UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú

RECUPERACIÓ DE LA INFORMACIÓ (REIN)

Sessió 5 laboratori: MapReduce i anàlisi del comportament de compra

En aquesta sessió:

- Aprendrem a usar la llibreria de Python `MRJob` per programar `map-reduce` jobs.
- Implementarem una aplicació d'Anàlisi del comportament de compra (*Market Basket Analysis*), calculant i generant un conjunt de regles d'associació d'ordre 2 (amb un antecedent i un conseqüent) donat un suport i confiança mínims.

1 MRJob: programar `map-reduce` jobs

Hi ha diverses implementacions de l'entorn `map-reduce`. Hem escollit la llibreria `MRJob` perquè és flexible i ofereix la possibilitat d'usar un clúster de Hadoop o Spark o fins i tot serveis al núvol com Amazon Elastic MapReduce (EMR) o Google Cloud Dataproc.

Trobareu la documentació sobre com programar `map-reduce` jobs amb `mrjob` aquí.

La llibreria es basa en dues classes: `mrjob.job.MRJob`, que defineix un `map-reduce` job, i `mrjob.step.MRStep`, que permet definir diversos `map-reduce` passos en un objecte `MRJob`.

Per definir un job bàsic només cal crear un objecte `MRJob` i redefinir els mètodes `mapper` i `reducer`. En la documentació veureu que es poden definir més mètodes per fer coses abans i després d'un pas de map i reduce, i que també hi ha un pas addicional anomenat `combiner` que permet combinar resultats d'un pas de map abans de ser enviats al pas de reduce.

Els mètodes `mapper`, `combiner` i `reducer` reben, tots tres, una clau i un valor com a paràmetres. El `mapper` rep, en el paràmetre valor, un element de l'entrada cada vegada que és cridat, però el `combiner` i el `reducer` reben en el paràmetre valor un generador de Python que ha de ser recorregut per obtenir els seus valors i només pot ser recorregut una sola vegada.

La resta de mètodes no reben paràmetres i han d'usar estructures internes que creen i omplen els passos `map-reduce` durant la seva execució.

2 Un exemple de `map-reduce`

Suposeu que volem comptar les paraules d'una quantitat enorme de documents que tenim en una base de dades o en un sistema de fitxers. Una solució podria ser definir un programa `map-reduce` on el mapper dividís tots els documents en paraules i el reducer sumés les ocurrències de totes les paraules.

L'script `StreamDocs.py` transmet al canal estàndard de sortida tots els documents d'un índex d'una base de dades d'Elasticsearch. Tots aquests documents es poden passar a un programa `mrjob` perquè faci el recompte.

El programa de `map-reduce` següent, compta el nombre d'ocurrències de cada paraula en una seqüència de documents.

```
"""
WordCountMR
"""
from mrjob.job import MRJob
import re

WORD_RE = re.compile(r"[a-z]+")

class MRWordFrequencyCount(MRJob):

    def mapper(self, _, line):
        for word in WORD_RE.findall(line):
            yield word.lower(), 1

    def reducer(self, key, values):
        yield key, sum(values)


if __name__ == '__main__':
    MRWordFrequencyCount.run()
```

Codi 1: MRWordCount.py

Algunes qüestions importants són:

- Extraiem de la seqüència text pla. Per poder descartar tot allò que no siguin paraules, usem una estratègia de tokenització bàsica: buscar els strings que només continguin lletres usant una expressió regular. Per normalitzar el text, convertim totes les paraules en minúscules.

- Per fer el recompte, el mapper emet un parell (clau, valor) que consisteix en una paraula i l'enter 1.
- El reducer només suma tots els 1 que corresponen a una mateixa clau, en aquest cas, a una mateixa paraula.
- L'script escriu els resultats en el canal de sortida estàndard.

Per provar aquests scripts, escriviu en el terminal:[1]:

```
$ python3 StreamDocs.py --index inovels | python3 MRWordCount.py -r local
```

El paràmetre `-r local` simula un clúster local de Hadoop per executar el *job* en paral·lel. Podem controlar el nombre de processos que s'executen en paral·lel usant alguns flags de configuració, amb `--num-cores n` controlem a quants processos s'assignen als mappers i reducers (on `n` és el nombre de processos).

Si esteu usant una màquina amb diversos nuclis, podeu comprovar com es redueix el temps d'execució quan incrementeu el nombre de processos. No espereu obtenir una reducció lineal perquè no esteu treballant en un clúster de Hadoop real i, per tant, les dades no estan distribuïdes sinó que són transmeses per un sol procés. A més, hi ha molts processos de comunicació interns implicats.

3 Anàlisi del comportament de compra

L'Anàlisi del comportament de compra (*Market Basket Analysis*) fa referència al tipus d'anàlisis que es fan sobre bases de dades de compres efectuades per clients, anomenades transaccions.

L'entrada serà un fitxer csv: `groceries.csv`. Si mireu les primeres línies del fitxer, veureu que cada línia és una transacció, que bàsicament és un conjunt de productes que un client ha comprat alhora.

```
$ head -n 5 groceries.csv
citrus fruit,semi-finished bread,margarine,ready soups
tropical fruit,yogurt,coffee
whole milk
pip fruit,yogurt,cream cheese ,meat spreads
other vegetables,whole milk,condensed milk,long life bakery product
```

[1] Heu de tenir l'Elasticsearch executant-se i una col·lecció de documents indexats en l'índex `inovels`.

Per exemple, el primer client (línia 1) va comprar fruita, pa, margarina i sopa. El segon client (línia 2) va comprar fruita, iogurt i cafè. Etcètera.

La idea de l'anàlisi del comportament de compra és esbrinar els hàbits dels consumidors per tal d'explotar aquest coneixement en les campanyes de màrqueting, calcular la distribució dels articles en un supermercat per maximitzar-ne les vendes, etc.

