INFO 209 - HW 1

Eloi Pereira

September 14, 2012

1 Learn the tutorial

1.1

The `DISTINCT` keyword is used to eliminate duplicates in data. In this particular case the duplicates may arise due to the fact that the `NGramGenerator` generates all ngrams of size 1 and 2. The concatenation of two ngrams of size 1 may create another ngram of size 2 which may lead to duplicate ngrams.

1.2

The command `GROUP A by f` groups the tuples of relation `A` according to the specification `f`. It creats a partition of `A` where the first field is named `group` and contains the values in which the grouping has been performed. The second field are bags of records that meet the grouping specification.

In the example `hour_frequency1=GROUP ngramed2 BY (ngram,hour);`, the first field of `hour_frequency1` contains tuples `(ngram,hour)` where `ngram` is a n-gram and `hour` is the hour, and the second field contains the bags for each pair `(ngram,hour)`.

1.3

The operator `FLATTEN` flattens data structures removing "nesting". The operator is polymorphic since it can be applied to tuples and bags. The semantics of applying `FLATTEN` to tuples or bags is quite different so one must use it with care.

The example `hour_frequency2 = FOREACH hour_frequency1 GENERATE flatten($0), COUNT($1) as count;` applies `FLATTEN` to tuples of the kind `(ngram,hour)` since it is flattening the first column ($0). Thus, `flatten($0)` flattens the keys of the group with the result of `COUNT($1)` which counts the number of elements in each bag of each group. The result is a tuple of the kind `(ngram,hour,count)`.

1.4

The command `uniq_frequency1 = GROUP hour_frequency2 BY group::ngram;` is taking the record of tuples `hour_frequency2` and group them by the `ngram`, producing tuples of the kind `(ngram, {(ngram,hour,count)})` where the first field is the key of the group.

2 Compute overall query log statistics

For each question, it is presented the pig code with some comments on the reasoning behind the implementation. Together with this report it is provided a pig file named `hw1.pig` containing all the code in this report. We also provide all the results in folders named `hwa` where `a` is the problem number.

```
REGISTER ./tutorial.jar;
raw = LOAD ’excite-small.log’ USING PigStorage(’\t’) AS (user, time, query);
```

2.a

```
-- question 2.a: The total number of query records
-- group all the records and count the elements in the group
allGrouped = GROUP raw ALL;
count = FOREACH allGrouped GENERATE COUNT(raw);
STORE count INTO ’Hw2a’ USING PigStorage();
-- answer (4501)
```

2.b

```
-- question 2.b: The maximum query length in words
-- tokenize the queries, count tokens in each query, group them all and take the maximum
removeEmpty = FILTER raw BY org.apache.pig.tutorial.NonURLDetector(query);
withTokens = FOREACH removeEmpty GENERATE TOKENIZE(query) as tokens;
countTokens = FOREACH withTokens GENERATE COUNT(tokens) as numTokens;
groupedTokens = GROUP countTokens ALL;
maxTokens = FOREACH groupedTokens GENERATE MAX($1);
STORE maxTokens INTO ’Hw2b’ USING PigStorage();
-- answer (14)
```

2.c

```
-- question 2.c: The average query length in words
-- same as per 2.b except that we take the average instead of the maximum
avgTokens = FOREACH groupedTokens GENERATE AVG($1);
```

```
STORE avgTokens INTO ’Hw2c’ USING PigStorage();
-- answer (2.445388974755281)
```

2.d

```
-- question 2.d: The total number of unique users
-- group by users, take the keys of the groups, group them all, and count them
usersGroup = GROUP raw BY user;
listUsers = FOREACH usersGroup GENERATE group;
listGrouped = GROUP listUsers ALL;
countUsers = FOREACH listGrouped GENERATE COUNT(listUsers);
STORE countUsers INTO ’Hw2d’ USING PigStorage();
-- answer (891)
```

