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

Tarea Recuperativa

Profesor Vicente Domínguez
8 de diciembre de 2020

Indicaciones

- Fecha de Entrega: 17 de Diciembre 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/1UP1bALw.
- 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 cualquier que no esté entre las siguientes: numpy, pandas, itertools y Networkx.
- Todo código entregado se comparará con implementaciones del algoritmo en internet para evitar la copia de este.
- Esta tarea es **OPCIONAL**, reemplazará tu peor nota en tareas y puede subir tu nota en tareas hasta tener un 4.0 como máximo, por lo tanto, **SÓLO** se tomarán en cuenta para la corrección alumnos que tengan promedio de tareas inferior a 4.0 al momento de hacerla. Es decir, si tienes un promedio de tareas inferior a 4.0, haces esta tarea y obtienes una mejor nota que tu peor nota en tareas, tu nota final de tareas será la siguiente:

  $$NT_{final} = min\{NT, 4.0\} \tag{1}$$

  Donde NT es el promedio que obtendrás reemplazando la peor nota de tareas por la nota que obtengas en esta.
- Solo serán evaluadas tareas que tengan implementado el algoritmo solicitado. En caso de usar una librería para omitir el paso de la implementación, **NO SE DARÁ PUNTAJE** por los demás items como un posible preprocesamiento o análisis de resultados.

Introducción

Esta tarea consiste en implementar el algoritmo **Apriori**, propuesto por Agrawal y Srikant en 1994. 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 playlists de la plataforma *spotify* creadas por usuarios de esta. Este es una muestra del dataset publicado para el RecSys Challenge 2018.

En específico, se entragará un solo archivo `.npy`:

- **`spotify.npy`** : Cada una de sus filas contiene una lista de canciones, la que representa una playlist de spotify. Un ejemplo de una de las filas del dataset se muestra a continuación:

```
[
    'Me & U',
    'Ice Box',
    'Sk8er Boi',
    'Run It!',
    'Check On It - feat. Bun B and Slim Thug',
    "Jumpin', Jumpin'",
    'Soak Up The Sun'
]
```

Nota: Cargar este archivo con la función `load` de `numpy`.

Se espera que en el punto **2.** de las actividades obtengan reglas de asociación de los ítems del dataset que obtendrán de `spotify.npy`. Mediante estas reglas, buscamos descubrir patrones de gustos musicales parecidos entre los usuarios de la plataforma.

Actividades

Las actividades que deberás realizar se detallan a continuación.

1. Implementar el algoritmo Apriori

Se debe implementar el algoritmo según lo visto en clases.

Para estructurar mejor tu código, deberás implementar las siguientes funciones:

- **`get_frequent_itemsets(playlists, min_support)`**: Recibe la esctructura de datos que contiene a las playlists y retorna una estructura con los itemsets frecuentes, bajo un umbral mínimo de confianza.
- **`generate_association_rules(frequent_itemsets, confidence = 0, lift = 0)`**: Recibe los itemsets frecuentes generados por la función anterior y retorna las reglas de asociación. Se le puede entregar umbrales de confianza o *lift* para las reglas que se retornarán. Por ejemplo, si se llama esta función con los argumentos `confidence = 0.5` y `lift = 1.2`, se espera que se retornen aquellas reglas que cumplan con una confianza ≥ 0.5 y un lift ≥ 1.2.

Puedes utilizar las estructuras de datos que estimes convenientes para representar la información, tanto de las **playlists**, los **itemsets frecuentes** y las **reglas de asociación**. Estas pueden ser lista de listas, `numpy` arrays o `pandas DataFrames`. Sin embargo, esto **debe estar claramente documentado en el mismo notebook**.

2. Aplicar el algoritmo y obtener reglas de asociación

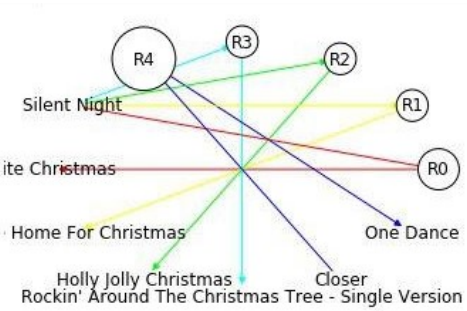
Deberás aplicar el algoritmo implementado en la base de datos entregada y filtrar las **mejores 10 reglas de asociación** de acuerdo a dos criterios de calidad **definidos por ti**. Deberás explicar por qué elegiste estos criterios.

3. Explicar las reglas obtenidas

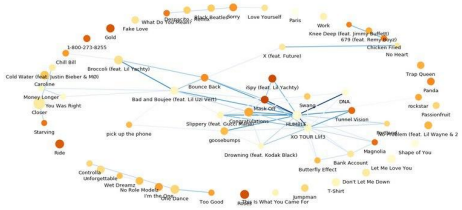
Deberás seleccionar 4 reglas y comentar su calidad de acuerdo a los diferentes indicadores disponibles (support, confidence y lift). Además, según el género y/o el artista de las canciones (que puedes buscar según el nombre de la canción) deberás darle una breve interpretación a las reglas.

4. 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, deberás hacer uso de la librería Networkx. De alguna forma se debe representar la confianza de las reglas, ya sea por la cercanía de los nodos, color de las aristas, etc. Ejemplos de visualizaciones pueden ser las siguientes:



(a) Ejemplo 1



(b) Ejemplo 2

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 **markdown** de *jupyter notebook* para entregar un informe más ordenado.

Evaluación

- **(2.0 pts)** Función `get_frequent_itemsets`.
- **(1.0 pts)** Función `generate_association_rules`.
- **(1.0 pts)** Aplicación de algoritmo a dataset de spotify, obtención de mejores 10 reglas y explicación criterios de calidad.
- **(1.0 pts)** Interpretación y explicación de 4 reglas.
- **(1.0 pts)** Visualización de las 10 mejores reglas obtenidas según tu criterio.

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.
- 0.5 por subir archivos que no sean .ipynb, .gitignore o readme.md.

Referencias

- Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, A Inkeri Verkamo, et al. Fast discovery of association rules. Advances in knowledge discovery and data mining, 12(1):307328, 1996.
- 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.