3.1 Regles d'associació

Les regles d'associació ofereixen una manera d'entendre els hàbits de compra dels clients. En aquesta sessió de laboratori, s'explica una versió simplificada de les regles d'associació amb aquesta forma general:

IF $item_1$ és comprat **THEN** $item_2$ també és comprat

Representarem aquesta regla com $item_1 \rightarrow item_2$. El que haureu de buscar són, doncs, parells de productes que habitualment es comprin alhora.

Per l'avaluació d'una regla d'associació s'usen dos paràmetres: *suport* i *factor de confiança*.

Suport. El suport d'una regla $item_1 \rightarrow item_2$ és el nombre de vegades que el parell $item_1$ i $item_2$ apareix en les transaccions. Es pot donar com a valor absolut o com a valor relatiu (en forma de percentatge).

Confiança. El factor de confiança d'una regla $item_1 \rightarrow item_2$ és el percentatge de transaccions que contenen $item_1$ que també contenen $item_2$. És una mena de mesura condicional: de totes les transaccions que contenen $item_1$, quantes contenen també $item_2$?

Vegem un exemple per explicar aquests conceptes. Suposeu que la vostra base de dades de transaccions és:

```
a,b,c
a,c
a,d
```

Aleshores, la regla d'associació $a \rightarrow d$ té un suport del 33% i una confiança del 33%, i la regla $b \rightarrow c$ té un suport del 33% i una confiança del 100%.

3.2 L'extracció de regles d'associació

La tasca d'extracció de regles d'associació és la següent: donat un *llindar de suport* s i un *llindar de confiança* c (en percentatge), trobar totes les regles que tinguin un suport mínim de s i un factor de confiança mínim de c.

Com a estratègia general per resoldre el problema, és molt útil tenir els recomptes de suport d'**un sol** ítem i els recomptes de suport de **parells** d'ítems.

A partir d'aquests recomptes, es calculen el suport i la confiança de les regles com segueix:

$$sup(X \to Y) = \frac{count(\{X, Y\})}{count(\{\})}$$

i

$$conf(X \to Y) = \frac{count(\{X, Y\}}{count(\{X\})}$$

on $count(S)$ és el nombre de transaccions que contenen el conjunt d'ítems S. Observeu que en aquestes equacions, $count(\{\})$ és el nombre de transaccions de la base de dades, atès que el conjunt buit el contenen totes les transaccions.

Tornant a l'exemple de la base de dades de transaccions

```
a,b,c
a,c
a,d
```

els càlculs serien

$$sup(a \to b) = \frac{count(\{a, b\})}{count(\{\})} = \frac{1}{3}$$

i

$$conf(a \to b) = \frac{count(\{a, b\})}{count(\{a\})} = \frac{1}{3}$$

4 Tasca

Heu d'escriure un programa que permeti extraure regles d'associació a partir de les dades d'entrada contingudes en un fitxer csv. És obligatori fer ús del paradigma `map-reduce` per fer els recomptes dels suports d'un sol ítem i dels suports de parells d'ítems.

Es proposa que implementeu les tasques següents:

- Un o més jobs amb les funcions `map-reduce` que comptin en quantes transaccions apareix cada ítem i també en quantes transaccions apareixen parells d'ítems.
- Un programa que, donats uns llindars de suport i confiança, faci servir els recomptes del codi `map-reduce` anterior per generar la llista de regles d'associació que superen els dos llindars: el de suport i el de confiança.

La implementació `map-reduce` incompleta de l'anàlisi del comportament de compra que us donem, està dividida en tres scripts:

- `MRMarketBasketAnalysis.py` és el programa principal que conté les crides als `map-reduce` jobs i processa els seus resultats.
- `MRMarketBasket1.py` està incomplet i ha d'implementar el job que fa el recompte de parells d'ítems.
- `MRMarketBasket2.py` està incomplet i ha d'implementar el job que fa el recompte d'un sol ítem.

Llegiu els comentaris escrits en els scripts per tenir més detalls del que cal fer.

Per defecte, `MRMarketBasketAnalysis.py` configura `MRjob` per usar dos mappers i dos reducers en mode `local`. Podeu modificar aquests valors usant els paràmetres `--ncores`. Feu proves amb diferents valors i recolliu estadístiques sobre els temps d'execució.

Atenció! MRjob usa el canal estàndard de sortida per processos de comunicació interna, per tant, no escriviu (`print`) directament al canal de sortida estàndard des del vostre job perquè això podria confondre els processos.

5 Lliuraments

Heu d'escriure un breu report (2-3 pàgines màxim) en el que respongueu les preguntes que s'indiquen a continuació. En el report podeu incloure les dificultats i decisions més importants que hàgiu hagut de prendre durant la implementació. També heu de lliurar tots els scripts que hàgiu implementat o aquells que hàgiu modificat.

El que heu de penjar a la Tasca d'Atenea per aquesta activitat és un fitxer `.zip` que contingui:

1. Un fitxer en format `.pdf` amb el report. Recordeu d'incloure els vostres noms i el de l'activitat.
2. Els scripts `.py` implementats i/o modificats.

Pregunta 1. A partir de la vostra implementació, ompliu la taula següent; els valors de la primera fila ja se us donen calculats.

Fila	*Suport*	*Confiança*	*Nre. de regles d'associació trobades*
1	1%	1%	426
2	1%	25%	?
3	1%	50%	?
4	1%	75%	?
5	5%	25%	?
6	7%	25%	?
7	20%	25%	?
8	50%	25%	?

Pregunta 2. Doneu la llista de regles d'associació que hàgiu trobat corresponents a les files 4, 5 i 6 de la taula de la pregunta 1.