Análisis de Medidas Disimilitud y Similitud

Practica 1.3

Alumnos: Karla Rivera Lima, Natali Meza Barranco, Didier García Toquiantzi, Juan Diego Espinosa Sandoval.

1. Introducción

Es útil examinar las similitudes o distancias para tratar de localizar agrupaciones de películas que son similares. Al estudiar los patrones de las películas, también podemos obtener una idea de qué pasa con las relaciones que existen entre un titulo y otro, para ello obtener sugerencias de que películas ver con tema en común.

Metodología

En esta práctica utilizaremos las medidas de coseno, Jaccard, para determinar la similitud que existe en las películas.

En la cual consiste en un enfoque para responder problemas mediante medidas de similitud. A continuación, se describen las actividades realizadas en las que se analizaron las muestras y exceptuando las fases de desarrollo para obtener la implementación de dicha práctica.

2. Desarrollo

2.1 Creación de las funciones que calculen la distancia de coseno y Jaccard utilizando lenguaje de programación Python con ayuda de Colab.

```python
# Función para calcular la distancia de coseno entre dos vectores
def coseno_distancia(vector1, vector2):
    dot_product = np.dot(vector1, vector2)
    norm_vector1 = np.linalg.norm(vector1)
    norm_vector2 = np.linalg.norm(vector2)

    cosine_similarity = dot_product / (norm_vector1 * norm_vector2)
    cosine_distance = 1 - cosine_similarity

    return cosine_distance
```

```python
# Función para calcular la distancia Jaccard entre dos conjuntos
def jaccard_distance(set1, set2):
    intersection = len(set1.intersection(set2))
    union = len(set1.union(set2))

    jaccard_similarity = intersection / union
    jaccard_distance = 1 - jaccard_similarity

    return jaccard_distance
```

Implementación de sistema sencillo para recomendar películas, para ello ocupamos el conjunto de datos movie.

2.2 Cargar el conjunto de datos, se utilizó pandas, es una librería de Python especializada en la manipulación y el análisis de datos. Ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales.

```python
df = pd.read_excel("movie.xlsx")

df
```

	Title	Year	Genres	Language	Country	Content Rating	Duration	Aspect Ratio	Budget	Gross Earnings	Director	Actor1	Actor2
0	102 Dalmatians	2000	Adventure Comedy Family	English	USA	G	100.0	1.85	85000000.0	66941559.0	Kevin Lima	Ioan Gruffudd	Eric Idle
1	28 Days	2000	Comedy Drama	English	USA	PG-13	103.0	1.37	43000000.0	37035515.0	Betty Thomas	Steve Buscemi	Viggo Mortensen
2	3 Strikes	2000	Comedy	English	USA	R	82.0	1.85	6000000.0	9821335.0	DJ Pooh	Mo'Nique	Mike Epps
3	Aberdeen	2000	Drama	English	UK	NaN	106.0	1.85	6500000.0	64148.0	Hans Petter Moland	Charlotte Rampling	Sara-Marie Maltha
4	All the Pretty Horses	2000	Drama Romance Western	English	USA	PG-13	220.0	2.35	57000000.0	15527125.0	Billy Bob Thornton	Matt Damon	Henry Thomas

2.3 Para el análisis solo se consideraros las siguientes columnas título, el género, actores y director.

```
# Seleccionar las características relevantes: título, género, director y actores
caracteristica = ['Title', 'Genres', 'Director', 'Actor1','Actor2','Actor3']
data_c = df[caracteristica]

data_c
```

	Title	Genres	Director	Actor1	Actor2
0	102 Dalmatians	Adventure Comedy Family	Kevin Lima	Ioan Gruffudd	Eric Idle
1	28 Days	Comedy Drama	Betty Thomas	Steve Buscemi	Viggo Mortensen
2	3 Strikes	Comedy	DJ Pooh	Mo'Nique	Mike Epps
3	Aberdeen	Drama	Hans Petter Moland	Charlotte Rampling	Sara-Marie Maltha
4	All the Pretty Horses	Drama Romance Western	Billy Bob Thornton	Matt Damon	Henry Thomas

0 s completado a las 14:28

2.4 En este punto se utilizó las funciones para calcular las distancias entre los títulos, para emplear ambas distancias, se tuvo que vectorizar los datos del conjunto para poder calcular las distancias.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

df2 = pd.DataFrame(df)

