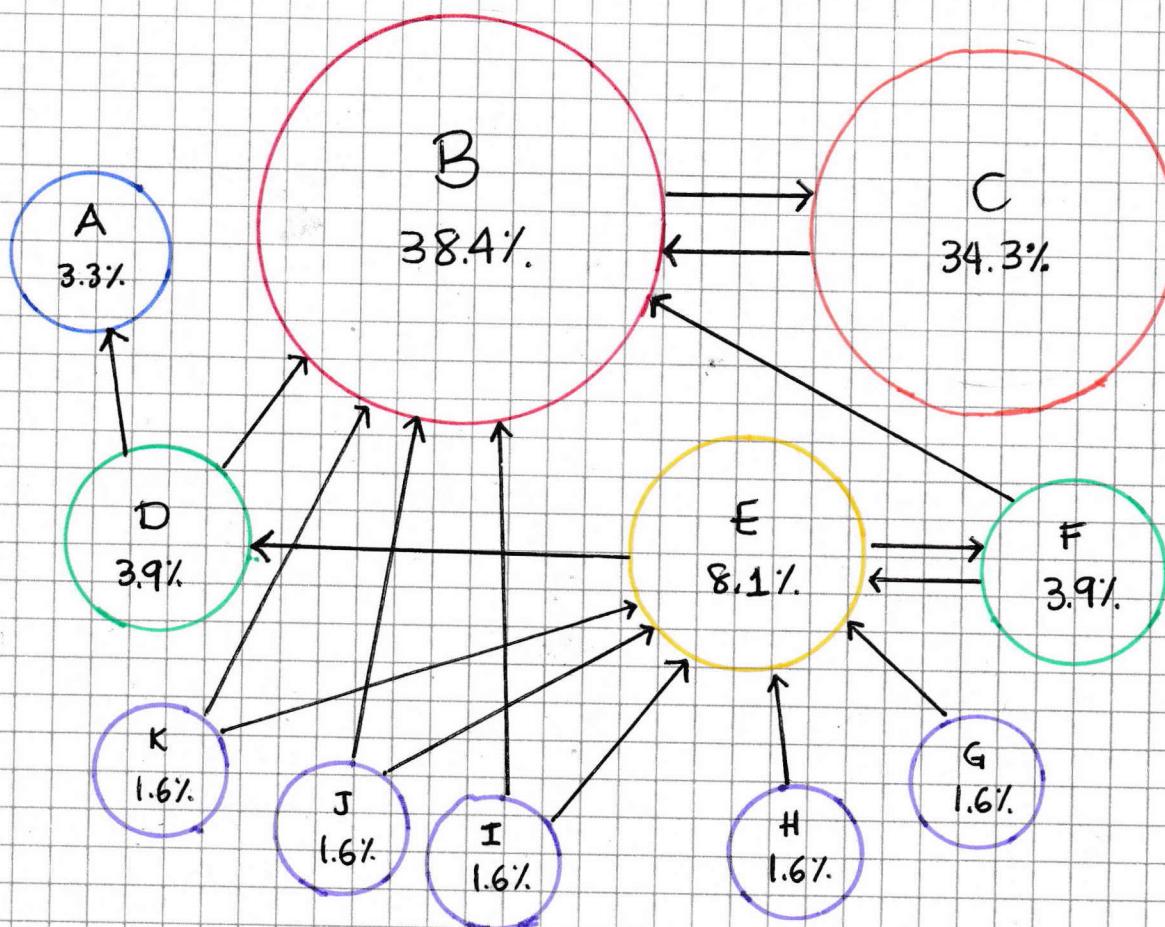
A diferencia de simplemente tomar el **promedio** de los valores **TF-IDF** **TextRank** se inspira en cómo las oraciones están relacionadas entre sí, en términos de contenido y contexto.

¿Cómo funciona **PageRank** de Google?

Detrás del funcionamiento de **TextRank** encontramos **PageRank**, que funciona de la siguiente manera:

1. Cada página web se considera un **nodo** en la red.
2. Se asigna una puntuación a cada **página web** basada en la cantidad y calidad de los enlaces entrantes.
3. El internet se visualiza como una red de **páginas web** o **nodos** interconectados.

La idea detrás de **PageRank** es que una **página** es importante si otras páginas importantes se enlazan.



Aplicación de TextRank

Para adaptar el concepto de **PageRank** al texto:

1. Se tratan las oraciones como páginas web o nodos.
2. Los enlaces entre oraciones se determinan mediante la similitud del contenido (coseno-similitud de sus vectores TF-IDF).
3. Las oraciones que son similares a muchas otras oraciones importantes obtienen una puntuación más alta.

Proceso del TextRank

1. Representación basada en Grafos.
2. Métrica de similitud.
3. Mecanismo de clasificación.
4. Suavizado y regularización.
5. Interpretación del resultado.

Paso a paso del TextRank

1. Procesar el documento: Tokenizar el documento en oraciones.
2. Vectorizar: Convertir oraciones o palabras en vectores (TF-IDF, BOW).
3. Similitud: Crear una matriz de similitud donde cada elemento representa la similitud entre dos oraciones o palabras.
4. Normalizar: Asegurarse que cada fila de 1. \Rightarrow Suavizado-Markov
5. Grafo: Usando la matriz de similitud, crear un grafo donde los nodos representen oraciones o palabras y los bordes o líneas representen puntuaciones de similitud.
6. Puntuar Nodos: Implementar el mecanismo de clasificación de TextRank para puntuar cada nodo.
7. Extraer Resultados: Para resumir, seleccionar las oraciones con la mejor clasificación. Para la extracción de palabras clave, elegir las palabras mejor clasificadas.

Aplicación de resumen de textos utilizando TextRank

Ejemplo Práctico: *Proyecto15.pdf* Anexo 15

Para desarrollar este ejercicio, utilizaremos la misma base de datos del *Proyecto14.ipynb* del Módulo 3.

Tengamos en cuenta que la base de datos ya está en la ruta correcta según el ejemplo mencionado.

Modelado de Temas con LDA

LDA (Latent Dirichlet Allocation) es un método de modelado de temas que identifica temas latentes en un conjunto de documentos.

Utiliza la distribución de Dirichlet para determinar la probabilidad de que ciertas palabras aparezcan juntas en documentos y, por lo tanto, pueden ser consideradas como parte del mismo tema.

A grandes rasgos, el LDA trata de determinar qué palabras son más probables que aparezcan en los mismos documentos y, basándose en eso, decide qué documentos tratan del mismo tema.

Aplicaciones del LDA

Descubrimiento de temas

Identificar de qué habla un documento

Reducción de dimensionalidad

Agrupar y clasificar por temas

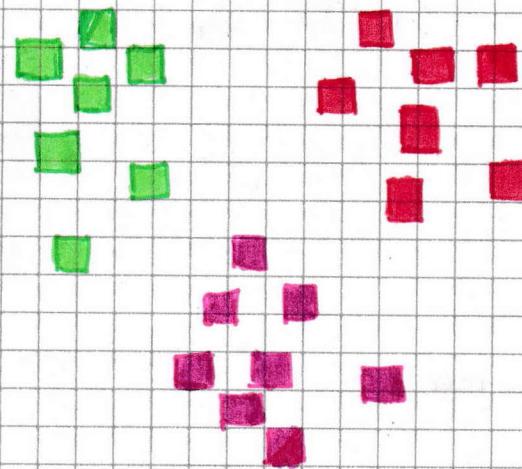
Recomendaciones

Direccionalizar búsquedas

Aprendizaje No supervisado con LDA

A diferencia del aprendizaje supervisado que usa datos etiquetados, el LDA no usa etiquetas.

