Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2433 - Minería de Datos

Tarea 1: Reglas de asociación

Profesor Vicente Domínguez
5 de septiembre de 2020

Indicaciones

- Fecha de Entrega: 28 de septiembre a las 23:59.
- Se debe entregar la tarea en el repositorio asignado a cada uno por Github Classroom. El *link* para generar su repositorio es `https://classroom.github.com/a/2JH7KZIl`.
- Cada hora de atraso descuenta 1 punto de la nota que obtengas.
- La tarea es *individual*. La copia será sancionada con una nota 1.1 en la tarea, además de las sanciones disciplinarias correspondientes.
- **No se permite** el uso de librerías diferentes a las mencionadas en este enunciado. Además, si quieres optar al bonus, **no puedes utilizar la librería itertools** en la implementación de tu algoritmo.
- Todo código entregado se comparará con implementaciones del algoritmo en internet para evitar la copia de este.

Introducción

Esta tarea consiste en implementar el algoritmo **FP-Growth**, propuesto por Christian Borgelt en 2005. Este tiene como objetivo encontrar itemsets frecuentes dentro de una base de datos y generar reglas de asociación bajo determinados umbrales de soporte y confianza.

En *data mining*, las reglas de asociación son ampliamente utilizadas para descubrir relaciones entre variables en bases de datos de gran tamaño. Aplicaciones clásicas de este tipo de estrategias pueden ser encontradas en análisis de compras y de características socio-demográficas desde bases de datos censales, entre otras.

Base de Datos

La base de datos a utilizar corresponde a múltiples ratings de películas hechos por usuarios, en una escala del 1 al 5. Este dataset fue extraido de *Movielens*, un servicio y una comunidad dedicados a la opinión y recomendación de contenido cinematográfico.

En específico, se entragarán dos archivos `.csv`:

- **`ratings.csv`** : Contiene, entre otras, las columnas `userId`, `movieId` y `rating`. Cada entrada representa un rating hecho por un usuario a una película, es decir, que el usuario de id `userId` calificó a la película de id `movieId` con una puntuación de `rating`.
- **`movies.csv`** : Contiene el título y género de cada película. Estos datos serán útiles para obtener información de las películas que nos entreguen las reglas de asociación, por lo que se utilizará una vez que ya se hayan obtenido las reglas mediante los ratings. Con esta información, podremos explicar mejor las reglas de asociación obtenidas.

Se espera que en el punto **3.** de las actividades obtengan reglas de asociación de las películas del dataset que obtendrán de `ratings.csv`. Mediante estas reglas, buscamos descubrir patrones de gustos de películas parecidos entre los usuarios. Para hacer esto, deberás agrupar los datos por usuario y aplicar el algoritmo sobre los grupos generados.

Por lo tanto, queremos obtener reglas de asociación que nos permitan inferir películas que le pueden gustar a una persona, dado un conjunto de películas calificadas positivamente por el. Por esto, debemos filtrar el conjunto por rating, dejando solo puntuaciones altas.

Actividades

Las tareas que deberás realizar se detallan a continuación:

1. Preprocesar los datos

Deberás preprocesar los datos presentes en el dataset, con el fin de facilitar su manejo y evitar errores durante la ejecución del algoritmo. Además, debes dejar solo las puntuaciones altas de los ratings. Para esto, deberás implementar las siguientes funciones:

- **`preprocess(df)`**: Recibe un `DataFrame` con los ratings (del archivo `ratings.csv`) y lo preprocesa. Esto incluye las siguientes tareas:
  - Eliminar las columnas innecesarias.
  - Eliminar las filas con valores nulos en alguno de sus datos.
  - Filtrar los ratings con puntuación baja, dejando sólo aquellos con puntuación ≥ 4.

  Retorna el mismo `DataFrame` ya preprocesado.

- **`dataframe_to_ndarray(df)`**: Recibe el `DataFrame` preprocesado y retorna un `ndarray` bajo los siguientes criterios:
  - La posición de cada arreglo dentro del `ndarray` corresponde al identificador del usuario (de la columna `userId`) menos 1. Es decir, al primer usuario le corresponde la posición 0, al segundo la posición 1, al tercero la posición 2, etc.
  - El contenido del arreglo dentro del `ndarray` correspondiente al identificador de un usuario, corresponde a los ids de las películas que le gustaron a este usuario (calificadas con puntuación ≥ 4).

  Para ejemplificar lo pedido, tomemos el siguiente `ndarray`:

  $$[[4, 5, 8], [2, 3], [7, 9, 11]]$$

  Lo anterior significa que:

  - Al usuario 1 le gustaron las películas de `movieId` 4, 5 y 8.
  - Al usuario 2 le gustaron las películas de `movieId` 2 y 3.
  - Al usuario 3 le gustaron las películas de `movieId` 7, 9 y 11.

2. Implementar el algoritmo FP-Growth

Se debe implementar el algoritmo según lo visto en clases, separando el código en la creación del árbol y la mina de este. Para esto, **solo** podrás utilizar las librerías numpy, pandas e itertools.

