La Importancia de Regex en Ciencia de Datos y Algoritmos de Machine Learning y Deep Learning

Las **expresiones regulares (regex)** son una herramienta clave en **ciencia de datos**, especialmente en tareas de **preprocesamiento y limpieza de texto**. Los modelos de **Machine Learning (ML) y Deep Learning (DL)** dependen en gran medida de datos estructurados y bien procesados, lo que hace que regex sea una solución eficiente para manipular grandes volúmenes de datos no estructurados.

Aplicaciones Claves en Machine Learning y Deep Learning

1. **Procesamiento de Texto para Modelos de NLP (Procesamiento de Lenguaje Natural)**
   - Eliminación de caracteres especiales, signos de puntuación y espacios extra.
   - Normalización de datos textuales (convertir texto a minúsculas, eliminar acentos, etc.).
   - Extracción de palabras clave y frases relevantes.
2. **Estandarización y Validación de Datos**
   - Detección de patrones como direcciones de correo electrónico, números de teléfono y códigos postales.
   - Filtrado de datos inválidos en bases de datos estructuradas.
   - Identificación de valores duplicados o mal formateados en grandes volúmenes de información.
3. **Optimización de Modelos de Aprendizaje Automático**
   - Mejora en la calidad de los datos de entrada al eliminar ruido en el texto.
   - Creación de características (features) más relevantes para modelos de clasificación y predicción.
   - Uso de regex en la generación de etiquetas para el entrenamiento supervisado.
4. **Filtrado de Datos en Tiempo Real**
   - Detección de palabras clave en flujos de datos como redes sociales, chats y correos electrónicos.
   - Monitorización automática de contenidos en plataformas digitales.
   - Filtrado de datos confidenciales o sensibles antes de su almacenamiento o procesamiento.

Aplicaciones Principales de Regex para Generación de Texto Estructurado y Filtrado Eficiente

Las expresiones regulares se utilizan ampliamente para **transformar, estructurar y filtrar información textual**, permitiendo la creación de datos más organizados y útiles para análisis avanzado.

Generación de Texto Estructurado

Normalización y Homogeneización de Datos

- Uniformización de nombres, direcciones y otras entidades textuales.
- Reformateo de fechas, valores numéricos y estructuras de texto para estandarización.
- Corrección de errores tipográficos o formatos incorrectos en bases de datos.

Segmentación y Clasificación de Texto

- Extracción de información clave en documentos extensos mediante patrones predefinidos.
- Creación de etiquetas para análisis semántico basado en patrones específicos.
- Identificación automática de partes del discurso (verbos, sustantivos, adjetivos, etc.).

Limpieza y Enriquecimiento de Datos

- Eliminación de elementos innecesarios como HTML, etiquetas de código y caracteres especiales.
- Conversión de texto sin formato en estructuras más organizadas para análisis posterior.
- Creación de nuevas variables a partir de datos textuales mediante patrones específicos.

Filtrado de Datos de Manera Eficiente

Detección y Eliminación de Datos No Válidos

- **Identificación de registros con errores en bases de datos empresariales.**
- **Detección y eliminación de contenido duplicado en grandes conjuntos de datos.**
- **Validación de la estructura de información antes de su uso en modelos predictivos.**

Análisis de Opiniones y Sentimientos

- **Extracción de frases clave en reseñas y comentarios para entender el sentimiento del usuario.**
- **Identificación de términos positivos y negativos en análisis de feedback.**
- **Creación de modelos de clasificación de sentimientos basados en patrones textuales.**

Moderación y Seguridad de Contenidos

- **Detección automática de lenguaje ofensivo o contenido inapropiado en plataformas digitales.**
- **Filtrado de información confidencial o sensible antes de su publicación.**
- **Implementación de sistemas de monitoreo para evitar el fraude y el spam.**

Principales Comandos de Regex en Ciencia de Datos

Las expresiones regulares utilizan una serie de comandos y patrones que permiten realizar búsquedas, sustituciones y segmentaciones en texto de manera eficiente. En ciencia de datos, estos comandos son fundamentales para manipular grandes volúmenes de información y mejorar la calidad de los datos de entrada en modelos predictivos.

