Relazione su Frequent Itemsets Mining

Fontolan Federico 854230
Rosada Fabio 851772

January 21, 2017

1 Introduzione

L'esercitazione che ci è stata assegnata prevedeva di implementare un algoritmo che prendesse in analisi dieci milioni di tweet e ritornasse i frequent itemsets; il nostro gruppo ha deciso di implementare un algoritmo random sampling perché la grande mole dei dati permette di raggiungere risultati soddisfacenti anche solo analizzandone una parte relativamente piccola come il 10% senza compromettere la validità dei dati essendo lo stesso campione estremamente grande.
Il core del nostro codice è stato scritto nella sua interezza in simultanea dai membri del gruppo, alternandosi nella scrittura come previsto nella tecnica del "pair programming". La revisione finale del codice con la riscrittura dei commenti e l'eliminazione delle righe superflue è stata affidata a Fabio Rosada, mentre la scrittura di questa relazione a Federico Fontolan.

2 Scelte Fatte nella Stesura del Codice

Il nostro codice all'avvio, dopo avere stampato i dettagli del settaggio corrente (sample, threshold e file di input), genera uno SparkContext e carica in memoria la lista di stopwords che servirà per la rimozione delle stesse: la peculiarità di queste parole è quella di essere estremamente ricorrenti, quindi ogni itemset che le contenga sarà di scarso interesse.
Successivamente leggiamo ed estraiamo un sample con la combinazione delle funzioni textFile() e sample() e generiamo una lista rappresentante il contenuto dei bucket: questa scelta permette di limitare il numero di letture del file che è la parte tempisticamente più onerosa permettendoci di eseguire l'intero programma in meno di dieci minuti. Nel file è comunque presente sotto forma di commento una versione funzionante del software che legge i file ogni volta questo sia necessario per eliminare lo spazio occupato da questa lista che però causa un abbattimento generale delle prestazioni.
La lettura dei tweet viene effettuata considerando i tweet come testo e non come json: questo scelta è dovuta alla lentezza nell'elaborazione del file con il secondo metodo riscontratasi nella fase di test. Finita questa fase viene eseguito un algoritmo apriori che continua a ricercare itemsets finché esiste almeno un itemset di rango inferiore: questo è dovuto al principio di monotonicità.