2.e

```
-- question 2.e: The average number of query records per user
-- for each users group count their bags, group them all, and take the average
usersNumQueries = FOREACH usersGroup GENERATE COUNT(raw);
usersNumQueriesGroupAll = GROUP usersNumQueries ALL;
avgUsersNumQueries = FOREACH usersNumQueriesGroupAll GENERATE AVG($1);
STORE avgUsersNumQueries INTO ’Hw2e’ USING PigStorage();
-- answer (5.051627384960718)
```

2.f

```
-- question 2.f: What percent of query records contain queries with Boolean operators (AND,OR,NOT,or+)
-- use regexp to filter queries (previously lower cased), count them and calculate the percentage
lowerCase = FOREACH removeEmpty GENERATE user, time, LOWER(query) as query;
withBool = FILTER lowerCase BY query matches ’(.*((( and )|( or )|( not )|(\u002b))).*)’;
withBoolDist = DISTINCT withBool;
groupWithBoolAll = GROUP withBoolDist ALL;
countBool = FOREACH groupWithBoolAll GENERATE COUNT(withBoolDist);
pair = CROSS countBool, count;
percentageBool = FOREACH pair GENERATE (double)$0/(double)$1*100;
STORE percentageBool INTO ’Hw2f’ USING PigStorage();
-- answer (6.2208)
```

2.g

```
-- question 2.g: The 10 longest distinct queries
-- generate tuples of queries and their size, group them all, take the top 10
queriesAndSize = FOREACH removeEmpty GENERATE query, SIZE(query) as querySize;
queriesAndSize2 = DISTINCT queriesAndSize;
grp = GROUP queriesAndSize2 ALL;
top10 = FOREACH grp {
  sorted = order queriesAndSize2 by querySize desc;
  top = limit sorted 10;
  generate group, flatten(top);
};
STORE top10 INTO ’Hw2g’ USING PigStorage();
-- answer:
-- (all,alanta,georgia/contractors independent excavation contractor excavating subcontractors
--  construction earthwork bidding ,118)
-- (all,alanta,georgia/contractors independent excavation contractor excavating subcontractors
--  construction earthwork ,110)
-- (all,alanta,georgia/contractors independent excavation contractor excavating subcontractors
--  construction ,100)
-- (all,alanta,georgia/contractors independent excavation contractor excavating subcontractors ,87)
-- (all,taper optic tapers fiber fibre waveguide optics waveguides cladding optical photonics ,86)
-- (all,iron and steel +oil and gas pipelines +valves +flanges +russia +ukraine,71)
-- (all,clow piping ductile supply awwa hydrants valves couplings fittings ,67)
-- (all,blues guitar "stevie ray vaughn" vaughan songs interviews pictures ,67)
-- (all,stockings panties exhibitionist girlfriends housewives schoolgirl ,66)
-- (all,accoustic guitars guild guitar stringed instrument instruments ,63)
```

2.h

```
-- question 2.h: The 10 most frequently occurring queries
-- group queries by query, count the number of queries inside each bag, group them all and take the top 10
queries = FOREACH removeEmpty GENERATE query;
queriesGrp = GROUP queries BY query;
queriesCount = FOREACH queriesGrp GENERATE group, COUNT(queries) as freq;
grp2 = GROUP queriesCount ALL;
top10_freq = FOREACH grp2 {
  sorted2 = order queriesCount by freq desc;
  top_freq = limit sorted2 10;
  generate group, flatten(top_freq);
};
STORE top10_freq INTO 'Hw2h' USING PigStorage();
-- answer:
-- (all,maytag,41)
-- (all,vanderheiden,27)
-- (all,change bowel habits,24)
-- (all,en vogue,23)
-- (all,running shoes,22)
-- (all,pregnant,20)
-- (all,ebony divas black,19)
-- (all,yahoo chat,16)
-- (all,the byker wall,16)
-- (all,jarrow,16)
```

3 Compute user session information

This question is not mandatory.