Comando Regex	Descripción
`\d`	Encuentra un dígito numérico (0-9).
`\D`	Encuentra cualquier carácter que NO sea un dígito.
`\w`	Encuentra letras, números y guiones bajos.
`\W`	Encuentra cualquier carácter que NO sea una letra, número o guion bajo.
`\s`	Encuentra espacios en blanco, incluyendo tabulaciones y saltos de línea.
`\S`	Encuentra cualquier carácter que NO sea un espacio en blanco.
`.`	Coincide con cualquier carácter excepto saltos de línea.
`^`	Coincide con el inicio de una línea.
`$`	Coincide con el final de una línea.
`*`	Coincide con 0 o más repeticiones del carácter previo.
`+`	Coincide con 1 o más repeticiones del carácter previo.
`?`	Coincide con 0 o 1 ocurrencia del carácter previo.
`{n,m}`	Coincide entre `n` y `m` repeticiones del carácter previo.
` | `	` | ` Operador OR, busca una opción u otra.
`()`	Agrupa expresiones regulares para aplicar reglas específicas.

En ciencia de datos, estos comandos son empleados para tareas como:

Extracción de patrones complejos en documentos y bases de datos.
Limpieza y filtrado de información en flujos de datos en tiempo real.
Transformación y enriquecimiento de texto para análisis de contenido estructurado.

Las **expresiones regulares (regex)** son una herramienta poderosa para la **ciencia de datos, NLP y Machine**

Learning. Su capacidad para manipular, limpiar y extraer información clave hace que sean esenciales en la creación de modelos predictivos más precisos.

Al integrar regex en procesos de preprocesamiento de datos, filtrado de información y análisis de texto, es posible mejorar significativamente la calidad y eficiencia de los modelos de Machine Learning y Deep Learning, permitiendo mejores resultados en la toma de decisiones basada en datos.

Ejercicio 1: Estandarizar nombres de empresas en un DataFrame

Enunciado: En un conjunto de datos de empresas, los nombres pueden estar escritos de distintas formas. Debemos limpiar y estandarizar los nombres eliminando caracteres especiales, espacios extra y términos como "S.A.", "Ltd.", "Inc." o "Corp.".

Datos de entrada:

In [1]:

```
import pandas as pd

df = pd.DataFrame({'empresa': [
    'Google, Inc.',
    'Amazon Ltd.',
    'Facebook  S.A.',
    'Apple   Corp',
    'Tesla Inc'
]})
```

Objetivo: Transformar los nombres en un formato estándar sin caracteres especiales ni términos innecesarios.

Salida esperada:

empresa	empresa_limpia
Google, Inc.	google
Amazon Ltd.	amazon
Facebook S.A.	facebook
Apple Corp	apple

Solución:

In [2]:

```
import re

# Función para limpiar nombres de empresas
def limpiar_nombre_empresa(nombre):
    # 1. Convertir a minúsculas
    nombre = nombre.lower()
    # 2. Remover términos como 'inc.', 'ltd.', 's.a.', 'corp.'
    nombre = re.sub(r'\b(inc|ltd|s\.a\.|corp)\b', '', nombre)
    # 3. Eliminar caracteres especiales (excepto espacios y letras)
    nombre = re.sub(r'[^a-z\s]', '', nombre)
    # 4. Eliminar espacios extra
    nombre = re.sub(r'\s+', ' ', nombre).strip()
    return nombre

df['empresa_limpia'] = df['empresa'].apply(limpiar_nombre_empresa)

df
```

Out[2]:

	empresa	empresa_limpia
0	Google, Inc.	google
1	Amazon Ltd.	amazon

	empresa	empresa_limpia
2	Facebook S.A.	facebook sa
3	Apple Corp	apple
4	Tesla Inc	tesla

Ejercicio 2: Extraer información estructurada de descripciones de productos

Enunciado: Tienes un dataset con descripciones de productos que contienen información como modelo, tamaño y color en texto libre. Tu tarea es extraer estos valores en columnas separadas.

