Bigrams Processing with Amazon S3 & EMR

Khai Nguyen
khainguyen@temple.edu
CIS 4517 Data Intensive and Cloud Computing

(50 points) Part 1: MapReduce
Count bigrams: Take the word count from the previous problem and extend it to count bigrams. bigrams are sequences of two consecutive words. (Recall our lecture on shingles.) Don't worry about doing anything fancy in terms of tokenization. You can use Java's StringTokenizer.
Deliverables (on Pride and Prejudice):

1. (10 points) How many unique bigrams are there?
View number of words OR number of lines since each line is designated for 1 word:

$ find /c /v "" "./output/combined.txt"

```
C:\Users\KhaiNguyen\Documents\CS_4517\Project\Project_5\BigramCount\output> find /c /v "" "./combined.txt"

---------- ./COMBINED.TXT: 53092

C:\Users\KhaiNguyen\Documents\CS_4517\Project\Project_5\BigramCount\output>
```

Give 3 examples of such bigrams.

Grabbing data starting at a random line, for example line 20330

```
C:\Users\KhaiNguyen\Documents\CS_4517\Project\Project_5\BigramCount\output>more +20330 combined.txt
Her_&_mother's  2
Her_&_mother    3
her_&_mother    51
her_&_Mr        5
her_&_Mrs       2
her_&_much      1
her_&_must      1
her_&_My        1
her_&_name      2
her_&_natural   1
her_&_nature    3
her_&_nearest   1
her_&_neighbours'       1
her_&_neighbours        2
her_&_nephew's  1
her_&_nephew    5
```

2. (5 points) List the top ten most frequent bigrams and their counts.

Using pandas, we can import our recently generated output file, “combined.txt” into a dataframe. Then we can sort by the number of appearances of the bigrams. **Here, 2 bigrams are connected with a “_&_” string.**

In [44]:
```
import pandas as pd
import os
file_path = os.path.join("./BigramCount", "output", "combined.txt")
price_and_prejudice_df = pd.read_csv(file_path, sep="\s+", header = None)

price_and_prejudice_df.columns = ["Bigram", "Num_appearance"]

top_ten_bigrams = price_and_prejudice_df.sort_values(by="Num_appearance", ascending=False).head(10)

top_ten_bigrams
# top_ten_bigrams.sum()
```

Out[44]:

	Bigram	Num_appearance
32521	of_&_the	456
46451	to_&_be	420
23289	in_&_the	383
22315	I_&_am	299
32172	of_&_her	252
47034	to_&_the	241
29362	Mr_&_Darcy	226
32179	of_&_his	215
18296	had_&_been	195
22408	I_&_have	181

3. (10 points) What fraction of all bigrams occurrences does the top ten bigrams account for? That is, what is the cumulative frequency of the top ten bigrams?

In [56]:
```
cumulative_frequency_top_ten = float(top_ten_bigrams["Num_appearance"].sum())/\
                               float(price_and_prejudice_df["Num_appearance"].sum())

print("Cumulative frequency of top ten is {:0.4f}\n".format(cumulative_frequency_top_ten))
```

```
Cumulative frequency of top ten is 0.0248
```

4. (5 points) How many bigrams appear only once?

In [72]:
```
appear_once = price_and_prejudice_df[price_and_prejudice_df["Num_appearance"]==1]
print("Number of bigrams that appear once " + str(len(appear_once)))
```

```
Number of bigrams that appear once 39297
```

Give 3 examples of bigrams that appear only once.

In [73]:
```
appear_once = price_and_prejudice_df[price_and_prejudice_df["Num_appearance"]==1]
appear_once.tail(10)
```

Out[73]:

	Bigram	Num_appearance
53080	yourself_&_when	1
53082	yourself_&_your	1
53083	yourselves_&_Do	1
53084	yourselves_&_so	1
53086	youth_&_as	1
53087	youth_&_health	1
53088	youth_&_the	1
53089	youth_&_there	1
53090	youth_&_was	1
53091	youths_&_as	1

5. (10 points) What are the five most frequent words following the word "light"? What is the frequency of observing each word?

From below, we can tell that the 5 most frequent words are *in, A, and, as, began*

