Práctica 2.1: Codificación de Texto en Vectores

UNR - TUIA - Procesamiento de Lenguaje Natural

PARTE 1: ONE-HOT ENCODING

Ejercicio 1: Implementación Manual de One-Hot Encoding

```
corpus = [
    "el perro ladra fuerte",
    "el gato maulla suave",
    "el perro y el gato juegan",
    "el pájaro canta bonito",
    "el caballo corre rápido",
    "la vaca muge despacio",
    "el pez nada tranquilo",
    "la abeja zumba cerca",
    "el ratón corre pequeño",
    "el gallo canta temprano",
    "el lobo aúlla de noche",
    "la rana salta alto",
    "el león ruge fuerte",
    "la oveja bala bajito",
    "el pato nada contento",
    "el burro rebuzna fuerte",
    "el tigre acecha sigiloso",
    "la tortuga camina lento",
    "el mono grita alegre",
    "el perro duerme tranquilo"
]

def crear_one_hot_manual(corpus):
    """
    TODO: Implementar One-Hot Encoding sin usar sklearn
    1. Crear vocabulario único
    2. Asignar índice a cada palabra
    3. Crear matriz one-hot para cada documento
    4. Retornar matriz y vocabulario
    """
```

```
    # seguir
    pass

# Visualización requerida:
# - Imprimir vocabulario con índices
# - Mostrar matriz completa
# - Crear heatmap con seaborn
```

Ejercicio 2: Comparación One-Hot vs sklearn

```
documentos = [
    "inteligencia artificial revoluciona la tecnología",
    "aprendizaje automático es parte de inteligencia artificial",
    "redes neuronales profundas dominan el aprendizaje",
    "el futuro es inteligencia artificial y robótica",
    "los algoritmos de machine learning analizan grandes datos",
    "procesamiento de lenguaje natural entiende textos humanos",
    "visión por computadora detecta objetos en imágenes",
    "la robótica combina hardware y algoritmos inteligentes",
    "modelos generativos crean contenido nuevo",
    "el big data alimenta a la inteligencia artificial",
    "la automatización transforma la industria moderna",
    "la ética en inteligencia artificial es fundamental",
    "los sistemas expertos ayudan a tomar decisiones",
    "el aprendizaje profundo impulsa los autos autónomos",
    "las redes convolucionales procesan imágenes",
    "las redes recurrentes analizan secuencias temporales",
    "la computación en la nube acelera la inteligencia artificial",
    "el internet de las cosas conecta dispositivos inteligentes",
    "la inteligencia artificial mejora la medicina",
    "los asistentes virtuales usan procesamiento de lenguaje natural"
]

def comparar_implementaciones(documentos):
    """
    TODO: Comparar tu implementación vs sklearn
    1. Implementar con tu función manual
    2. Implementar con OneHotEncoder de sklearn
    3. Verificar que las matrices sean iguales
    4. Medir tiempos
    """
    # Escribir
    pass

# Generar reporte con:
```

```
# - Tabla comparativa de tiempos
```

PARTE 2: COUNT VECTORIZER

Ejercicio 3: Count Vectorizer desde Cero

```
corpus_noticias = [
    "el gobierno anuncia nuevas medidas económicas para combatir
inflación",
    "inflación alcanza niveles históricos según gobierno nacional",
    "medidas económicas insuficientes critican economistas expertos",
    "banco central interviene ante crisis económica nacional",
    "economistas advierten sobre riesgos de nuevas medidas gobierno",
    "el presidente promete reformas fiscales para estabilizar la
economía",
    "el congreso debate el presupuesto nacional para el próximo año",
    "los sindicatos convocan a huelga general contra ajuste económico",
    "el desempleo aumenta en sectores industriales clave",
    "la oposición critica la política monetaria del gobierno",
    "los mercados financieros reaccionan con incertidumbre",
    "el dólar sube tras anuncios del banco central",
    "los precios de alimentos y combustibles presionan la inflación",
    "inversores extranjeros muestran cautela ante crisis política",
    "la deuda externa preocupa a analistas internacionales",
    "el ministro de economía defiende plan de ajuste",
    "organismos internacionales recomiendan disciplina fiscal",
    "las exportaciones crecen a pesar de la recesión interna",
    "el sector privado pide incentivos para invertir",
    "analistas prevén recuperación lenta de la economía nacional"
]

def count_vectorizer_manual(corpus):
    """
    TODO: Implementar Count Vectorizer manualmente
    1. Tokenizar cada documento
    2. Crear vocabulario
    3. Contar frecuencias por documento
    4. Construir matriz de conteo
    """
    #Escribir aca
    pass

# Análisis adicional:
# - Palabras más frecuentes globalmente
```

```
# - Palabras únicas por documento
# - Distribución de frecuencias (histograma)
```