# Crear una instancia del vectorizador TfidfVectorizer
vectorizer = TfidfVectorizer()

# Utilizar el método fit_transform para vectorizar la columna "Title"
vectorized_titles = vectorizer.fit_transform(df2['Title'].fillna(''))

# El resultado, vectorized_titles, contiene las representaciones vectoriales TF-IDF de los títulos
print(vectorized_titles)
```

```
  (0, 573)      0.7071067811865476
  (0, 2)        0.7071067811865476
  (1, 592)      0.6470979483872036
  (1, 22)       0.7624068764072579
  (2, 2244)     1.0
  (3, 42)       1.0
  (4, 1132)     0.6278231368207947
  (4, 1816)     0.5958312193181347
  (4, 2326)     0.17609328275428607
```

```
data = pd.DataFrame(data_c)

# Lista de columnas a vectorizar
columnas_a_vectorizar = ['Title', 'Genres', 'Director', 'Actor1', 'Actor2', 'Actor3']

# Crear un diccionario para almacenar los vectorizadores y las representaciones vectoriales
vectorizers = {}
vectorized_columns = {}

# Vectorizar cada columna de texto y almacenar las representaciones vectoriales en vectorized_columns
for columna in columnas_a_vectorizar:
    vectorizer = TfidfVectorizer()
    vectorized_data = vectorizer.fit_transform(data_c[columna].fillna(''))  # Llenar valores NaN con cadenas vací
    vectorizers[columna] = vectorizer
    vectorized_columns[columna] = vectorized_data.toarray()

vectorized_columns
```

```
{'Title': array([[0.        , 0.        , 0.70710678, ..., 0.        , 0.        ,
         0.        ],
        [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
         0.        ],
```

2.5 Una vez vectorizado las columnas, se va ingresar un título de la película, se calcula distancias con cada medida de similitud respecto de cada característica y al final comparamos las distancias más pequeñas y muestra la información, que tengan mayor similitud, (Título, genero, director, actores, distancia coseno, distancia Jaccard) de los cinco mejores resultados

```
# Muestra las 5 mejores recomendaciones basadas en similitud coseno
top_coseno_recomendador = cosine_scores[1:6]

# Muestra las 5 mejores recomendaciones basadas en similitud Jaccard
top_jaccard_recomendador = jaccard_scores[1:6]

# muestra las recomendaciones
print("Recomendaciones basadas en similitud coseno:")
for i, score in top_coseno_recomendador:
    print(data.iloc[i]['Title'], data.iloc[i]['Genres'], data.iloc[i]['Actor1'])

print("\nRecomendaciones basadas en similitud Jaccard:")
for i, score in top_jaccard_recomendador:
    print(data.iloc[i]['Title'], data.iloc[i]['Genres'], data.iloc[i]['Actor1'])
```

```
Recomendaciones basadas en similitud coseno:
Black Hawk Down  Drama History War Ioan Gruffudd
Zathura: A Space Adventure  Action Adventure Comedy Family Fantasy Sci-Fi Kristen Stewart
Up  Adventure Animation Comedy Family John Ratzenberger
Hoot  Adventure Comedy Family Logan Lerman
Cars  Adventure Animation Comedy Family Sport John Ratzenberger
```

3.- Conclusión

Las medidas de similitud son herramientas fundamentales en una amplia variedad de campos, desde la informática y la ciencia de Datos. Estas medidas permiten cuantificar la semejanza o la distancia entre objetos, datos o conceptos, lo que resulta crucial en tareas como la búsqueda de información, la agrupación de datos, la recomendación de contenido y mucho más. Identificamos que, con estas medidas de similitud, son importantes para desarrollar algoritmos. Algunos puntos importantes a destacar sobre las medidas de similitud son: Diversidad de aplicaciones, Elección de la medida adecuada, Impacto en la calidad de los resultados, Preprocesamiento de datos, Interpretación de resultados.

Las medidas de similitud continúan evolucionando con el avance de la tecnología y la investigación. Nuevos métodos y enfoques siguen surgiendo para abordar desafíos específicos en diferentes dominios, La elección adecuada de estas medidas y su correcta interpretación son fundamentales para aprovechar al máximo su potencial en la toma de decisiones y la resolución de problemas.