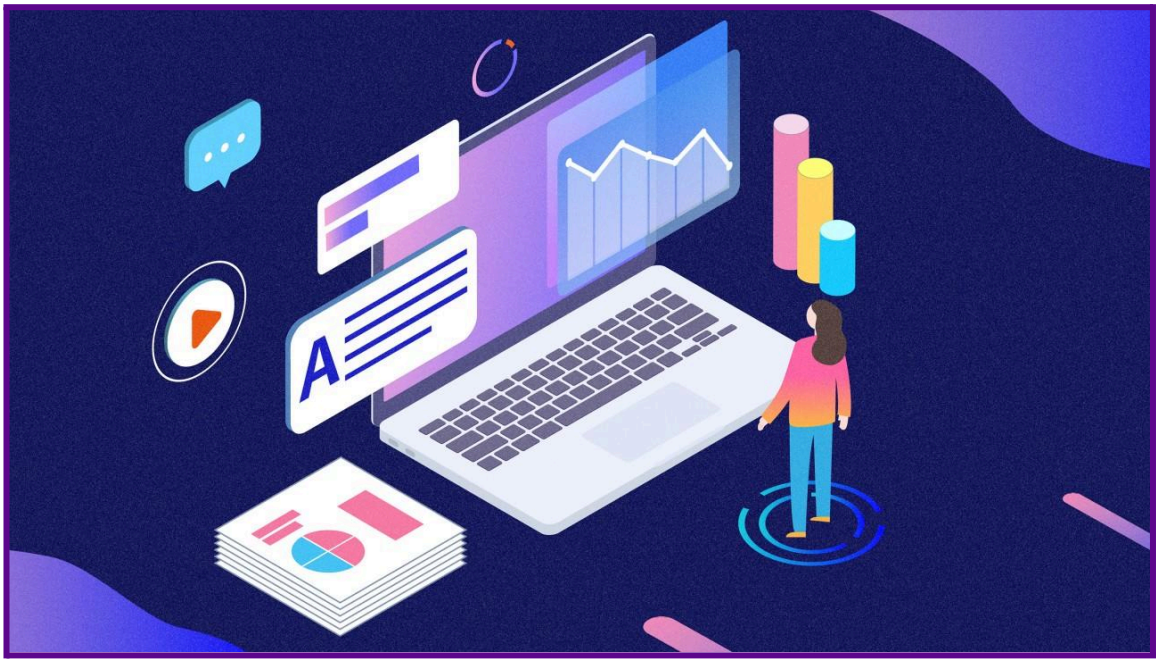
Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

25-26 GCO

Sistemas de recomendación.
Métodos de filtrado colaborativo



C/ Padre Herrera s/n
38207 La Laguna
Santa Cruz de Tenerife. España

T: 900 43 25 26

ull.es

Jean Franco Hernández García - alu0101538853@ull.edu.es
Arun Daswani Lakhani - alu0101560410@ull.edu.es
Javier González Brito - alu0101548197@ull.edu.es

Tabla de contenidos

Introducción

Para la realización de esta práctica se ha llevado a cabo el desarrollo de un sistema de recomendación basado en el filtrado colaborativo, una de las técnicas más utilizadas en la actualidad para la personalización de contenidos y recomendaciones en las plataformas digitales. El objetivo principal pasa por predecir las valoraciones que un usuario podría otorgar a ítems que aún no ha evaluado, utilizando para ello la información contenida en una estructura que se conoce como matriz de utilidad.

En nuestro caso particular, el sistema implementado permite calcular la similitud entre diversos usuarios haciendo uso de diferentes métricas (correlación de Pearson, distancia coseno y distancia euclídea), seleccionando un número determinado de vecinos y generando predicciones en base a dos enfoques: una predicción simple o la diferencia con la media. De estas predicciones, obtenemos la matriz de utilidad completa generando las recomendaciones personalizadas.

Por lo que en esta práctica buscamos comprender el funcionamiento de los métodos de filtrado colaborativo, analizar su aplicación y sacar conclusiones de su uso.