Ejercicio 4: Count Vectorizer con Preprocesamiento

```
textos_sucios = [
    "WOW!!! 😱 El concierto estuvo INCREÍBLEEEE 🔥🔥🔥 #MusicLovers",
    "RT @fan123: Messi GOAT 🐐⚽ https://futbol.com",
    "jajajaja ese video no puede ser real xD 😂😂",
    "ayer dormí 2hs nada +, toy destruidooo 😴☕",
    "me encantaaa el nuevo album de taylor swift 💖💖 #Swiftie",
    "comida riquísimaaaa!!! 🍕🍔🍟 quiero repetir YA!!!",
    "RT @tech_guru: nuevo iPhone 16 🔋⚡ specs brutales
https://apple.com",
    "hoy gym 💪💪 #fitnesslife pero toy sin ganas jajaja",
    "wtf?? la app se crasheó d nuevooo 🤯🤯",
    "mañana examen de mates... auxiliooo 😭📚",
    "NOOOOO me olvidé la contraseña otra vezsss 🤦‍♂️🔑",
    "jajaja mi perro corrió detrás d su cola x 10 min 🐶😂",
    "q onda con la lluviaaaa ☔☔ no para mas!",
    "el café d la facu ta MALÍSIMO 🤢🤮",
    "SIIIIII 😍😍 aprobé la materiaaa con 10!!!",
    "el cine estaba LLENÍSIMO ayerrr 🍿🎬",
    "toy escuchando el mismo tema en loop x1000 🎧🎶",
    "LOL esa serie se puso BUENÍSIMA 🔥🔥 #Netflix",
    "me cancelaron el pedido d pizza 😡🍕",
    "ajajajaja no paro d reirme con ese meme 🤣🤣🤣",
    "OMG la playa hoy estaba PERFECTA 🌊🌴 #Vacaciones",
    "RT @gamer45: nuevo parche en Fortnite 🚀🎮 https://epicgames.com",
    "ugh mi celu no carga másss 😤🔋",
    "ayer salí a correr y me caí JAJAJA 🏃‍♂️💥😂",
    "kiero dormir 15 horas seguidas plzzz 🛌😴",
    "la serie finalizó y toy vacíoooo 😭😭",
    "ufff q calor insoportableee ☀️🔥",
    "RT @cinefan: trailer de Avengers 8 😱🎬 https://marvel.com",
    "jajaj toy viciadísimo al nuevo jueguito 🎮 #gaminglife",
    "la pizza 🍕🍕 llegó FRÍA mal... q broncaaa",
    "awww mi gato me despertó con ronroneos 🐱❤️",
    "la app d banco se cayó otra vezzzz 🤦‍♀️",
    "diosss q temazo el d anoche 🎵🎶",
]

def pipeline_count_vectorizer(textos):
    """
    TODO: Pipeline completo con Count Vectorizer
```

```
    1. Limpieza de texto (puntuación, URLs, mentions)
    2. Normalización (minúsculas, números)
    3. Tokenización
    4. Eliminación de stopwords
    5. Count Vectorizer
    6. Análisis de resultados
    """
    # Escribir aca
    pass

# Visualizaciones requeridas:
# - Nube de palabras antes y después de limpieza
# - Matriz de frecuencias como heatmap
# - Top 10 palabras
```