Datos de entrada:

In [3]:

```
df = pd.DataFrame({'descripcion': [
    'Laptop Dell Inspiron 15.6" color negro',
    'Smartphone Samsung Galaxy S21 6.2" azul',
    'Tablet Apple iPad Pro 12.9" plata'
]})
```

Objetivo: Extraer modelo, tamaño (en pulgadas) y color.

Salida esperada:

descripcion	modelo	tamaño_pulgadas	color
Laptop Dell Inspiron 15.6" color negro	Dell Inspiron	15.6"	negro
Smartphone Samsung Galaxy S21 6.2" azul	Samsung Galaxy S21	6.2"	azul
Tablet Apple iPad Pro 12.9" plata	Apple iPad Pro	12.9"	plata

Solución:

In [4]:

```
def extraer_info_producto(texto):
    modelo = re.search(r'\b(Dell|Samsung|Apple)\s+\w+', texto)  # Extraer marca y modelo
    tamaño = re.search(r'\d{1,2}\.\d{1,2}"', texto)  # Buscar tamaño en pulgadas
    color = re.search(r'\b(negro|azul|plata|rojo|verde)\b', texto)  # Buscar color

    return {
        'modelo': modelo.group() if modelo else None,
        'tamaño_pulgadas': tamaño.group() if tamaño else None,
        'color': color.group() if color else None
    }

df_info = df['descripcion'].apply(extraer_info_producto).apply(pd.Series)
df = pd.concat([df, df_info], axis=1)

df
```

Out[4]:

	descripcion	modelo	tamaño_pulgadas	color
0	Laptop Dell Inspiron 15.6" color negro	Dell Inspiron	15.6"	negro
1	Smartphone Samsung Galaxy S21 6.2" azul	Samsung Galaxy	6.2"	azul
2	Tablet Apple iPad Pro 12.9" plata	Apple iPad	12.9"	plata

Ejercicio 3: Identificar patrones de fraude en transacciones financieras

Enunciado: En un conjunto de datos de transacciones, identifica patrones sospechosos, como:

- **Múltiples transacciones en la misma cuenta con valores inusuales (9999.99, 7777.77).**
- **Códigos de transacción con valores fuera de norma (deben ser TX- seguido de 6 números).**

Datos de entrada:

In [5]:

```
df = pd.DataFrame({'cuenta': ['12345', '67890', '12345', '67890', '11111'],
                   'monto': ['100.50', '9999.99', '7777.77', '50.00', '7777.77'],
                   'codigo_transaccion': ['TX-123456', 'TX-ABCDE', 'TX-777777', 'TX-0000
00', 'ABC-123456']})
```

Salida esperada:

cuenta	monto	codigo_transaccion	monto_sospechoso	codigo_valido
67890	9999.99	TX-ABCDE	True	False
12345	7777.77	TX-777777	True	True
11111	7777.77	ABC-123456	True	False

Solución:

In [6]:

```
# 1. Detectar montos sospechosos
df['monto_sospechoso'] = df['monto'].apply(lambda x: bool(re.match(r'^(9999\.99|7777\.77
)$', x)))

# 2. Validar códigos de transacción
df['codigo_valido'] = df['codigo_transaccion'].apply(lambda x: bool(re.match(r'^TX-\d{6}
$', x)))

# 3. Filtrar transacciones sospechosas
df_sospechosas = df[(df['monto_sospechoso'] == True) | (df['codigo_valido'] == False)]

df_sospechosas
```

Out[6]:

	cuenta	monto	codigo_transaccion	monto_sospechoso	codigo_valido
1	67890	9999.99	TX-ABCDE	True	False
2	12345	7777.77	TX-777777	True	True
4	11111	7777.77	ABC-123456	True	False

Ejercicio 4: Limpiar textos de reseñas de clientes

Enunciado: Tienes una columna con reseñas de clientes que contiene:

- **Emojis**
- **Repetición de caracteres (ejemplo: buenoooo → bueno)**
- **Puntuación innecesaria**

Debes limpiar estos textos para su análisis.