Por ejemplo, utiliza clustering, donde los datos se agrupan basados en similitudes. Esto tiene el gran desafío de no conocer el número de clusters, el cual debe ser definido por el usuario.



Outputs and Inputs of LDA

Entradas: Se utilizan vectores de cuenta (**BoW**) que no consideran el orden de las palabras.

Salidas:

Matriz de temas por palabras: Representa la probabilidad de que una palabra pertenezca a un tema.

Matriz de documentos por temas: Representa la probabilidad de que un documento pertenezca a un tema.

Método de conteo

Revisemos este tema en la página 5 del **Módulo 1**.

LDA

La probabilidad marginal de los datos está dada por:

$$P(w|\alpha, \beta) = \int P(\theta, \phi, z, w | \alpha, \beta) d\theta d\phi dz$$

donde,

α \Rightarrow Vector que representa el parámetro de **Dirichlet** que controla la distribución de temas por documento (prior de Θ_d).

β \Rightarrow Vector que representa el parámetro de **Dirichlet** que controla la distribución de palabras por tema (prior de ϕ_k).

Θ_d \Rightarrow **Dirichlet**(α) es la proporción de temas para el documento d .

ϕ_k \Rightarrow **Dirichlet**(β) es la proporción de palabras para el tema k .

Para cada palabra w del documento:

Elegimos un tema $Z_{d,n} \sim \text{Multinomial}(\Theta_d)$

Elegimos una palabra $W_{d,n} \sim \text{Multinomial}(Z_{d,n})$

Aplicación de modelado de temas con LDA

Ejemplo Práctico: Proyecto16.pdf Anexo 16

Para desarrollar este ejercicio, debemos ir al siguiente enlace:

huggingface.co/datasets/FrancophonIA/TED_talks/blob/main/ted-talks_es.csv

hacer clic en **download** y almacenar la base de datos en la ruta **Guía IA 2025/Modulo3/data** de la ubicación raíz de los proyectos anexada jupyter.

Módulo 4

Deep Learning y Redes Neuronales para NLP

Introducción al Deep Learning (Aprendizaje Profundo)

El Deep Learning es una rama de la Inteligencia Artificial que usa Redes Neuronales Artificiales con muchas capas para aprender patrones complejos en grandes cantidades de datos.

Introducción a TensorFlow

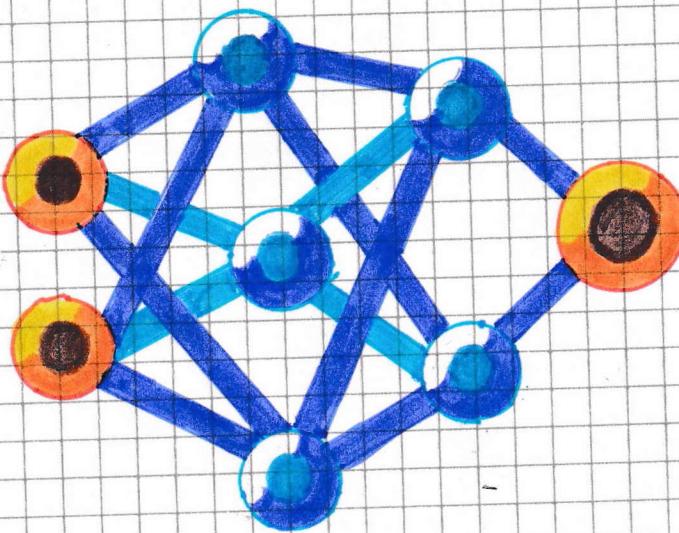
Tenemos la necesidad en python de utilizar una biblioteca que permita el manejo de las redes neuronales artificiales, para ello, tenemos TensorFlow.

Es una biblioteca de código abierto para realizar cálculos numéricos y construir modelos de aprendizaje automático ML.

Desarrollada por Google Brain Team, es ampliamente utilizada en Inteligencia Artificial y ML.

TensorFlow, permite definir y ejecutar algoritmos eficientemente en CPU, GPU y TPU. Ofrece una interfaz en Python, facilitando su integración con otras bibliotecas.

Se utiliza en aplicaciones como visión por computadora, procesamiento de lenguaje natural y reconocimiento de voz.



Redes Neuronales Convolucionales (CNNs)

Una red neuronal convolucional (CNN) es un tipo de red neuronal que se especializa en el procesamiento de datos estructurados, especialmente en tareas de visión por computadora. Utiliza capas de convolución para extraer características clave de las entradas y es ampliamente utilizada en reconocimiento de imágenes y detección de objetos.

Por ejemplo, imaginemos que tenemos una imagen (una matriz de píxeles) y una de filtros (Kernels), que son pequeñas ventanas (por ejemplo, 3×3 o 5×5) que "se deslizan" sobre la imagen para detectar patrones.

En cada posición, el filtro realiza una operación matemática llamada convolución (multiplicación elemento a elemento y suma), y así genera una nueva imagen transformada, llamada mapa de activación o feature map.

Redes Neuronales Recurrentes (RNNs)

Las redes neuronales recurrentes (RNNs) son un tipo de arquitectura de red neuronal diseñada para trabajar con datos secuenciales, como series temporales, texto o audio.

Utilizan conexiones recurrentes entre neuronas, lo que les permite mantener y procesar información a lo largo del tiempo, lo que las hace especialmente adecuadas para problemas de predicción y modelado de secuencias.

Las RNNs son capaces de aprender dependencias temporales y contextuales en los datos de entrada, lo que las hace útiles en una amplia gama de aplicaciones, como traducción automática, reconocimiento de voz y generación de texto.

Por ejemplo, imaginemos que tenemos la frase "los gatos duermen todo el..." y queremos predecir la siguiente palabra. Una RNN procesará palabra por palabra, recordando el contexto anterior:

1. Los → guarda algo de información (plural, artículo definido).
2. gatos → añade que hablamos de animales, en plural.
3. duermen → entiende que la acción es dormir.
4. todo → prepara el contexto para una predicción.

Con esta secuencia, la RNN podrá predecir "el día", porque ha aprendido que esa frase es común.

Modelado y TensorFlow

Tenemos:

Capas de Entrada y Densa: La primera capa es la de entrada que recibe los datos con dimensiones D . La siguiente es una capa densa, que realiza una transformación afín

$$W\mathbf{x} + b$$

donde, W es una matriz de pesos y b es un vector de sesgo.

Función de activación Sigmoidal: Para la clasificación binaria, se aplica una función sigmoidal después de la capa densa para mapear la salida a una probabilidad entre 0 y 1.

Compilación del Modelo: Se utiliza la entropía cruzada binaria como función de pérdida, y Adam como optimizador. La métrica de precisión accuracy se utiliza para evaluar el rendimiento del modelo.

Entropía Cruzada Binaria

Esta función de pérdida mide el rendimiento de un modelo de clasificación cuyas salidas son probabilidades. Es mínima cuando la predicción es idéntica a la etiqueta real y se approxima al infinito cuando son opuestas.

$$L = - \sum_{i=1}^N (y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i))$$

Estabilidad numérica

Funciones como la exponencial y la logarítmica, pueden ser numéricamente inestables, TensorFlow combina la sigmoidal y la función de pérdida de manera que impide valores inestables.

Regresión lineal con TensorFlow

Es un modelo que intenta ajustar una línea recta a los datos para predecir una variable y a partir de una variable x , con la fórmula:

$$y = \omega x + b$$

donde,

ω es el peso (pendiente).

b es la intersección (bias) o (sesgo).

y es la salida predicha.

En una neurona, la salida se calcula como:

$$y = f(\omega_1 x_1 + \omega_2 x_2 + \dots + \omega_n x_n + b)$$

donde f es la función de activación.