3 Risultati

NOTA: le funzioni per sinteticità mostrano i primi 20 risultati ordinati per frequenza, ma è possibile visualizzarli cambiando i valori di input nel codice.

```
#NOTA#
tempo di esecuzione: 7.9 minuti


### main.py ###

File: /srv/2015-01-08_geo_en_it_10M.plain.json
Threshold: 1000
Sample size: 0.1
Finita la lettura del basket...

Finito il conteggio delle parole...


        TOP 20 ABSOLUTE

1       love: 49010
2       just: 46679
3       like: 42163
4       follow: 30957
5       don: 30400
6       amp: 27490
7       good: 25804
8       know: 22369
9       lol: 21365
10      day: 21125
11      time: 21120
12      new: 20812
13      want: 19514
14      happy: 18860
15      today: 18384
16      got: 18317
17      need: 16991
18      people: 16686
19      make: 15972
20      life: 15216
```

```
Inizio calcolo rank: 2
lunghezza risultati: 48

        TOP 20 ABSOLUTE

1       (u'follow', u'love'): 6696
2       (u'@harry_styles', u'follow'): 3482
3       (u'birthday', u'happy'): 3194
4       (u'@harry_styles', u'love'): 2753
5       (u'don', u'know'): 2635
6       (u'just', u'like'): 2128
7       (u'don', u'like'): 2068
8       (u'don', u'just'): 2032
9       (u'new', u'watch'): 2027
10      (u'video', u'watch'): 2009
11      (u'good', u'morning'): 1916
12      (u'feel', u'like'): 1841
13      (u'follow', u'nplease'): 1830
14      (u'dream', u'love'): 1818
15      (u'love', u'nplease'): 1702
16      (u'happy', u'love'): 1584
17      (u'love', u'make'): 1549
18      (u'don', u'want'): 1515
19      (u'just', u'want'): 1491
20      (u'love', u'nyou'): 1457


Inizio calcolo rank: 3
lunghezza risultati: 15

        TOP 20 ABSOLUTE

1       (u'@harry_styles', u'follow', u'love'): 2203
2       (u'new', u'video', u'watch'): 1694
3       (u'just', u'photo', u'posted'): 1355
4       (u'follow', u'love', u'nplease'): 1297
5       (u'@fifthharmony', u'prize', u'win'): 1267
6       (u'@fifthharmony', u'entering', u'prize'): 1221
7       (u'@fifthharmony', u'daily', u'prize'): 1220
8       (u'@fifthharmony', u'fanuary', u'prize'): 1217
9       (u'barometer', u'temperature', u'today'): 1197
10      (u'@fifthharmony', u'harmony', u'prize'): 1192
11      (u'@fifthharmony', u'fifth', u'prize'): 1191
12      (u'dream', u'follow', u'love'): 1181
13      (u'@themattespinosa', u'video', u'watch'): 1133
14      (u'barometer', u'humidity', u'temperature'): 1125
15      (u'barometer', u'rain', u'temperature'): 1061
```

```
Inizio calcolo rank: 4
lunghezza risultati: 7

        TOP 20 ABSOLUTE

1       (u'@fifthharmony', u'daily', u'prize', u'win'): 1266
2       (u'@fifthharmony', u'entering', u'prize', u'win'): 1263
3       (u'@fifthharmony', u'fanuary', u'prize', u'win'): 1259
4       (u'@fifthharmony', u'harmony', u'prize', u'win'): 1234
5       (u'@fifthharmony', u'fifth', u'prize', u'win'): 1233
6       (u'barometer', u'humidity', u'temperature', u'today'): 1191
7       (u'@themattespinosa', u'new', u'video', u'watch'): 1045


Inizio calcolo rank: 5
lunghezza risultati: 6

        TOP 20 ABSOLUTE

1       (u'@fifthharmony', u'daily', u'entering', u'prize', u'win'): 1266
2       (u'@fifthharmony', u'daily', u'fanuary', u'prize', u'win'): 1216
3       (u'@fifthharmony', u'daily', u'harmony', u'prize', u'win'): 1193
4       (u'@fifthharmony', u'daily', u'fifth', u'prize', u'win'): 1192
5       (u'barometer', u'humidity', u'temperature', u'today', u'wind'): 1191
6       (u'barometer', u'humidity', u'rain', u'temperature', u'today'): 1191


Inizio calcolo rank: 6
lunghezza risultati: 4

        TOP 20 ABSOLUTE

1       (u'@fifthharmony', u'daily', u'entering', u'harmony', u'prize', u'win'): 1239
2       (u'@fifthharmony', u'daily', u'fifth', u'harmony', u'prize', u'win'): 1191
3       (u'barometer', u'humidity', u'rain', u'temperature', u'today', u'wind'): 1191
4       (u'@fifthharmony', u'daily', u'fanuary', u'harmony', u'prize', u'win'): 1189


Inizio calcolo rank: 7
lunghezza risultati: 2

        TOP 20 ABSOLUTE

1       (u'@fifthharmony', u'daily', u'entering', u'fifth', u'harmony', u'prize', u'win'): 1238
2       (u'@fifthharmony', u'daily', u'entering', u'fanuary', u'harmony', u'prize', u'win'): 1236


Inizio calcolo rank: 8
lunghezza risultati: 1

        TOP 20 ABSOLUTE

1       (u'@fifthharmony', u'daily', u'entering', u'fanuary', u'fifth', u'harmony', u'prize', u'win'): 1235


Inizio calcolo rank: 9
lunghezza risultati: 0
Non ci sono piu' risultati utili
```

Commenti Per questo esperimento è stato scelto un threshold molto basso(lo 0.001) per permettere di ottenere anche un buon numero di tuple di rango superiore al 2.
Nel corso di vari testi sia con il file maggiore sia con quello minore abbiamo notato che alcune particolari tuple molto lunghe compaiono un numero molto elevato di volte: questo fenomeno succede se un tweet viene retwettato un numero di volte superiore al threshold.

4 Confronto

```
#NOTA#
tempo di esecuzione: 16 secondi

### main.py ###

File: /srv/sample.json
Threshold: 50
Sample size: 1
Finita la lettura del basket...

Finito il conteggio delle parole...


        TOP 20 ABSOLUTE

1       love: 500
2       just: 462
3       follow: 428
4       like: 392
5       don: 321
6       amp: 283
7       new: 246
8       today: 243
9       good: 239
10      day: 233
11      happy: 229
12      know: 225
13      want: 214
14      lol: 214
15      world: 200
16      @madisonellebeer: 190
17      people: 186
18      time: 183
19      thanks: 182
20      got: 174

Inizio calcolo rank: 2
lunghezza risultati: 21

        TOP 20 ABSOLUTE

1       (u'@madisonellebeer', u'follow'): 110
2       (u'@madisonellebeer', u'nplease'): 99
3       (u'follow', u'love'): 83
4       (u'birthday', u'happy'): 68
5       (u'@madisonellebeer', u'ignore'): 65
6       (u'@madisonellebeer', u'ncan'): 65
7       (u'@madisonellebeer', u'ndon'): 65
8       (u'@madisonellebeer', u'world'): 65
9       (u'@madisonellebeer', u'nyou'): 65
10      (u'@madisonellebeer', u'nilysfm'): 65
11      (u'@madisonellebeer', u'nmy'): 65
12      (u'@feriafitur', u'world'): 58
13      (u'@feriafitur', u'retweet'): 58
14      (u'2015', u'@feriafitur'): 58
15      (u'@justinbieber', u'love'): 58
16      (u'@feriafitur', u'travelblogger'): 58
17      (u'@feriafitur', u'vote'): 58
18      (u'@feriafitur', u'nelsoncarvalheiro'): 58
19      (u'happy', u'song'): 52
20      (u'new', u'song'): 50
```