Descripción del método

- Matriz de utilidad:
  - La matriz de utilidad está construida de la siguiente manera: las dos primeras líneas muestran dos números, los cuales indican el rango de valores(máximo y mínimo), luego encontramos la matriz como tal, con sus filas(usuario), columnas(ítems) y valores desconocidos('-').
  - Hay matrices con diferentes tamaños para probar su funcionamiento.
  - ```
    0.000
    5.000
    4.746 - 0.444 4.765 4.995 0.794 4.547 4.521 4.590 -
    2.391 4.377 3.601 0.160 0.481 1.301 4.824 0.176 4.072 2.798
    0.986 4.324 4.099 0.983 3.398 1.233 2.327 2.878 3.851 1.274
    3.611 0.572 1.189 4.045 - 0.136 0.435 3.553 0.887 1.222
    0.110 4.602 2.236 1.531 4.335 1.189 4.877 3.926 0.711 4.295
    ```
- Cálculo de similitud:
  - Para el cálculo de similitud usamos tres métricas distintas:
    - Correlación de Pearson: mide la relación lineal entre usuarios
    - Distancia coseno: mide el ángulo entre los vectores de valoración.
    - Distancia euclídea: mide la proximidad geométrica.
- Selección de vecinos:
  - Para cada usuario se seleccionan los *k* vecinos más similares con similitud positiva.
- Predicción de valoraciones:
  - Predicción simple: se calcula el valor que falta haciendo un promedio de las valoraciones de los vecinos más parecidos, dándole más peso a los que tienen mayor similitud.
  - Predicción con diferencia respecto a la media: tiene en cuenta que cada usuario valora distinto. Se usa la media del usuario y se ajusta según cómo valoran los vecinos ese ítem respecto a sus propias medias.
    ```

Desarrollo e implementación

Nuestro código está condensado todo en un solo archivo el cual tiene funciones diferentes para caso específico

- Tenemos estas diferentes funciones:
  - Función de lectura:
    - cargar_matriz(archivo), esta función se le pasa el parámetro del nombre del archivo y te devuelve los diferentes valores recogidos del txt
  - Funciones de cálculo de similitudes:
    - pearson_correlation(x,y), esta función se encarga de calcular la correlación de pearson para posteriormente usarla en en la función pearson_similarity() y se le pasa la lista de valoraciones del usuario x e y
    - pearson_similarity(matrix), se encarga de calcular pasandole la matriz como parámetros la similitud en base a pearson
    - cosine_similarity(matrix), se encarga de calcular pasandole la matriz como parámetros la similitud en base al coseno
    - euclidean_similarity(matrix), se encarga de calcular pasandole la matriz como parámetros la similitud en base a la distancia euclidea
  - Funciones de predicciones:
    - get_neighbors(similarity, user, k), esta función se le pasa la matriz de similitudes al usuario, el usuario y número de vecinos k y luego devuelve una lista de vecinos más parecidos
    - predict_simple(matrix, similarity, user, item, k) esta función recibe la matriz original, la matriz de similitudes, luego el índice del usuario a predecir y su ítem y el número de vecinos a tener en cuenta que sería k y devuelve la predicción hecha con predicción simple
    - predict_with_mean(matrix, similarity, user, item, k) esta función recibe la matriz original, la matriz de similitudes, luego el índice del usuario a predecir y su ítem y el número de vecinos a tener en cuenta que sería k y devuelve la predicción hecha con predicción sobre la media
  - Funciones restantes:
    - print_matriz(m), función que le pasas la matriz y te la imprime
    - main(), función que se encarga de tomar parametros de linea de comando y ejecutar el programa con las especificaciones seleccionadas
- Ejecución del programa con parámetros:

  python3 main.py utility-matrix-5-10-1.txt pearson 3 simple

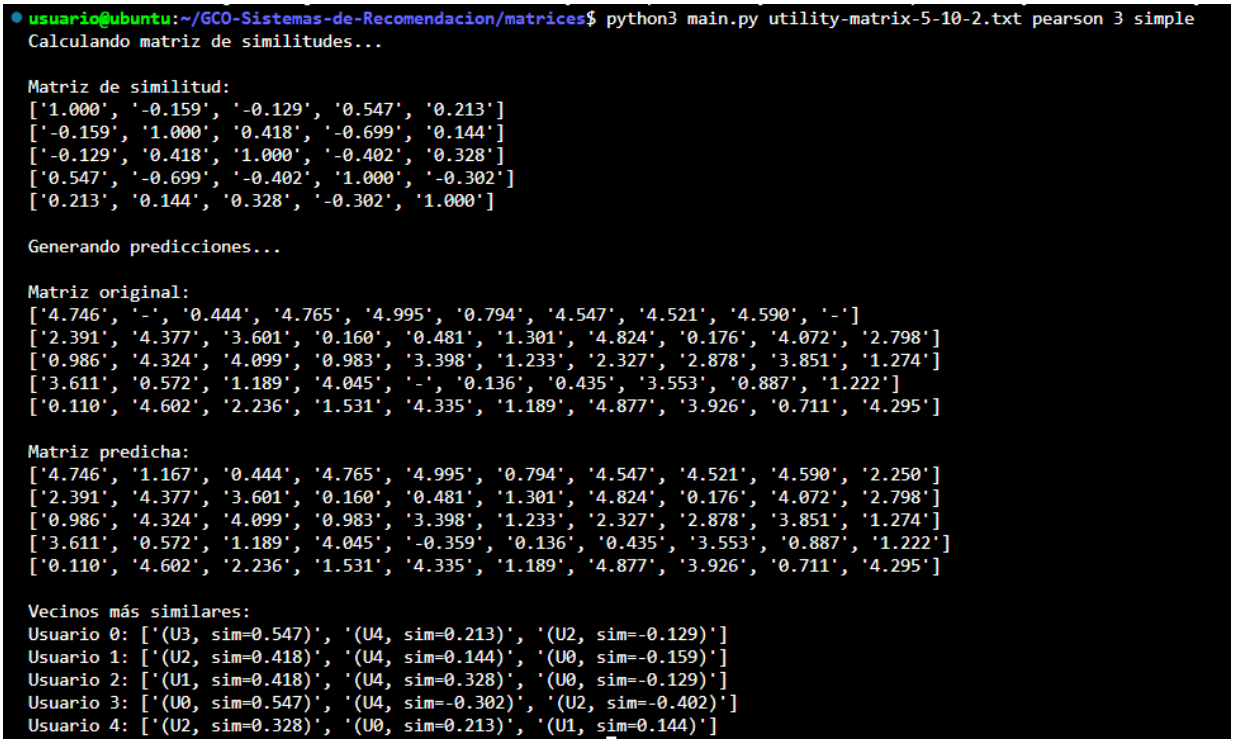
Y los parámetros:

1. Archivo de matriz
2. Métrica (pearson, coseno, euclidea)
3. Número de vecinos (ej. 3)
4. Tipo de predicción (simple o media)