Ejemplo Práctico: [Proyecto17.pdf](#)

Anexo17

En este ejercicio, se desarrollará una regresión lineal utilizando TensorFlow, con datos numéricos, de tal manera que se pueda explicar de mejor manera el objetivo de dicho análisis estadístico.

Clasificador de texto con TensorFlow

Clasificación Binaria: Es un tipo de problema de aprendizaje supervisado en el que el objetivo es predecir cuál de dos clases posibles pertenece cada instancia de datos. Este tipo de clasificación es fundamental en el campo de la ML y tiene aplicaciones en múltiples dominios, como la detección de spam y el análisis de sentimientos en textos.

Aplicación de clasificador con TensorFlow

Ejemplo Práctico: Proyecto18.pdf Anexo 18

Para desarrollar este ejercicio, debemos ingresar a [Kaggle.com](https://www.kaggle.com) y buscar el siguiente dataset:

Sentiment of reviews Movies

o simplemente, ingresar al enlace:

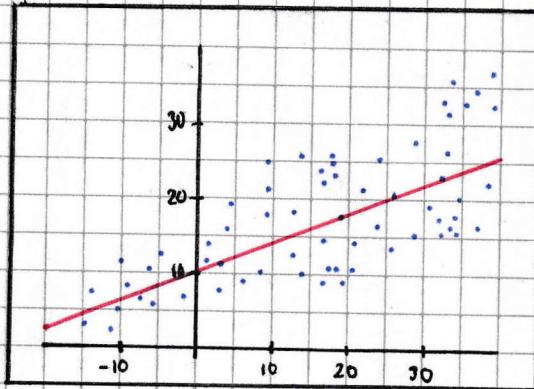
[Kaggle.com/datasets/luisfester2025/sentiment-of-reviews-movies](https://www.kaggle.com/luisfester2025/sentiment-of-reviews-movies)

y descargar la base de datos en la carpeta del proyecto [Guia IA 2025](#) de la ubicación raíz de los proyectos de anaconda-jupyter.

La Nerona

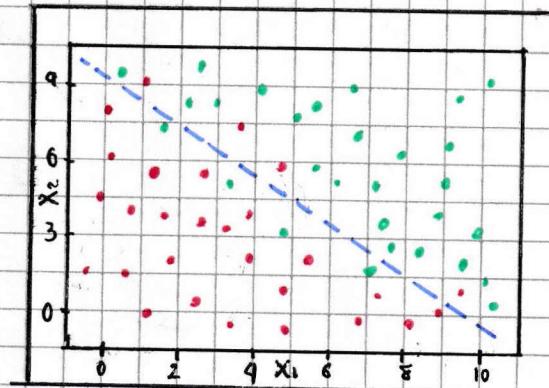
Vamos a revisar algunos conceptos para determinar este elemento, como base fundamental de la inteligencia artificial.

Regresión Lineal vs Regresión Logística



$$Y = WX + b$$

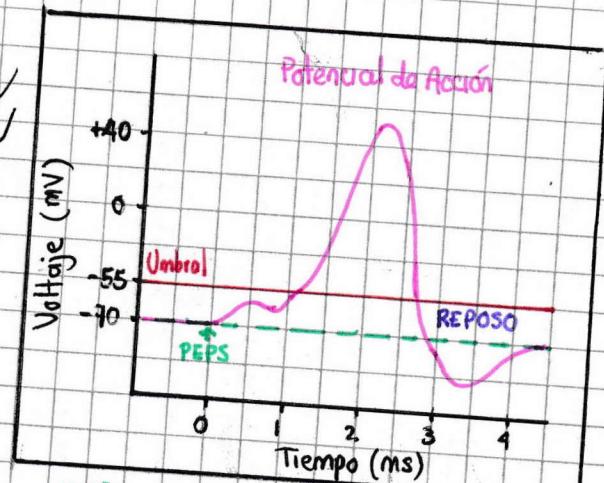
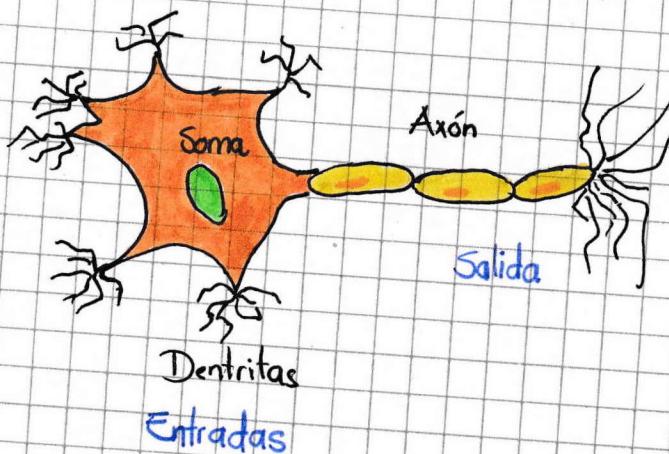
Predice el comportamiento



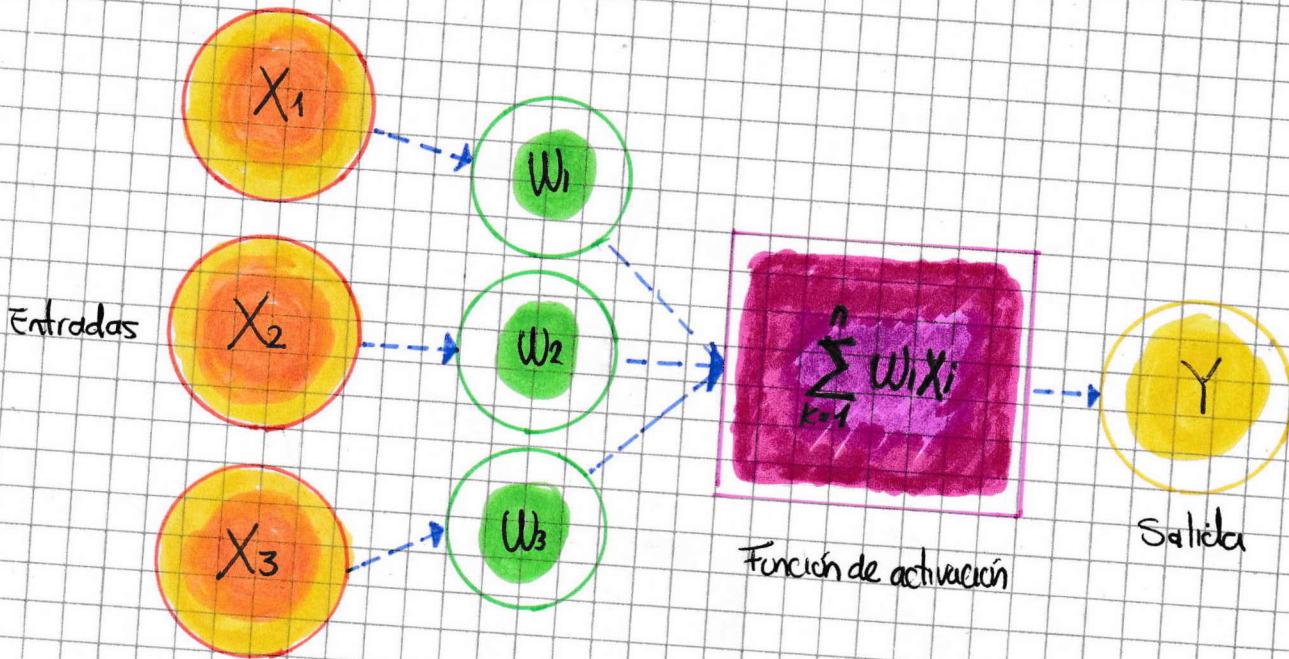
$$P(Y=1) = \frac{1}{1 + e^{mx+b}}$$

Clasifica

De la regresión a las Neuronas



Neurona Artificial



Una neurona decide si pasa o no la señal cuando se aderva el umbral de activación, similar al concepto de clasificación en la regresión logística. Para la activación, tenemos:

- Función sigmoidal
- Función Tangente Hiperbólica (tanh)
- Función de Unidad Lineal Rectificada (ReLU)
- Softmax

Aplicaciones de las neuronas artificiales

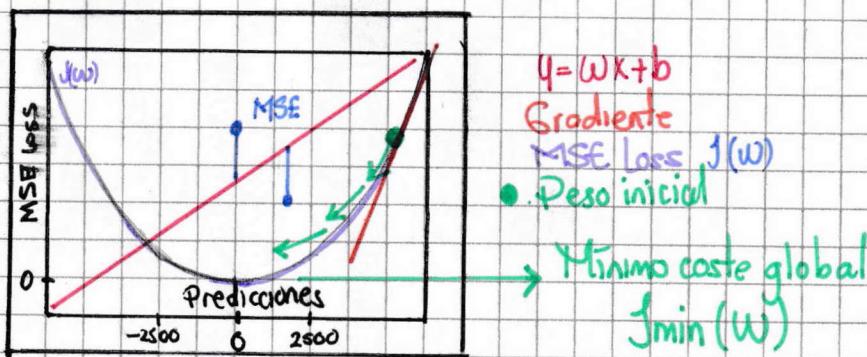
- Reconocimiento de imágenes
- Visión por computadora
- Procesamiento de lenguaje NLP
- Predicciones financieras
- Juegos
- Medicina
- Sistemas de recomendación
- Robótica
- Agricultura inteligente
- Generación y modificación de contenido
- Vehículos autónomos

¿Cómo aprende un modelo?

A través de la función de regresión lineal, se busca encontrar la distancia mínima promedio de las predicciones, es decir, encontrar una recta que se acerque a todos los puntos con el menor error. Para ello se usa el error cuadrático medio

$$MSE = \frac{1}{n} \sum (y - \hat{y})^2$$

donde el error mínimo se da en la parte mínima de la curva, cuando debería ser 0.

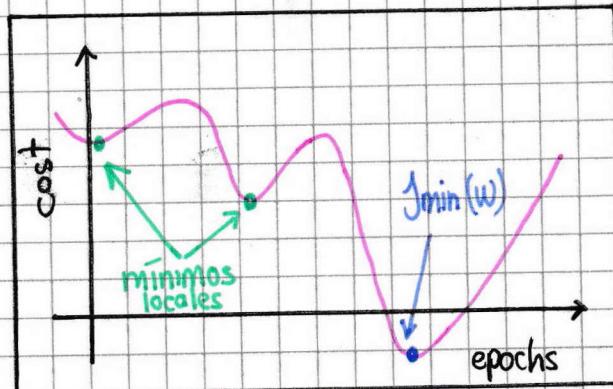


Para ello analizamos el descenso del gradiente, un método iterativo que ajusta gradualmente w para reducir el coste. Se emplea con valores aleatorios y, en cada paso, movemos estos valores en la dirección que más reduce el error.

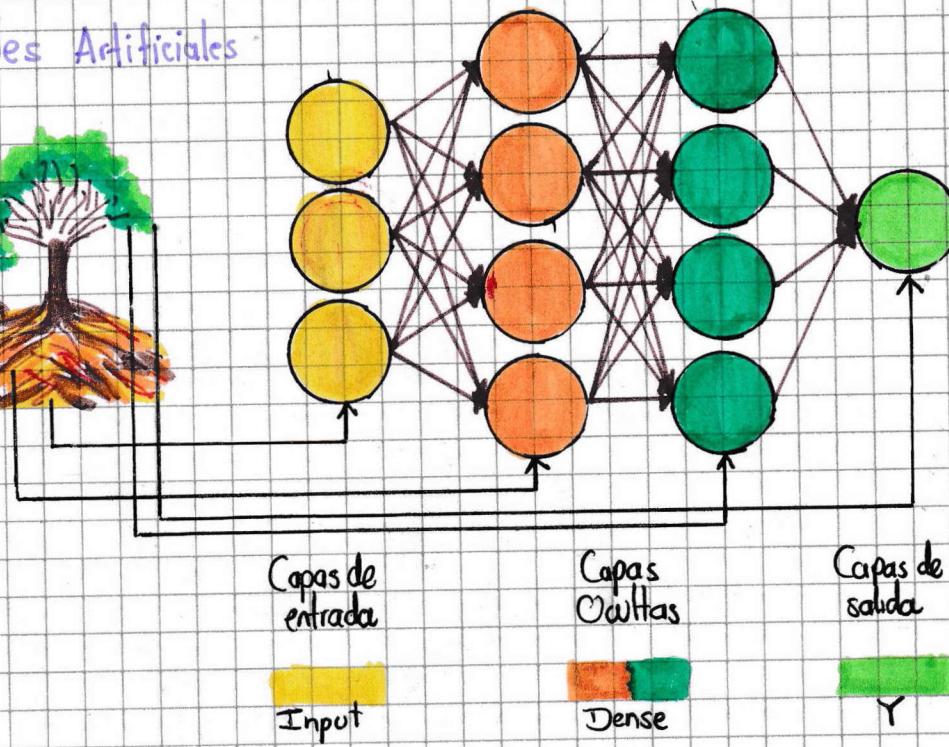
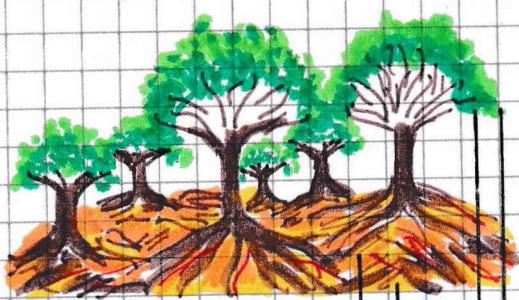
La tasa de aprendizaje es el espacio entre cada paso (el salto) para acercar el MSE a 0.

Tasa de aprendizaje

Determina cuán grandes son estos pasos. Si es demasiado alta, podríamos saltarnos la solución. Si es demasiado baja, el proceso podría ser muy lento o detenerse prematuramente.



Redes Neuronales Artificiales



El simbol anterior, representa la manera en que las redes neuronales se interconectan, las raíces de cada árbol responden a las conexiones entre cada uno (neurona) y de esa manera pasa por el tallo y las ramas (capas densas) para entregar un resultado (Y o predicción).

Forward Propagation

Es el proceso de pasar los datos de entrada por toda la red neuronal, capa por capa, hasta obtener una salida.

Supongamos una red simple con

- 1 capa de entrada (3 neuronas)
- 1 capa oculta (2 neuronas)
- 1 capa de salida (1 neurona)

1. Entradas → Neuronas de la primera capa

Para cada neurona:

1. Multiplica cada entrada por su peso.
2. Suma todos los valores
3. Suma un sesgo (umbral)
4. Aplica una función de activación

$$z = (w_1 x_1 + w_2 x_2 + w_3 x_3) + b$$

$$a = f(z)$$

donde:

x → valores de entrada

w → pesos

b → sesgo (bias)

f → función de activación (ReLU, Sigmoid, etc.)

a → activación (salida de la neurona)

2. Capas ocultas → Neuronas de la segunda capa

Repite el paso 1. en las capas ocultas, ya que las salidas de una capa se convierten en las entradas de la siguiente capa.