Datos de entrada:

In [7]:

```
df = pd.DataFrame({'reseñas': [
    "¡Excelennnteeee! □ Muy buen servicio...",
    "Mmmmmm... No me gustó 😕",
    "Perfectooooo!!!! Lo recomiendo!!! 😊😊"
]})
```

Salida esperada:

reseñas	reseña_limpia
¡Excelennnteeee! Muy buen servicio...	Excelente Muy buen servicio
Mmmmmm... No me gustó 😕	M No me gustó
Perfectooooo!!!! Lo recomiendo!!! 😊😊	Perfecto Lo recomiendo

Solución:

In [8]:

```
# Función para limpiar reseñas
def limpiar_reseña(texto):
    texto = re.sub(r'[\U0001F600-\U0001F64F]', '', texto)  # Quitar emojis
    texto = re.sub(r'(\w)\1{2,}', r'\1', texto)  # Reemplazar letras repetidas
    texto = re.sub(r'[^\w\s]', '', texto)  # Eliminar puntuación innecesaria
    return texto.strip()

df['reseña_limpia'] = df['reseñas'].apply(limpiar_reseña)

df
```

Out[8]:

	reseñas	reseña_limpia
0	¡Excelennnteeee! Muy buen servicio...	Excelente Muy buen servicio
1	Mmmmmm... No me gustó 😕	Mm No me gustó
2	Perfectooooo!!!! Lo recomiendo!!! 😊😊	Perfecto Lo recomiendo

Ejercicio 5: Identificar menciones a marcas en un conjunto de reseñas

Enunciado: Tienes un conjunto de reseñas de productos y debes identificar las menciones a marcas específicas en las reseñas (como 'Apple', 'Samsung', 'Sony', etc.). Si la marca está presente, debes marcarla como "mencionada". Si no está, debe aparecer como "no mencionada".

Datos de entrada:

In [9]:

```
df = pd.DataFrame({'reseña': [
    'Me encanta el nuevo iPhone de Apple, es increíble!',
    'Samsung hace unos televisores muy buenos.',
    'Sony es el líder en cámaras digitales.',
    'Aún no tengo una idea clara sobre los productos de LG.',
    'No estoy seguro si comprar un Xiaomi.'
]})
```

Objetivo:

Identificar las menciones a las marcas en las reseñas y marcar si están presentes o no.

Salida esperada:

reseña	marca_mencionada
Me encanta el nuevo iPhone de Apple, es increíble!	mencionada
Samsung hace unos televisores muy buenos.	mencionada
Sony es el líder en cámaras digitales.	no mencionada
Aún no tengo una idea clara sobre los productos de	. .

reseña	marca_mencionada
No estoy seguro si comprar un Xiaomi.	mencionada

Solución:

In [10]:

```
import re

marcas = ['Apple', 'Samsung', 'LG', 'Xiaomi']

def marca_mencionada(reseña):
    for marca in marcas:
        if re.search(r'\b' + re.escape(marca) + r'\b', reseña, re.IGNORECASE):
            return 'mencionada'
    return 'no mencionada'

df['marca_mencionada'] = df['reseña'].apply(marca_mencionada)

df
```

Out[10]:

	reseña	marca_mencionada
0	Me encanta el nuevo iPhone de Apple, es increí...	mencionada
1	Samsung hace unos televisores muy buenos.	mencionada
2	Sony es el líder en cámaras digitales.	no mencionada
3	Aún no tengo una idea clara sobre los producto...	mencionada
4	No estoy seguro si comprar un Xiaomi.	mencionada

Ejercicio 6: Tokenización de palabras clave en opiniones de productos

Enunciado: Tienes una columna con opiniones de clientes sobre productos. Debes extraer todas las palabras clave relevantes (sustantivos y adjetivos) eliminando conectores y signos de puntuación.

