Cloud Computing

Kamakshi Bansal
UCL
Student ID: 16114486

Assignment Part 1

Conditional probability (pairs)

Code: Bigram/mr_conditional_prob_pairs.py

- Each mapper takes a sentence:
  - Generate all co-occurring term pairs
  - For all pairs, emit (a, b) → count
- Reducers sum up counts associated with these pairs

Pre processing of data:

Pre processing of data is done so as to consider the last word of the first line and the first word of the consecutive line. The following mappers and reducers are used:

def init_get_words(self):
This function holds the "prevLineLastWord" variable so that it can refer it while processing the first word of next line in the mapper to count the bigram.

def mapper_get_words(self,_,line)

This mapper reads each line. It then strips and splits the data set and then cleans each and every word. After cleaning, this mapper forms the bigram (word pairs) and throws the (current word, next word) as key and the count 1 as value.
It also throws unigram words with count as 1

The code also considers the bigram of the last word of the first line and the first word of the second line, in order to read bigram words across lines.

The code also ignores the two word sequence across paragraph boundaries as required.

def combiner_count_words(self, word, counts):
This combines the intermediate output of the mapper, i.e. it sums the count for the same key and sends it to the reducer.

def reducer_count_words(self, word, counts):
This reducer further sums the count for the same key which it receives from different combiners.

def reducer_find_max_word(self, _, words):
This reducer finds the total number of unigrams for each word and total number of pairs and finally finds the conditional probability for each a word X that occurs immediately after another word w, i.e., **Pr[X |w] = count(w, X)/count(w)** for each two-word-sequence (w, X) .
The final result gives the top 10 conditional probability for each a word X that occurs immediately after another word “for” , i.e.,

Pr[X | for] = count(for, X)/count(for) for each two-word-sequence (for, X)

Steps used to run the file on AWS EMR :

Kamakshi@ThinkpadX220 MINGW64 /d/UCL/Cloud Computing/Cloud Computing/Bigram
$ python mr_conditional_prob_pairs.py datacondition/* -r emr -c etc/.mrjob.conf

OUTPUT OF AWS EMR

Streaming final output from
s3://mapreduce22/tmp/mr_conditional_prob_stripes.Kam
akshi.20161218.195131.463000/output/...
7171
["for", "the"] 0.13680100404406637
["for", "her"] 0.07195649142379026
["for", "a"]   0.06902802956351974
["for", "his"] 0.029005717473155765
["for", "it"]   0.028029563519732254
["for", "him"] 0.022730442058290334
["for", "i"]   0.018686375679821503
["for", "me"]   0.018686375679821503
["for", "you"] 0.01784967229117278
["for", "my"]   0.017012968902524055
Removing s3 temp directory
s3://mapreduce22/tmp/mr_conditional_prob_stripes.Kama
kshi.20161218.195131.463000/...
Removing temp directory
c:\users\kamakshi\appdata\local\temp\mr_conditional_prob
_stripes.Kamakshi.20161218.195131.463000...
Removing log files in s3://mapreduce22/logs/j-6UKCKN8TB90M/...
Terminating cluster: j-6UKCKN8TB90M

Conditional probability (Stripes)

Code: Bigram/mr_conditional_prob_pairs.py

- Each mapper takes a sentence:
  - Generate all co-occurring term pairs
  - For each term, emit

    $a \rightarrow \{ b: count_b, c: count_c, d: count_d \ldots \}$
- Reducers perform element-wise sum of associative arrays

$$
\begin{array}{rl}
 & a \rightarrow \{ b: 1, \quad d: 5, e: 3 \quad \} \\
+ & a \rightarrow \{ b: 1, c: 2, d: 2, \quad f: 2 \} \\
\hline
= & a \rightarrow \{ b: 2, c: 2, d: 7, e: 3, f: 2 \}
\end{array}
$$

The following mapper and reducers are used :

def init_get_words(self):
This function holds the "prevLineLastWord " variable so that it can refer it while processing the first word of the next line in the mapper to count the bigram.

def mapper_get_words(self, _, line):
This mapper receives a line as an input and creates 2 D dictionary and stores current word and next word and counts the bigram pairs. **This mapper also considers the last word of the first line and the first word of the consecutive line and ignores the paragraph boundaries.** It yields the unigram and bigram count in the form of stripes and propagates it to the reducer.

def reducer_count_words(self, word, bigram):
This mapper receives the unigram and bigram counts from the mapper and sums it up for the same key and sends it to the reducer after shuffling and sorting.