3. Salida → Neuronas de la última capa

Al final, obtenemos un valor de salida:

- Un número (regresión)
- Una probabilidad (clasificación)
- Una etiqueta (clase más probable)

Observemos este paso a paso, ahora con datos numéricos:

Supongamos:

1. Entrada $\rightarrow X = [1.0, 2.0]$
 Pesos $\rightarrow W = [0.5, -1.0]$
 Sesgo (bias) $\rightarrow b = 0.1$

Activación de función sigmoidal:

$$z = (1.0)(0.5) + (2.0)(-1.0) + 0.1$$

$$z = 0.5 - 2.0 + 0.1$$

$$z = -1.4$$

$$a = \sigma(z) = \frac{1}{1 + e^{-(-1.4)}} \rightarrow a \approx 0.197$$

La salida de esa neurona sería ≈ 0.197

Importancia de las redes neuronales

Si tenemos una ficha de rompecabezas, una sola neurona puede determinar las características de la ficha, pero, no tiene la habilidad de predecir a qué rompecabezas pertenece.

Las redes de varias neuronas, pueden percibir distintas ideas de una posible solución al rompecabezas, y de acuerdo a su aprendizaje, pueden estructurar predicciones para que esa ficha se acomode al mejor rompecabezas.

A modo de comparación, un modelo de regresión lineal o regresión logística, simbolizan una neurona pero una estructura de capas de estos modelos aplicados a sus posibles variantes, representan una red neuronal.

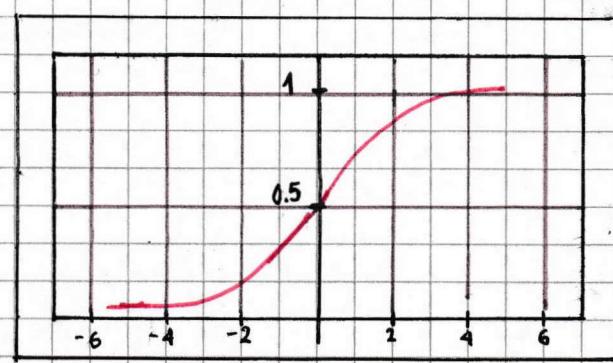
Funciones de Activación

$$z = \sum_{k=1}^n (W_k X_k) + b$$

Función Sigmoidal

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

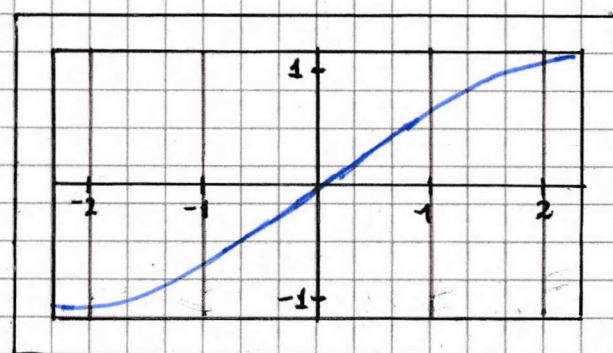
Útil para probabilidades por que sus valores son entre 0 y 1.



Función Tangente Hiperbólica

Útil para clasificación por su dominio $[-1, 1]$ y su rango $[-1, 1]$.

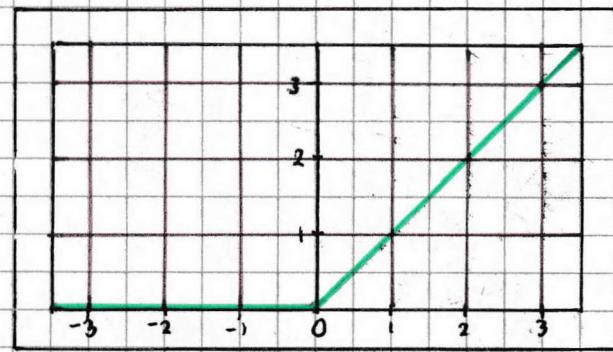
$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$



Función ReLU

Útil en ambos casos, ya que se desactiva cuando es menor que cero y entrega el mismo valor positivo de entrada ya predicho.

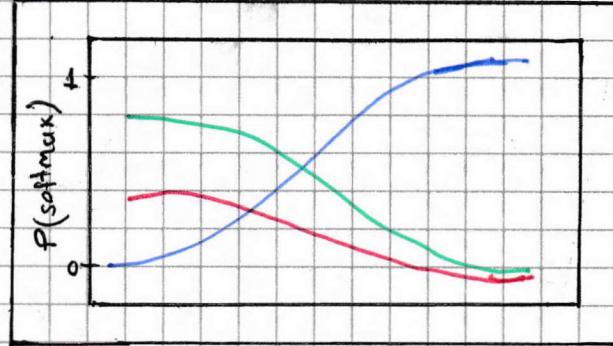
$$R(z) = \max(0, z)$$



Función Softmax

Útil para salidas multiclase, donde no tenemos clases binarias.

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$



En todos los puntos, la probabilidad debe ser 1.0

Embeddings

Es una representación vectorial de una palabra que captura su esencia y su relación con otras palabras. El Deep learning, permite que una máquina comprenda el texto no como una serie de caracteres dislados, sino como ideas y conceptos interconectados.

Word	Word Embedding							Dimensionality reduction	Visualization 2D
	Living being	feline	human	gender	royalty	verb	plural		
cat	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2		
kitten	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1	from 7D to 2D	houses cat dog woman man queen King
dog	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3		
houses	0.1	-0.4	-0.5	0.1	-0.9	0.3	0.8		
man	0.6	0.2	0.8	0.9	-0.1	-0.9	-0.7		
women	0.7	0.3	0.9	-0.7	0.1	-0.5	-0.4		
King	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6		
queen	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9		

Codificación One-Hot

Cada palabra se convierte en un vector enorme con un 1 en una ubicación y de ceros en todas las demás. Para un lenguaje con miles de palabras, esto es como buscar una aguja en un pajar computacional.