Risultati La fase di confronto è stata eseguita con sul file da diecimila tweet e concentriamo la nostra attenzione fino al rango due dei risultati trovati dal nostro software. Si può notare che le parole frequenti individuate (escluse stopwords) sono le stesse, anche sono un valore leggermente inferiore dovuto al sistema di pulizia eseguito sulla stringa. Il nostro software è leggermente più veloce (circa 2-3 secondi) rispetto a fpgrowth con il file piccolo.

```
FreqItemset(items=[u'im'], freq=694)
FreqItemset(items=[u'love'], freq=519)
FreqItemset(items=[u'just'], freq=464)
FreqItemset(items=[u'follow'], freq=441)
FreqItemset(items=[u'dont'], freq=413)
FreqItemset(items=[u'like'], freq=406)
FreqItemset(items=[u'can'], freq=343)
FreqItemset(items=[u'please'], freq=304)
FreqItemset(items=[u'one'], freq=286)
FreqItemset(items=[u'amp'], freq=269)
FreqItemset(items=[u'new'], freq=267)
FreqItemset(items=[u'good'], freq=237)
FreqItemset(items=[u'u'], freq=231)
FreqItemset(items=[u'go'], freq=228)
FreqItemset(items=[u'happy'], freq=226)
FreqItemset(items=[u'know'], freq=222)
FreqItemset(items=[u'please', u'follow'], freq=221)
FreqItemset(items=[u'now'], freq=220)
FreqItemset(items=[u'want'], freq=218)
FreqItemset(items=[u'today'], freq=216)
FreqItemset(items=[u'day'], freq=208)
FreqItemset(items=[u'lol'], freq=207)
FreqItemset(items=[u'youre'], freq=203)
FreqItemset(items=[u'world'], freq=196)
FreqItemset(items=[u'madisonellebeer'], freq=190)
FreqItemset(items=[u'thanks'], freq=182)
FreqItemset(items=[u'people'], freq=180)
FreqItemset(items=[u'cant'], freq=177)
FreqItemset(items=[u'got'], freq=176)
FreqItemset(items=[u'time'], freq=175)
FreqItemset(items=[u'much'], freq=175)
FreqItemset(items=[u'follow', u'love'], freq=162)
FreqItemset(items=[u'need'], freq=162)
FreqItemset(items=[u'year'], freq=162)
FreqItemset(items=[u'back'], freq=162)
FreqItemset(items=[u'job'], freq=160)
FreqItemset(items=[u'make'], freq=158)
FreqItemset(items=[u'see'], freq=156)
FreqItemset(items=[u'really'], freq=152)
FreqItemset(items=[u'never'], freq=147)
FreqItemset(items=[u'think'], freq=143)
FreqItemset(items=[u'life'], freq=143)
FreqItemset(items=[u'still'], freq=141)
FreqItemset(items=[u'always'], freq=137)
FreqItemset(items=[u'birthday'], freq=135)
FreqItemset(items=[u'thats'], freq=135)
FreqItemset(items=[u'madisonellebeer', u'follow'], freq=134)
FreqItemset(items=[u'hi'], freq=132)
FreqItemset(items=[u'thank'], freq=130)
FreqItemset(items=[u'come'], freq=128)
FreqItemset(items=[u'everything'], freq=125)
FreqItemset(items=[u'cold'], freq=124)
FreqItemset(items=[u'work'], freq=123)
FreqItemset(items=[u'night'], freq=122)
FreqItemset(items=[u'going'], freq=122)
FreqItemset(items=[u'2'], freq=121)
FreqItemset(items=[u'dream'], freq=121)
FreqItemset(items=[u'madisonellebeer', u'please'], freq=120)
FreqItemset(items=[u'madisonellebeer', u'please', u'follow'], freq=118)
FreqItemset(items=[u'best'], freq=118)
FreqItemset(items=[u'2015'], freq=116)
FreqItemset(items=[u'can', u'follow'], freq=114)
FreqItemset(items=[u'even'], freq=112)
FreqItemset(items=[u'\U0001f62d'], freq=108)
FreqItemset(items=[u'shit'], freq=107)
FreqItemset(items=[u'youre', u'follow'], freq=105)
FreqItemset(items=[u'school'], freq=104)
FreqItemset(items=[u'take'], freq=100)
```

5 Conclusione

In generale possiamo ritenerci soddisfatti dei risultati, i tempi di esecuzione sono accettabili e i risultati anche, purtroppo alcuni risultati risultano errati a causa di alcuni tweet che si ripetono, ma escluso questo le tuple trovate sono soddisfacenti.

Purtroppo non abbiamo avuto modo di testare a fondo e di ottimizzare sia l'algoritmo oggetto di questa esercitazione, che l'algoritmo contenuto in MLIB, a causa di code che si sono venute a formare nel cluster fornito dall'università e a causa di alcuni inconvenienti che si sono verificati con persone che terminavano i processi altrui.