Python MrJobs

Para trabajar mejor con el archivos de datos, se importo un paquete de datos para poder cambiar el protocolo de entrada y salida.
Para esto se uso :

https://pypi.python.org/pypi/mr3px
Y se importo

```
from mr3px.csvprotocol import CsvProtocol
```

El materiar bibliografico que se uso fue :

http://stackoverflow.com/questions/4941753/is-there-a-math-ncr-function-in-python

Para funciones de combinatoria:

https://docs.python.org/2/library/itertools.html

Pregunta 1

La pregunta 1 cuenta con 1 mapeo, 1 combinador y 3 reductores.

El primer mapeo es del estilo: clave general , película
Este no es el peor método para formar parejas, ya que el rendimiento es bastante mediocre. Una mejor solución pudo haber sido formar parejas con las películas de los usuarios, y con esas parejas ir reduciendo hasta llegar al resultado esperado.

combine_count_word:
Este método es el mismo que usa la función itertools.combine(). Se arman parejas de películas a la fuerza y se le coloca un uno al lado para poder sumarlas posteriormente.

Los siguientes pasos fueron los mismos vistos en clases para encontrar el máximo. La diferencia es que ahora como clave vamos a tener un par de películas.

Tiempos:

Debido a que tuvimos un muy mal comienzo mapiando los datos, los tiempos de ejecución de este algoritmo son muy altos.

1000 datos → 15 segundos
10000 datos → 15 minutos
100000 datos → no ha terminado.

Pregunta 2

Para esta pregunta, primero se juntaron los usuarios con cada película que han visto. Después, se saco la cantidad de películas que había visto cada usuario y se le asocio a una película, de esta manera, vamos a tener como clave las películas con los usuario asociados a ellas y con la cantidad de películas por cada usuario.

El mapeo se realizo de la siguiente manera:
clave id usuario
valor película

Reductor 2:
en esta parte, asociamos cada usuario con su película, de esta manera, vamos a tener los usuarios con todas sus película.

Clave : usuario
valor [película 1, película 2....película n]

count_user_movies:
Para este reductor, solo se contó la cantidad de elementos en la lista de películas y se le asocio a cada tupla.
clave : película
valores : (usuario , numero de películas)

En esta parte se vuelve a usar el segundo reductor para volver a formar grupos. Esta vez, la clave son las películas con los usuarios asociados y la cantidad de películas que vieron.

Clave película
valores : (usuario1 ,3), (usuaio2,5)...(usuario N,7)

J**oin_movies**:

Con este método, lo que hacemos es formar parejas de usuarios por cada película, para así poder agruparlos por sus preferencias. También se les pone un uno para contabilizarlos mas adelante.

Calve : (usuario1,1), (usuaio2,3)
valor :1

count_movies:

Este método los cuenta las tuplas repetidas

clave : Calve : (usuario1,1), (usuaio2,3)
valor : cantidad de veces que se repite.

j**acard_reducer:**

En este método, sacamos el índice de Jacard y filtramos por los valores mayores a 0.5

Tiempos:

Los tiempos de ejecución no fueron los mas óptimos, sin embargo, con un numero acotado de datos se logro el resultado esperado.

2000 datos → 1 segundo
10000 datos → 1 segundo
100000 datos → 50 segundos
1000000 datos → 5 minutos
10000000 datos → 35 minutos, sin terminar.

Pregunta 3

Como cada usuario le puso una nota a una película, cada vez que un usuario estuvo asociado a un película, esta tiene una nota.

El mapeo se hizo colocando como:
clave: id película
valores: nota de el usuario

Reductor 1:
Para esto se juntaron todas las notas asociadas a una película. El valor de retorno es :
clave : id película
valores : [nota usuario 1, nota usuario 2, …., nota usuario n]

Reductor 3:

En este caso, solo contamos la cantidad de usuarios que habían visto la película. Para esto, basto con saber el largo del arreglo de notas. Con esto, nuestros valores quedan:
clave: id película
valores: ([nota usuario 1, nota usuario 2, …., nota usuario n] , n)
con n el largo del arreglo.

Reductor 4:
Este reductor solo junto las notas de los usuario y saco el promedio de ellas. Es claro que la forma de abordar el problema en este punto no fue el mas adecuado, ya que este reductor no esta reduciendo ningún termino.
Los valores finales son :
clave : None
valores [id película, promedio]

Esta es la parte mas ineficiente del algoritmo. Mas a delante se propondrán otras soluciones.
Reductor 5:
reductor Una vez se formaron los pares, se procede a calcular el promedio ponderado entre ambas películas.
Los valores de retorno son :
clave: None
valores : ([id película 1, id película 2], promedio)

No se llego al final del problema por temas de rendimiento del algoritmo, ya que el siguiente paso era aplicar otro reducer para agrupar las películas por sus máximos y mínimos. Por ahora solo encuentra el máximo.

Una mejor forma de abordar este problema pudo haber sido ir agrupando los pares de películas por los usuario que las vieron, sacar los pared por usuario, y a esa tuplas tratar de calculares el promedio.
Luego, una vez tengamos los promedios de las parejas por usuario, podríamos filtrar por calificaciones medias.
Una vez tenemos las parejas con sus promedios, asociados a los usuarios, se pueden tratar de reducir entre ellas sin tomar en cuanta a los usuarios.

Esta seria un mejor solcucion, pero por temas de tiempo no se alcanzo a implementar.

Tiempos:

1000 datos → 4 segundos
10000 datos → 2 minutos
100000 datos → 13 minutos
1000000 datos → no ha terminado