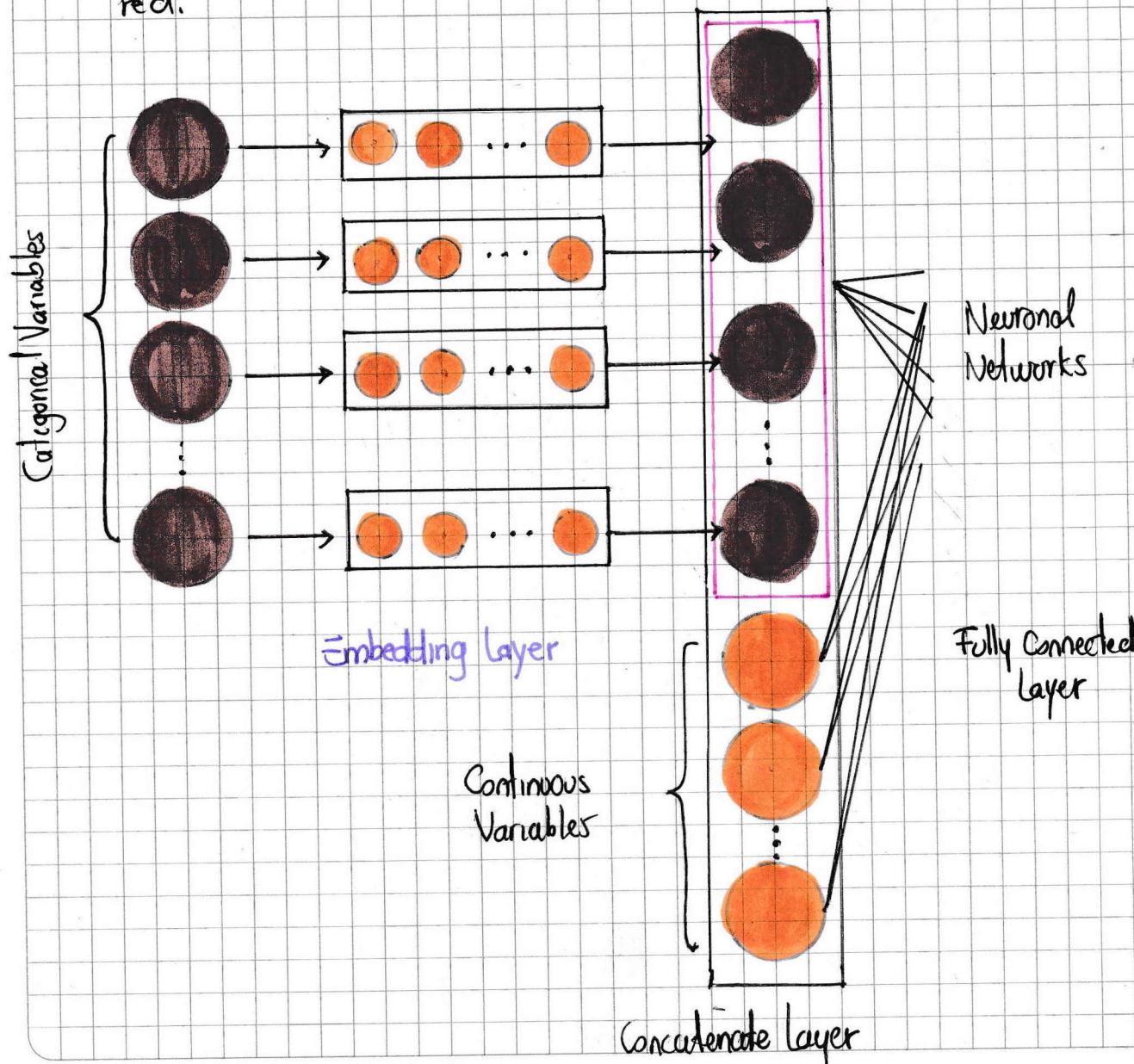
	a	cat	is	this	...
this	0	0	0	1	...
is	0	0	1	0	...
a	1	0	0	0	...
cat	0	1	0	0	...
				:	

Cumple con la función aunque poco eficiente, lo que implica alto coste computacional.

Los **embeddings** ofrecen un atajo. En lugar de esa multiplicación masiva, simplemente seleccionamos el vector correspondiente de una matriz de pesos ya precalculada. Esto es como tener una biblioteca en la que puedes sacar un libro instantáneamente solo con su número de referencia, en lugar de tener que buscar en toda la estantería.

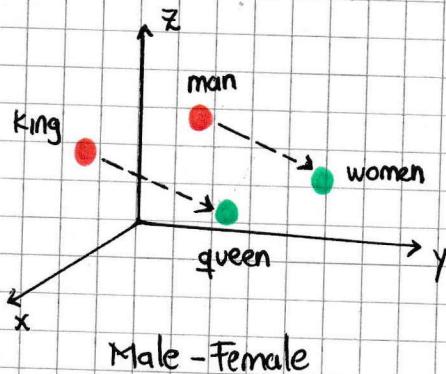
¿Cómo funcionan los Embeddings en una red neuronal?

Cuando construimos una **NN** para NLP empezamos con una capa de entrada, que se alimenta a una capa de **embedding**. Aquí, cada palabra es representada por un vector denso. Esto convierte nuestro texto en una serie de vectores ricos en información, listos para ser procesados por la red.



Geometría de los Embeddings

Los **embedding** organizan palabras en un espacio **geométrico** donde la distancia y la dirección tienen significado. Palabras con significados similares están más cerca entre sí. Esto no es solo matemática, es una representación del entendimiento humano del lenguaje.



Country - Capital	
Spain	Madrid
Italy	Rome
Germany	Berlin
Turkey	Ankara
Russia	Moscow

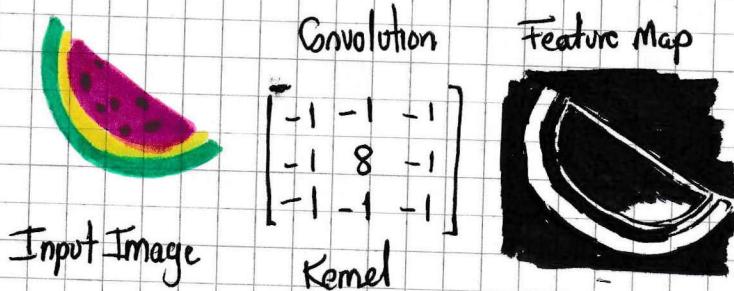
Redes Neuronales Convolucionales CNN

La convolución es una operación matemática simple que se basa en dos acciones: sumar y multiplicar, como vimos en el módulo anterior.

Imagenes y Filtros (o Núcleos): En el contexto de las imágenes, la convolución implica una imagen de entrada, un filtro y produce una imagen de salida.

Operación: Consiste en aplicar el filtro sobre la imagen de entrada para obtener una imagen de salida modificada.

Desenfoque y Detección de Bordes: Se desenfocan las imágenes y se obtienen los bordes. La diferencia en la salida resalta del tipo de núcleo que se elija.



Convolución en redes neuronales

Filtros: En las **CNNs**, los filtros son aprendidos automáticamente para realizar tareas específicas como reconocer patrones o características en las imágenes.

Importancia de entender la convolución: Aunque las bibliotecas de aprendizaje profundo realizan estas operaciones automáticamente, entender la convolución ayuda a comprender cómo funcionan las redes neuronales convolucionales desde un nivel más fundamental.

¿Cómo funciona la convolución?

La **convolución** es como un filtro mágico que pasa por encima de una imagen, cambiándola de maneras específicas según las instrucciones del filtro. Imagina que tienes una foto y sobre ella pones una lupa que puede resaltar bordes o hacer que todo se vea borroso. Esta lupa es el filtro, y el efecto que produce (detectar bordes o borrar) depende de las instrucciones dadas.

En el mundo de las **NN**, estas instrucciones no las decidimos nosotros directamente. En su lugar, la red aprende cuáles son las mejores instrucciones para realizar tareas como reconocimiento de rostros, objetos o paisajes. Lo interesante es que todo se reduce a saber sumar y multiplicar: De esta manera, hacemos que una computadora "vea" y entienda las imágenes.

Finalmente, hay algunas técnicas y trucos que nos permiten controlar cómo queremos que se transforme la imagen de entrada. Esto es crucial para diseñar **NN** que funcionen adecuadamente en la práctica.

Algunas técnicas son, Identity, Ridge detection, Sharpen, Box blur, Gaussian blur; (3×3) y (5×5) , Unsharp masking 5×5 , entre otras.

$$\begin{array}{c}
 \begin{matrix}
 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
 1 & 1 & 0 & 0 & 0 & 0 & 0
 \end{matrix}
 \quad \text{I}
 \end{array}
 \quad
 \begin{array}{c}
 \begin{matrix}
 1 & 4 & 3 & 4 & 1 \\
 4 & 2 & 4 & 3 & 3 \\
 1 & 2 & 3 & 4 & 1 \\
 1 & 3 & 3 & 1 & 1 \\
 3 & 3 & 1 & 1 & 0
 \end{matrix}
 \quad \text{K}
 \end{array}
 \quad
 \begin{array}{l}
 \text{I} * \text{K} \\
 (1 \times 1 + 0 \times 0 + 0 \times 1) + (1 \times 0 + 1 \times 1 + 0 \times 0) + \\
 (1 \times 1 + 1 \times 0 + 1 \times 1) = 1 + 1 + 2 = 4
 \end{array}$$

Pattern Matching (Emparejamiento de patrones)

Es una técnica utilizada en la computación para verificar la presencia de subsecuencias (patrones) dentro de un conjunto de datos más grande.