Datos de entrada:

In [11]:

```
df = pd.DataFrame({'opiniones': [
    "El teléfono es increíblemente rápido y tiene una gran cámara!",
    "La batería dura poco, pero el diseño es elegante.",
    "No me gustó la pantalla, pero el sonido es excelente."
]})
```

Objetivo: Extraer solo sustantivos y adjetivos como palabras clave.

Salida esperada:

opiniones	palabras_clave
El teléfono es increíblemente rápido...	['rápido', 'gran', 'cámara']
La batería dura poco, pero el diseño...	['batería', 'diseño', 'elegante']
No me gustó la pantalla, pero el sonido...	['pantalla', 'sonido', 'excelente']

Solución:

In [12]:

```
import re
```

```
def extraer_palabras_clave(texto):
    # Eliminar puntuación y convertir a minúsculas
    texto = re.sub(r'[^\w\s]', '', texto.lower())
    # Extraer palabras clave (sustantivos y adjetivos)
    palabras = re.findall(r'\b(rápido|gran|batería|diseño|pantalla|sonido|excelente|elega
nte|cámara|increíblemente)\b', texto)
    return palabras

df['palabras_clave'] = df['opiniones'].apply(extraer_palabras_clave)

df
```

Out[12]:

	opiniones	palabras_clave
0	**El teléfono es increíblemente rápido y tiene u...**	**[increíblemente, rápido, gran, cámara]**
1	**La batería dura poco, pero el diseño es elegante.**	**[batería, diseño, elegante]**
2	**No me gustó la pantalla, pero el sonido es exc...**	**[pantalla, sonido, excelente]**

Ejercicio 7: Extracción de hashtags para análisis de sentimientos

Enunciado: Tienes un conjunto de tuits que incluyen hashtags (ej. #feliz, #triste). El objetivo es extraer todos los hashtags presentes en cada tuit.

Datos de entrada:

In [13]:

```
df = pd.DataFrame({'tuit': [
    '¡Me siento tan feliz hoy! #feliz #alegría',
    'Qué día tan horrible... #triste #maldía',
    'Nuevo producto lanzado! #innovación #tecnología',
    '¡Me encanta este lugar! #vacaciones #relax',
    'Odio esperar tanto... #desesperación'
]})
```

Objetivo: Extraer todos los hashtags presentes en cada tuit.

Salida esperada:

tuit	hashtags
¡Me siento tan feliz hoy! #feliz #alegría	**['#feliz', '#alegría']**
Qué día tan horrible... #triste #maldía	**['#triste', '#maldía']**
Nuevo producto lanzado! #innovación #tecnología	**['#innovación', '#tecnología']**
¡Me encanta este lugar! #vacaciones #relax	**['#vacaciones', '#relax']**
Odio esperar tanto... #desesperación	**['#desesperación']**

Solución:

In [14]:

```
def extraer_hashtags(tuit):
    hashtags = re.findall(r'#\w+', tuit)
    return hashtags

df['hashtags'] = df['tuit'].apply(extraer_hashtags)

df
```

Out[14]:

	tuit	hashtags

	tuit	hashtags
0	¡Me siento tan feliz hoy! #feliz #alegría	[#feliz, #alegría]
1	Qué día tan horrible... #triste #maldía	[#triste, #maldía]
2	Nuevo producto lanzado! #innovación #tecnología	[#innovación, #tecnología]
3	¡Me encanta este lugar! #vacaciones #relax	[#vacaciones, #relax]
4	Odio esperar tanto... #desesperación	[#desesperación]

Ejercicio 8: Analizar sentimiento basado en palabras clave

Enunciado: Tienes un conjunto de tuits y deseas clasificarlos como "positivos" o "negativos" basados en la presencia de ciertas palabras clave (por ejemplo, "bueno", "feliz" para positivo y "malo", "triste" para negativo).

Datos de entrada:

In [15]:

```
df = pd.DataFrame({'tuit': [
    'Me siento tan feliz hoy, el día ha sido increíble.',
    'Este producto es muy malo, no lo recomiendo.',
    'Excelente experiencia de compra, lo volveré a hacer.',
    'El servicio fue lento y la comida estaba fría.',
    'Muy feliz con el resultado final, totalmente recomendable.'
]})
```

Objetivo: Clasificar los tuits como "positivo" o "negativo" según las palabras clave.