4 Combine with another data set

4.a

```
-- question 4.a: Find queries with references to US zip codes
-- use regexp to filter matches with US Postal zip. the regexp takes 0 or more characters,
-- followed by 5 digits and an optional dash with more 4 digits and finished again with
-- 0 or more characters.
```

```
distQueries = DISTINCT queries;
withZip = FILTER distQueries BY query matches ’.*(\\d{5}(-\\d{4})?).*’;
STORE withZip INTO ’Hw4a’ USING PigStorage();
-- answer:
-- (tv AND 90210)
-- (beverly hills 90210)
-- ("beverly hills 90210")
-- ("beverley hills 90210")
```

4.b

```
-- question 4.b: Find queries with references to place names
-- load data for location names, create records of token-query pairs, join the places names with the
-- token-query pairs. Take the queries and save.
rawWorld = LOAD ’dataen.txt’ USING PigStorage(’\t’) AS (uid: int, nameEn: chararray, nameAlt: chararray,
 nameOrig: chararray, type: chararray, pop: int, lat: int, lon: int, parenCtry: chararray, parnAdm1: chararray,
 parenAdm2: chararray, parenAdm3: chararray);
worldNames = FOREACH rawWorld GENERATE LOWER(nameEn) as nameEn;
worldNamesSample = SAMPLE worldNames 0.01;
tokenQueryPairs = FOREACH lowerCase GENERATE FLATTEN(TOKENIZE(query)) as tokens, query;
tokenQueryPairs2 = DISTINCT tokenQueryPairs;
worldNameTokenQuery = JOIN worldNames BY nameEn, tokenQueryPairs2 BY tokens;
worldNameTokenQuery2 = DISTINCT worldNameTokenQuery;
queriesWithPlaces = FOREACH worldNameTokenQuery2 GENERATE query;
STORE queriesWithPlaces INTO ’Hw4b’ USING PigStorage();
-- answer: too large to include in the report
```

4.c

```
-- question 4.c: Find other indicators of geographic locations
-- look for references of addresses since they are probably associated with
-- a location (except if it is an email)
lowerDistQueries = FOREACH distQueries GENERATE LOWER(query) as query;
withAddr = FILTER lowerDistQueries BY query matches ’.*addr.*’;
withAddrNoEmail = FILTER withAddr BY query matches ’^(?:(?!email).)*$’;
STORE withAddrNoEmail INTO ’Hw4c1’ USING PigStorage();
-- result:
-- (business addressess)
-- (business addressess businesses )
-- (business addressess businesses companies )
-- (crime stoppers denver colo. address phone number)
-- (business addressess businesses companies directories )
-- The result shows 1 positive and 4 false positives

-- look for weather references. usually associated with a place
withWeather = FILTER lowerDistQueries BY query matches ’.*weather.*’;
-- results: no matches

-- look for school(s)/college(s)/university(ies) references
withSchool = FILTER lowerDistQueries BY query matches ’.*(( school)|( universit)|( college)).*’;
STORE withSchool INTO ’Hw4c2’ USING PigStorage();
-- result: 32 matches, 25 positives, 7 false positives
```

4.d

```
-- question 4.d: Further improve on (b) by doing something clever about ambiguous place names.
-- Exclude odd location names that increase a lot the false positives.
-- For example, there is a commune in the Burkina Faso named "To".
worldNamesRemoveSome = FILTER worldNames BY NOT nameEn matches '((to)|(at)|(by)|(da)|(di)|(ii)|(of))';
worldNameTokenQuery3 = JOIN worldNamesRemoveSome BY nameEn, tokenQueryPairs2 BY tokens;
worldNameTokenQuery4 = DISTINCT worldNameTokenQuery3;
queriesWithPlaces2 = FOREACH worldNameTokenQuery4 GENERATE query;
STORE queriesWithPlaces2 INTO 'Hw4d' USING PigStorage();
-- result: too large to include on the report.
```