```
if len(sys.argv) < 5:
    print("Uso: python3 main.py <archivo> <metrica> <num_vecinos> <tipo_prediccion>")
    print("Métricas: pearson | coseno | euclidea")
    print("Tipos: simple | media")
```

Resultados

A continuación mostraremos los resultados de la ejecución del programa con las diferentes matrices.



Los demás ejemplos se encuentran en un directo dentro del repositorio, en ficheros .txt que contienen los resultados de diferentes ejecuciones.

Conclusiones

A lo largo del desarrollo de esta práctica hemos aprendido acerca del funcionamiento y la lógica que está detrás del filtrado colaborativo, comprendiendo cómo funcionan y se rigen las relaciones entre usuarios e ítems, permitiendo estimar valoraciones que hasta el momento no han sido observadas. También hemos entendido la importancia de la similitud existente entre usuarios y cómo el número de vecinos dentro de la matriz de utilidad influyen de manera significativa en la calidad de las recomendaciones.

En las ejecuciones realizadas con el software que hemos implementado, la correlación de Pearson nos ofreció mejores resultados. Esto puede deberse a que esta métrica tiene en cuenta las diferencias en las escalas de valoración entre usuarios y es capaz de capturar mejor los patrones comunes.

Dentro de las principales limitaciones que encontramos durante el desarrollo de este ejercicio, podemos destacar en base al filtrado colaborativo su alta dependencia de los datos (problema de dispersión), la dificultad para tratar nuevos usuarios o ítems (cold start) y el alto coste computacional cuando la matriz de utilidad es muy grande.

Como posibles mejoras, sería interesante aplicar una normalización de las valoraciones, considerar vecinos negativos para captar relaciones inversas y explorar un enfoque híbrido que combine filtrado colaborativo con métodos basados en contenido, con el fin de mejorar la precisión y la cobertura del sistema.

Bibliografía / Fuentes

Incluye:

- Enlace al repositorio de ejemplos
- Documentación de NumPy.
- Apuntes de clase o recursos usados para Pearson/coseno.