def reducer_find_max_word(self, _, words):
This reducer finds the total number of unigrams for each word and total number of pairs and finally finds the conditional probability for each a word X that occurs immediately after another word w, i.e., **Pr[X |w] = count(w, X)/count(w)** for each two-word-sequence (w, X) .
The final result gives the top 10 conditional probability for each a word X that occurs immediately after another word "for" , i.e.,

Pr[X | for] = count(for, X)/count(for) for each two-word-sequence (for, X)

Steps used to run the file on AWS EMR :

Kamakshi@ThinkpadX220 MINGW64 /d/UCL/Cloud Computing/Cloud Computing/Bigram
$ python mr_conditional_prob_stripes.py datacondition/* -r emr -c etc/.mrjob.conf

OUTPUT OF AWS EMR
Streaming final output from s3://mapreduce22/tmp/mr_conditional_prob_stripes.Kam
akshi.20161218.195131.463000/output/...

["for", "the"] 0.13680100404406637
["for", "her"] 0.07195649142379026
["for", "a"] 0.06902802956351974
["for", "his"] 0.029005717473155765
["for", "it"] 0.028029563519732254
["for", "him"] 0.022730442058290334
["for", "i"] 0.018686375679821503
["for", "me"] 0.018686375679821503
["for", "you"] 0.01784967229117278
["for", "my"] 0.017012968902524055
Removing s3 temp directory s3://mapreduce22/tmp/mr_conditional_prob_stripes.Kama
kshi.20161218.195131.463000/...
Removing temp directory c:\users\kamakshi\appdata\local\temp\mr_conditional_prob
_stripes.Kamakshi.20161218.195131.463000...
Removing log files in s3://mapreduce22/logs/j-6UKCKN8TB90M/...
Terminating cluster: j-6UKCKN8TB90M

Assignment part 2

Page Rank

Code: pagerank/page_rank.py

The following mapper and reducers are used :

No. of iterations: 15

def mapper_get_total_nodes(self, _, line):
This mapper gets the total number of nodes and yields None as the key and each line as the value along with the total nodes.

def reducer_get_total_nodes(self, node, lines):
This reducer collects the values for the same key and combines them and propagates further.

def mapper_get_nodes(self, _, lines):
This mapper splits each line and assign from node id to Y and to node id to X and yields key as a tuple containing (node id, total number of nodes) and value as the outlinks to that node id.

def reducer_get_nodes(self, y, x):
This reducer collects the values for the same key and combines them and propogates further.

def mapper_inital(self, y, x):
This mapper assigns intial pprobabilities to each node and then yields a tuple containing node id and total no of nodes, and value as probability (page rank)

def reducer_inital(self, y, x):

Since now this reducer will receive key as the node id and value as probability and node id's of the outlinks. It checks for the value, if it is float value, it sums up the page rank, else it yields the node id and the outlink node id. It also yields the node id as key and updated page rank as value using :

$$P(n) = \alpha \left(\frac{1}{|G|}\right) + (1-\alpha) \sum_{m \in L(n)} \frac{P(m)}{C(m)}$$

With damping factor= 0.85

def reducer_final_merge(self,y,x):

This reducer sums up all the page rank it receives from the above reducer and propogates further.

def reducer_final_display(self,_,node):

This reducer sorts the input it receives according the page rank in descending order and displays the top ten.

Steps used to run the file on AWS EMR :

Kamakshi@ThinkpadX220 MINGW64 /d/UCL/Cloud Computing/Cloud Computing/pagerank
$ python page_rank.py data/* -r emr -c etc/.mrjob.conf

OUTPUT OF AWS EMR

```
Streaming final output from
s3://mapreduce22/tmp/page_rank.Kamakshi.20161218.220449.916000/output/...
0.0018580304371090194   18
0.0011964409875482162   737
0.0008968011065799934   1719
0.0008409033844204463   143
0.000840655004603865    118
0.0008376113382148554   790
0.0008141538527633274   136
0.000723356854849778    40
0.0006120679133850399   1619
0.0005947965240505457   1179
Removing s3 temp directory
s3://mapreduce22/tmp/page_rank.Kamakshi.20161218.220449.916000/...
```

```
Removing temp directory
c:\users\kamakshi\appdata\local\temp\page_rank.Kamakshi.20161218.220449.916000...
Removing log files in s3://mapreduce22/logs/j-3JANELSUDKQXB/...
Terminating cluster: j-3JANELSUDKQXB
```