Para estructurar mejor tu código, tendrás que implementar las clases **`node`** y **`fpgrowth`**. La clase `node` servirá para representar el árbol, el que deberá ser generado recursivamente. Esta debe tener como atributos una lista con sus hijos (de la misma clase), ademas de los atributos necesarios para guardar la información necesaria de cada nodo. Por otro lado, la clase `fpgrowth` servirá para implementar el algoritmo. Esta debe tener como mínimo los siguientes métodos:

- **`create_tree(ndarray_movies, min_support)`**: Crea el árbol a partir del `ndarray` que retorna la función `dataframe_to_ndarray(df)`, con un umbral mínimo de soporte. Retorna un objeto tipo `node` representando el árbol.
- **`mine_tree(fptree, min_support)`**: Recibe el árbol generado con el método anterior (objeto tipo `node`) y genera los itemsets frecuentes, es decir con un support mayor o igual al mínimo establecido. Estos itemsets puedes representarlos de la forma que te sea mas cómodo (`numpy` array, `DataFrame` o lista de listas), pero debes especificarlo en el mismo `.ipynb`.
- **`generate_association_rules(frequent_itemsets, confidence = 0, lift = 0)`**: Genera las reglas de asosciación a partir de los itemsets frecuentes generados por el método anterior. Es posible entregarle umbrales de confianza o *lift* para las reglas que se retornarán. Estas reglas puedes representarlas de la forma que te sea mas cómodo (`numpy` array, `DataFrame` o lista de listas), pero debes especificarlo en el mismo `.ipynb`.

3. Aplicar el algoritmo y obtener reglas de asociación

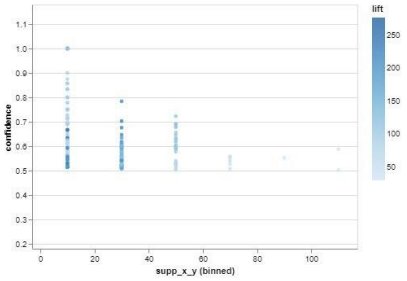
Deberás aplicar el algoritmo implementado en la base de datos entregada y filtrar las **mejores 10 reglas de asociación** de acuerdo a dos criterios de calidad **definidos por ti**. Deberás explicar por qué elegiste estos criterios.

4. Explicar las reglas obtenidas

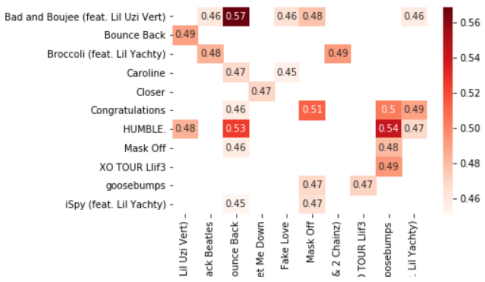
Deberás seleccionar 4 reglas y comentar su calidad de acuerdo a los diferentes indicadores disponibles (support, confidence y lift). Además, deberás identificar que implica la regla, y como se relaciona esto con los parámetros dichos.

5. Visualizar las reglas

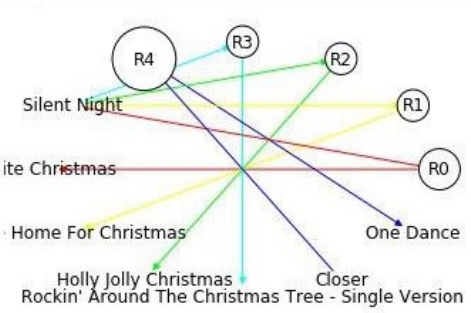
Para las reglas explicadas en el item 3, es decir, las 10 mejores reglas de asociación, deberás implementar una gráfica que permita entenderlas y discriminarlas de manera directa. En este punto, podrás hacer uso de todas las librerías de visualización disponibles según lo requieras. Algunos ejemplos pueden ser Altair, Matplotlib, Seaborn, Networkx, entre otras. Ejemplos de visualizaciones pueden ser las siguientes:



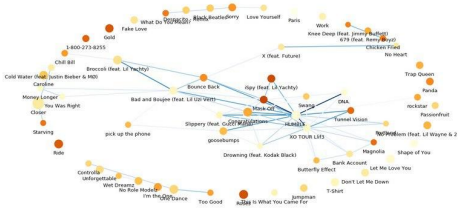
(a) Ejemplo 1



(b) Ejemplo 2



(a) Ejemplo 3



(b) Ejemplo 4

6. Comparar sus resultados con una librería

Por último, deberás comparar sus resultados con los que entrega la implementación del algoritmo por la librería mlxtend. Deberás contrastar los tiempos de ejecución de ambos algoritmos, los resultados entregados para un mismo input y comentar acerca de ello.

Bonus (0.5 puntos)

Se entregará un bonus a las **20** tareas que logren realizar el algoritmo en el **menor tiempo**. Se reitera que, para que puedas optar al bonus, **no puedes usar la librería itertools** para la implementación del algoritmo.

Entrega

En resumen, debes entregar en tu repositorio **sólo un archivo *.ipynb*** con el código y la documentación de las funciones pedidas. Debes **seguir el orden de actividades** especificadas en la sección actividades. Se aplicarán descuentos si no hay documentación o se entrega un informe muy desordenado.

Nota: Aprovecha el modo texto de *jupyter notebook* para entregar un informe más ordenado.

Evaluación

- **(0.5 pts)** Funciones `preprocess` y `dataframe_to_ndarray`.
- **(1.5 pts)** Clase `node` y método `create_tree`.
- **(1.0 pts)** Método `mine_tree`.
- **(1.0 pts)** Método `generate_association_rules`.
- **(1.0 pts)** Explicación de las reglas obtenidas.
- **(0.5 pts)** Visualización de las reglas obtenidas.
- **(0.5 pts)** Comparación de reglas con otra librería.

Descuentos

Se aplicarán los siguientes descuentos:

- Hasta 0.5 puntos por informes desordenados o mal escritos.
- 0.5 puntos por no documentar las funciones.
- 0.5 puntos por cada formato de la entrega que no se respete.

Referencias

- Christian Borgelt. An implementation of the fp-growth algorithm. En *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pages 1–5. ACM,2005.

- Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, A Inkeri Verkamo, et al. Fastdiscovery of association rules. *Advances in knowledge discovery and data mining*, 12(1):307–328,1996.