Es un concepto ampliamente aplicado en varios campos de la informática y la tecnología, como el NLP, búsqueda y reemplazo en el procesamiento de texto, diseño de algoritmos, análisis de datos y en ML y visión por computadora.

En NLP, el pattern matching se utiliza para entender y procesar el lenguaje humano. Esto incluye tareas como el análisis sintáctico, donde se buscan partes gramaticales en las oraciones.

Desde esta perspectiva, un filtro (Kernel) en la convolución, actúa como un detector de patrones que se desliza sobre la imagen de entrada, buscando similitudes con el patrón que contiene.

Cuando el patrón del filtro y un segmento de la imagen coinciden estrechamente, el resultado es un valor alto, indicando una fuerte presencia del patrón en esa posición de la imagen.

La operación de convolución, permite buscar coincidencias entre la imagen y un patrón específico representado por el filtro. La eficiencia y la potencia de este proceso radican en la capacidad de hacer estas comparaciones rápidamente a través de operaciones vectorizadas, sin tener que examinar cada parte de la imagen paso a paso de manera ineficiente.

Weight Sharing (Compartición de pesos)

Esta estrategia implica utilizar el mismo conjunto de datos (parámetros) para más de una unidad en un modelo.

En las CNNs, el compartir pesos se refiere principalmente al uso de filtros (Kernels) que se aplican a diferentes partes de la imagen de entrada. En lugar de aprender un conjunto único de pesos para cada ubicación de la imagen, un filtro aprende un conjunto de pesos y luego se "desliza" sobre toda la imagen, aplicando esos mismos pesos en cada ubicación. Esto significa que el filtro está buscando el mismo patrón o característica en todas las partes de la imagen.

Ventajas del Weight Sharing

Reducción de la complejidad computacional: Al compartir pesos, el número total de parámetros que la red necesita aprender se reduce significativamente. Esto hace que el modelo sea menos complejo y más eficiente desde el punto de vista computacional.

Mejora de la generalización: Al forzar a la red a utilizar los mismos pesos en todas partes, se promueve la detección de características independientemente de la ubicación en la imagen. Esto ayuda a que la red generalice mejor a nuevos datos, ya que aprende características útiles que son invariantes a la posición.

Capacidad de manejar imágenes de diferentes tamaños: Debido a que los filtros con pesos compartidos se pueden aplicar a imágenes de cualquier tamaño, las **CNNs** son flexibles en el manejo de entrada de diferentes dimensiones, lo cual es útil en aplicaciones reales donde el tamaño de las imágenes pueden variar.

Convolución con imágenes a Color

Las imágenes a color tienen tres dimensiones: alto, ancho y canales de color (generalmente **RGB**); A diferencia de las imágenes en escala de grises que son bidimensionales.

Filtros 3D

Para trabajar con imágenes a color, los filtros (**kernels**) también deben ser tridimensionales. Esto permite que los filtros detecten patrones específicos en los canales de color de la imagen.

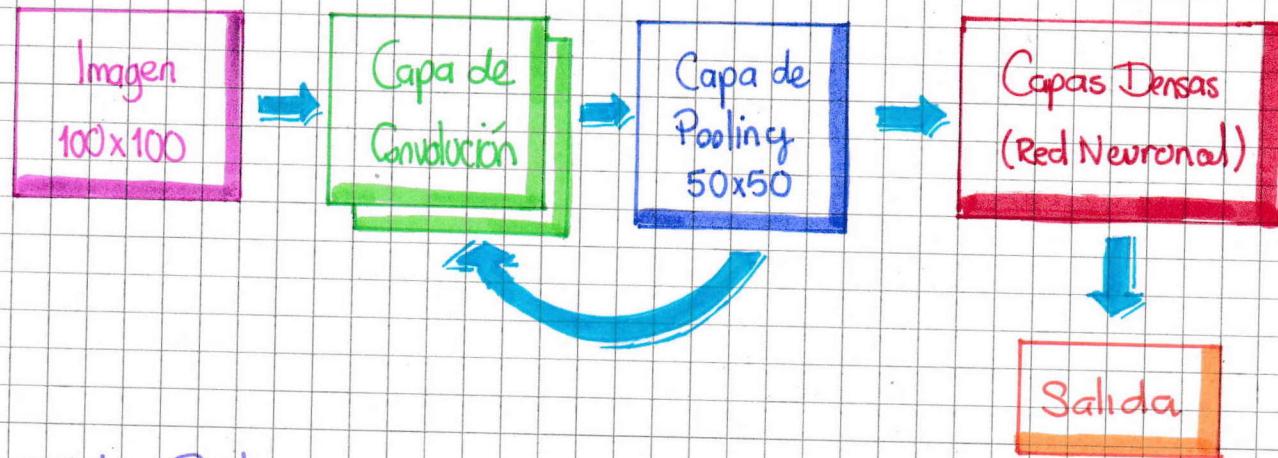
La convolución en este contexto sigue siendo una operación de sumar y multiplicar, pero extendida a través de los tres ejes. Esto significa que cada filtro se "desliza" a través de la imagen en el plano 3D.

Aunque la entrada y el Kernel son tridimensionales, el resultado del resultado genera una imagen bidimensional, esto se debe a colapsar la información de color en un solo valor de intensidad por ubicación del filtro.

Esto genera problemas de uniformidad, ya que la salida es 2D y no podemos aplicar una nueva capa 3D al resultado previo.

La solución está en usar **Filtros múltiples** que se encargan cada uno de una característica diferente de la imagen, lo que genera múltiples imágenes bidimensionales como salida (una por filtro).

Arquitectura de una CNN



Max Pooling.

1	2	9	5
5	7	7	2
3	4	12	5
8	10	11	6

Pooling

7	9
10	12

Average Pooling

3,75	5,75
6,25	9,5

¿Porqué es importante el pooling?

Reduce la cantidad de información que la red necesita procesar, permitiendo una computación más rápida y eficiente.

Además, al seleccionar características dominantes (max pooling) o promediar características (average pooling), la red puede mantener la robustez frente a pequeñas variaciones de la imagen o del texto procesado.

Convenciones y Estrategias en la estructura CNN

Las CNNs aprenden características de manera jerárquica; las capas iniciales aprenden detalles simples y las capas más profundas aprenden características más complejas.

A medida que la imagen se reduce por las capas de pooling, los filtros cubren proporcionalmente áreas más grandes, permitiendo detectar patrones más grandes y complejos.

Perdida de información

Aunque las capas de pooling resultan en una pérdida de información, las CNNs lo compensan aumentando el número de mapas de características como lo vimos anteriormente.

Flexibilidad de la arquitectura

Tamaños de filtros

Tamaños de pooling

Tipos de filtros

Tipos de pooling

CNNs para textos

Una secuencia, como una serie de palabras o datos temporales, se diferencia de las imágenes en que es unidimensional (solo una dimensión de tiempo o secuencia). Sin embargo, al igual que en las imágenes, en las secuencias también podemos correlacionar: datos cercanos en tiempo suelen ser similares o relacionados.

En lugar de usar un filtro bidimensional como en imágenes, para las secuencias usamos un filtro unidimensional. Este filtro se "desliza" a lo largo de la secuencia, multiplicando y sumando valores de manera similar a como lo haría en una imagen. Esta operación se conoce como convolución unidimensional.