Salida esperada:

tuit	sentimiento
Me siento tan feliz hoy, el día ha sido increíble.	positivo
Este producto es muy malo, no lo recomiendo.	negativo
Excelente experiencia de compra, lo volveré a hacer.	positivo
El servicio fue lento y la comida estaba fría.	negativo
Muy feliz con el resultado final, totalmente recomendable.	positivo

Solución:

In [16]:

```
palabras_positivas = ['feliz', 'bueno', 'excelente', 'increíble', 'recomiendo', 'felicid
ad']
palabras_negativas = ['malo', 'triste', 'lento', 'frío', 'pésimo', 'no recomiendo']

def clasificar_sentimiento(tuit):
    tuit = tuit.lower()
    if any(palabra in tuit for palabra in palabras_positivas):
        return 'positivo'
    elif any(palabra in tuit for palabra in palabras_negativas):
        return 'negativo'
    else:
        return 'neutral'

df['sentimiento'] = df['tuit'].apply(clasificar_sentimiento)

df
```

Out[16]:

	tuit	sentimiento
0	Me siento tan feliz hoy, el día ha sido increí...	positivo
1	Este producto es muy malo, no lo recomiendo.	positivo

	tuit	sentimiento
2	Excelente experiencia de compra, lo volveré a ...	positivo
3	El servicio fue lento y la comida estaba fría.	negativo
4	Muy feliz con el resultado final, totalmente r...	positivo

Ejercicio 9: Filtrar tuits con menciones a otros usuarios

Enunciado: Tienes un conjunto de tuits y deseas filtrar aquellos que contienen menciones a otros usuarios, es decir, aquellos que contienen el símbolo "@" seguido del nombre de usuario.

Datos de entrada:

In [17]:

```
df = pd.DataFrame({'tuit': [
    '@juanito ¡Nos vemos mañana!',
    'Estoy tan emocionado por este evento!',
    '@maría_22 ¿Cómo estás?',
    'No puedo esperar para probar este producto.',
    '@pedro123 ¿Te gustaría acompañarme?'
]})
```

Objetivo: Filtrar los tuits que mencionan a otros usuarios.

Salida esperada:

tuit
@juanito ¡Nos vemos mañana!
@maría_22 ¿Cómo estás?
@pedro123 ¿Te gustaría acompañarme?

Solución:

In [18]:

```
def filtrar_menciones(tuit):
    if re.search(r'@\w+', tuit):
        return True
    return False

df_menciones = df[df['tuit'].apply(filtrar_menciones)]

df_menciones
```

Out[18]:

	tuit
0	@juanito ¡Nos vemos mañana!
2	@maría_22 ¿Cómo estás?
4	@pedro123 ¿Te gustaría acompañarme?

Ejercicio 10: Identificar lenguaje ofensivo en comentarios

Enunciado: En un sistema de moderación, debemos detectar palabras ofensivas en comentarios y marcarlos como inapropiados.

Datos de entrada:

In [19]:

```
df = pd.DataFrame({'comentarios': [
    "Este producto es una porquería, no lo compren!",
```

```
    "Me gustó mucho, excelente servicio.",
    "Vaya basura de producto, un fraude total."
]})
```

Objetivo: Identificar cometarios con palabras ofensivas.

Salida esperada:

comentarios	es_ofensivo
Este producto es una porquería...	True
Me gustó mucho, excelente servicio.	False
Vaya basura de producto, un fraude total.	True

Solución:

In [20]:

```
def detectar_ofensivo(texto):
    return bool(re.search(r'\b(porquería|basura|fraude)\b', texto.lower()))

df['es_ofensivo'] = df['comentarios'].apply(detectar_ofensivo)

df
```

Out[20]:

	comentarios	es_ofensivo
0	Este producto es una porquería, no lo compren!	True
1	Me gustó mucho, excelente servicio.	False
2	Vaya basura de producto, un fraude total.	True