PARTE 3: TF-IDF

Ejercicio 5: Implementación Manual de TF-IDF

```
documentos = [
    "python es un lenguaje de programación versátil",
    "python se usa mucho en ciencia de datos",
    "java es otro lenguaje de programación popular",
    "ciencia de datos requiere python y estadística",
    "programación en python para análisis de datos",
    "javascript domina el desarrollo web moderno",
    "c++ se utiliza en sistemas de alto rendimiento",
    "r es muy usado en estadística y visualización",
    "sql es esencial para manejar bases de datos",
    "machine learning combina programación y matemáticas",
    "python facilita el desarrollo de inteligencia artificial",
    "java es común en aplicaciones empresariales",
    "go es eficiente para servicios en la nube",
    "scala se usa en procesamiento de datos masivos",
    "matlab es popular en ingeniería y simulaciones"
]

def calcular_tfidf_manual(documentos):
    """
    TODO: Calcular TF-IDF paso a paso
    1. Calcular TF para cada término en cada documento
    2. Calcular IDF para cada término
    3. Multiplicar TF * IDF
```

```
    4. Aplicar normalización L2
    """
    # Seguir
    # Mostrar cálculos intermedios para verificación
    pass

# Mostrar:
# - Matriz TF
# - Vector IDF
# - Matriz TF-IDF sin normalizar
# - Matriz TF-IDF normalizada
# - Comparación con sklearn
```

Ejercicio 6: TF-IDF para Búsqueda de Documentos

```
base_conocimiento = [
    "Python es ideal para principiantes en programación",
    "Java se usa en aplicaciones empresariales grandes",
    "JavaScript domina el desarrollo web frontend",
    "Python excel en machine learning y data science",
    "C++ ofrece máximo rendimiento en sistemas",
    "Ruby on Rails acelera desarrollo web",
    "Go es perfecto para microservicios modernos",
    "Rust garantiza seguridad de memoria",
    "PHP sigue siendo popular para web backends",
    "Swift es el lenguaje para desarrollo iOS"
]

queries = [
    "lenguaje para data science",
    "desarrollo web rápido",
    "programación para novatos",
    "máximo performance"
]

def motor_busqueda_tfidf(base, queries):
    """
    TODO: Implementar motor de búsqueda con TF-IDF
    1. Vectorizar base de conocimiento
    2. Vectorizar queries
    3. Calcular similitud coseno
    4. Ranking de resultados
    5. Mostrar top-3 para cada query
    """
```

```
    # Seguir
    pass

# Visualización:
# - Matriz de similitud query-documento
# - Gráfico de relevancia por query
# - Heatmap de términos importantes
```

PARTE 4: HASH VECTORIZER

Ejercicio 7: Implementación de Hash Vectorizer

```
import hashlib

vocabulario_grande = ["palabra" + str(i) for i in range(10000)]

def hash_vectorizer_manual(vocabulario, n_features=100):
    """
    TODO: Implementar Hash Vectorizer
    1. Crear función hash (usar hashlib)
    2. Mapear palabras a índices usando hash
    3. Manejar colisiones
    4. Crear vectores dispersos
    """
    # Escribir
    pass

# Análisis de colisiones:
# - Contar colisiones con diferentes n_features
# - Graficar tasa de colisión vs n_features
# - Identificar palabras que colisionan
# - Comparar diferentes funciones hash
```

Ejercicio 8: Comparación de Eficiencia

```
tamaños_corpus = [100, 500, 1000, 5000, 10000]

def benchmark_vectorizadores(tamaños):
    """
    TODO: Comparar todos los métodos
    1. Generar corpus de diferentes tamaños
    2. Medir tiempo de vectorización
    """
```

```
    resultados = {
        'one_hot': {},
        'count': {},
        'tfidf': {},
        'hashing': {}
    }
    # Escribir aca
    pass

# Generar gráfico:
# - Tiempo vs tamaño del corpus
```