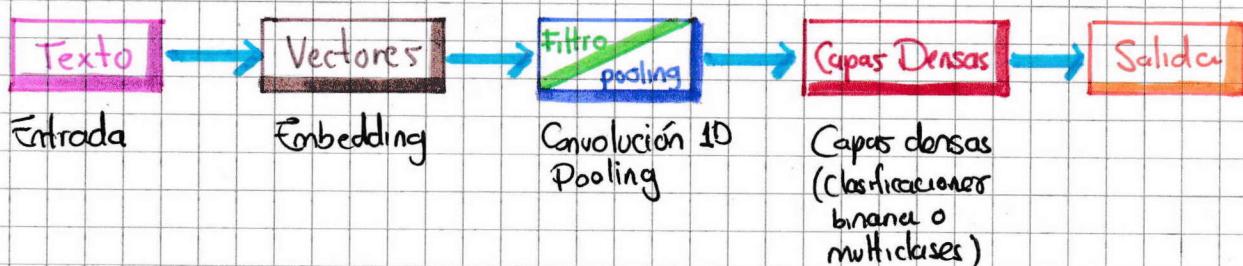
Esto lo podemos entender mejor con el siguiente ejemplo.

Supongamos que tenemos una secuencia de números y un filtro:

$$[1, 2, 3, 2, 1] * [1, -1] \rightarrow [-1, -1, 1, 1]$$

$$\begin{aligned} (1 \times 1 + 2 \times (-1)) &= 1 - 2 = -1 \\ (2 \times 1 + 3 \times (-1)) &= 2 - 3 = -1 \\ (3 \times 1 + 2 \times (-1)) &= 3 - 2 = 1 \\ (2 \times 1 + 1 \times (-1)) &= 2 - 1 = 1 \end{aligned}$$

Arquitectura en Textos



Aplicación de las CNNs en NLP

Ejemplo práctico: Proyecto22.pdf Anexo 22

Para desarrollar este ejercicio implementaremos los archivos del Proyecto 21 el cual nos arroja el modelo, tokenizador y el archivo de datos entrenados.

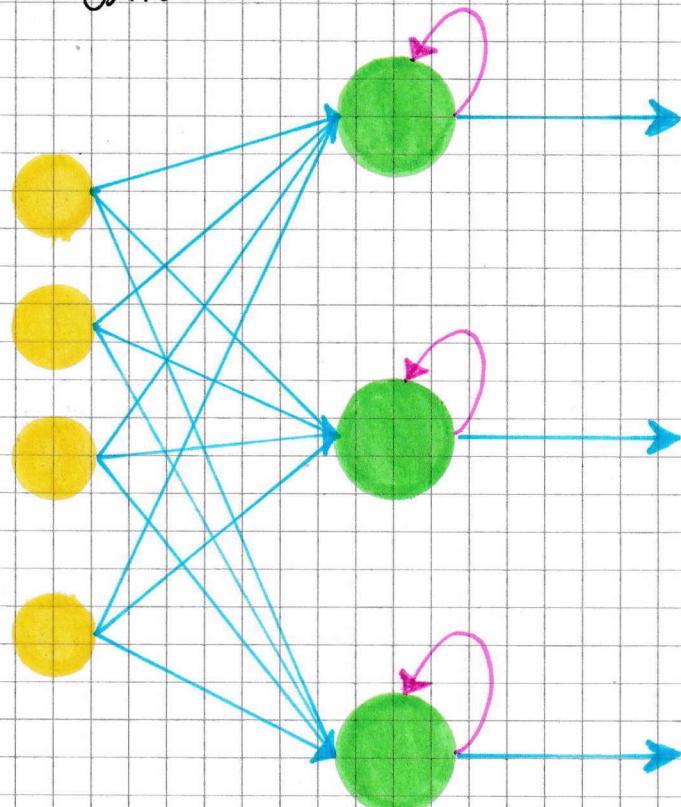
Con estos elementos, implementaremos un código .py de tal forma que podamos diseñar un dashboard para Streamlit.

Posteriormente, lo subimos como repositorio público en GitHub y lo sincronizamos desde streamlit.io.

Redes Neuronales Recurrentes RNN

Son redes capaces de reírse a información pasada, para modificar el presente y predecir el futuro.

Pueden recurrir a capas anteriores o a si mismas, dependiendo del desarrollo contextual.



Supongamos que nuestra tarea es clasificar palabras. Por ejemplo, determinar si una palabra es sustantivo, verbo, etc.

Si usamos una NN regular, esta simplemente toma una palabra y da una etiqueta (predicción). El problema es que la clasificación de una palabra puede depender del contexto.

Contexto 1

Si alguien dice **Encendí una vela**, la palabra **vela** se refiere a una vela de cera que se enciende para dar luz.

En este caso, **vela** es un sustantivo.

Contexto 2

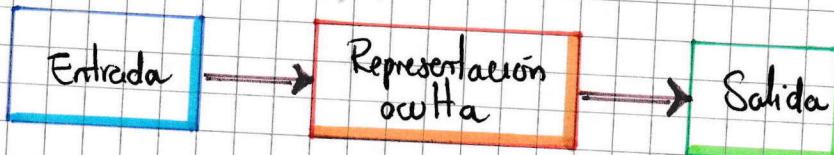
Si alguien dice *El marinero vela toda la noche*, la palabra *vela* se refiere al acto de vigilar o estar despierto.
En este caso, *vela* es un verbo.

Ventajas de las RNNs

Las RNNs son capaces de usar información previa (contexto) para hacer mejores predicciones.

En lugar de solo tomar una entrada actual, también toman en cuenta la representación oculta del paso de tiempo anterior.

Esto significa que, pueden "recordar" información de todos los pasos de tiempo anteriores.



Por ejemplo, supongamos que tenemos la oración *el gato come y* estamos procesando palabra por palabra.

Paso 1: La red recibe la primera palabra *el* y crea una representación oculta basada en esa palabra.

Paso 2: La red recibe la siguiente palabra *gato*. Ahora, en lugar de solo usar *gato* para hacer su predicción, también usa la representación oculta creada en el paso anterior con la palabra *el*.

Paso 3: La red recibe la siguiente palabra *come*. Usa la nueva entrada *come* y la representación oculta actual, que ahora incluye información sobre *el* y *gato*.

Este es un resumen a modo de ejemplo, -a grandes rasgos - de una Red Neuronal Recurrente RNN.

Representación Matemática de una RNN

$$h_1 = \sigma \left(W_h^T h_0 + b_h + W_x^T x_1 + b_x \right)$$

donde,

h_1 : Es el nuevo estado oculto en el primer paso de tiempo.

σ : Es la función de activación.

$W_h^T h_0 + b_h$: Es la contribución de la representación oculta del paso de tiempo anterior.

$W_x^T x_1 + b_x$: Es la contribución de la entrada actual.

Aplicaciones de las RNNs

Las RNNs se utilizan para resolver dos tipos de tareas: "muchos a uno" y "muchos a muchos".

muchos a uno: Tenemos una secuencia de entrada y solo un resultado

Ejemplos:

1. Detección de Spam
2. Análisis de sentimiento

muchos a muchos: Tenemos una secuencia de entrada y una secuencia de resultados.

Ejemplos:

1. **Etiquetados de parte del discurso:** Cada palabra en una oración recibe una etiqueta (sustantivo, verbo, etc.)

Aplicación de un clasificador de textos con RNN

Ejemplo Práctico: Proyecto 23.pdf Anexo 23

Se desarrolla un clasificador RNN, con la estructura del Proyecto 21 implementando LSTM con 32 neuronas.

Ejemplo Práctico: Proyecto 24.pdf Anexo 24

Implementación en Streamlit del proyecto 23.