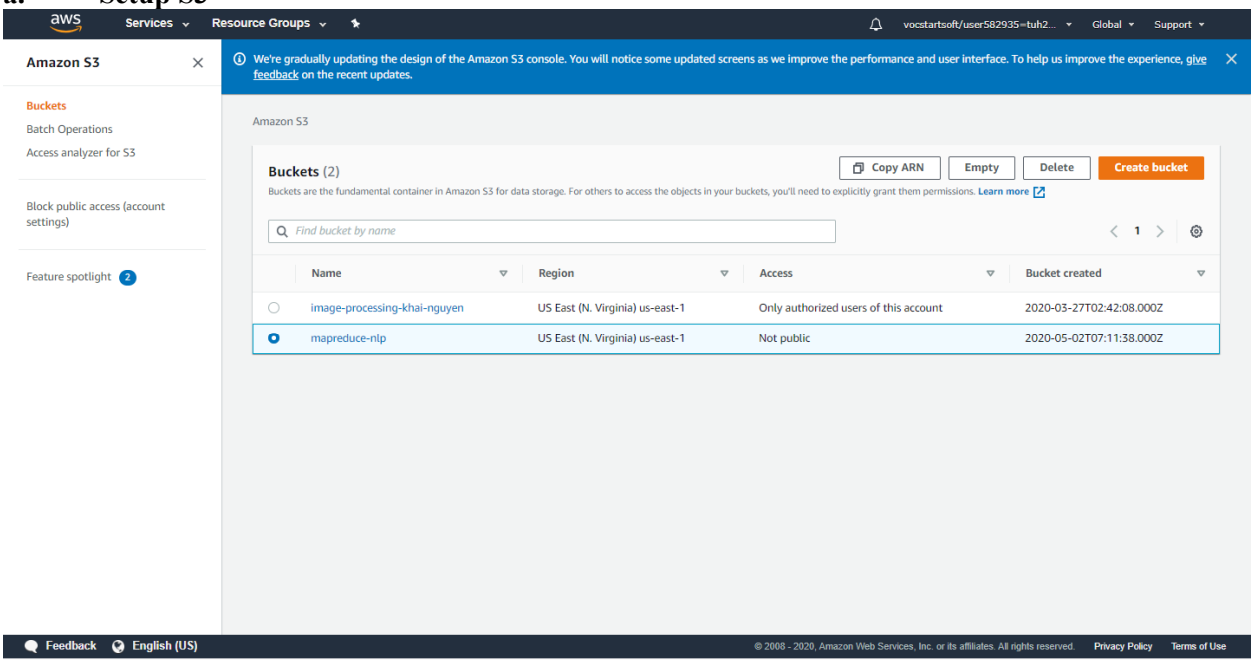
```
In [80]: #  What are the five most frequent words following the word "light"? What is the frequency of observing each word?
bigram_with_light = price_and_prejudice_df[price_and_prejudice_df["Bigram"].str.match('light_&_')]
bigram_with_light.sort_values(by="Num_appearance", ascending=False).head(100)
```

Out[80]:

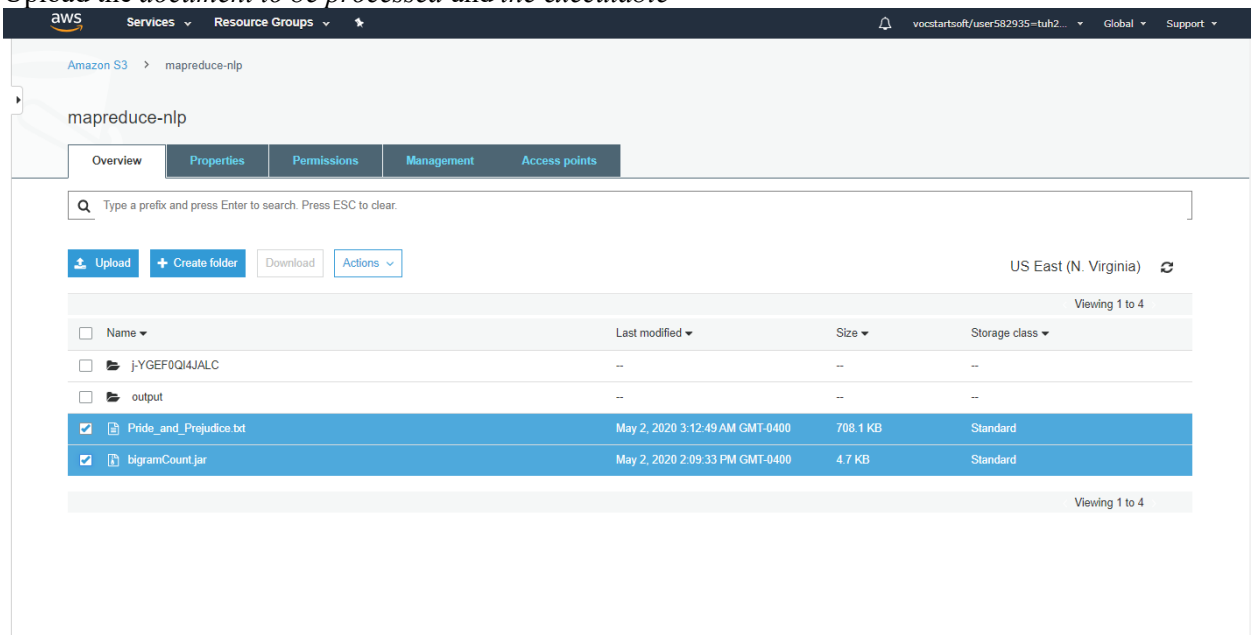
	Bigram	Num_appearance
25826	light_&_in	2
25817	light_&_A	1
25818	light_&_and	1
25819	light_&_as	1
25820	light_&_began	1
25821	light_&_does	1
25822	light_&_have	1
25823	light_&_He	1
25824	light_&_I	1
25825	light_&_importance	1
25827	light_&_it	1
25828	light_&_might	1
25829	light_&_nor	1
25830	light_&_seemed	1

6. (10 points) Running in the cloud.

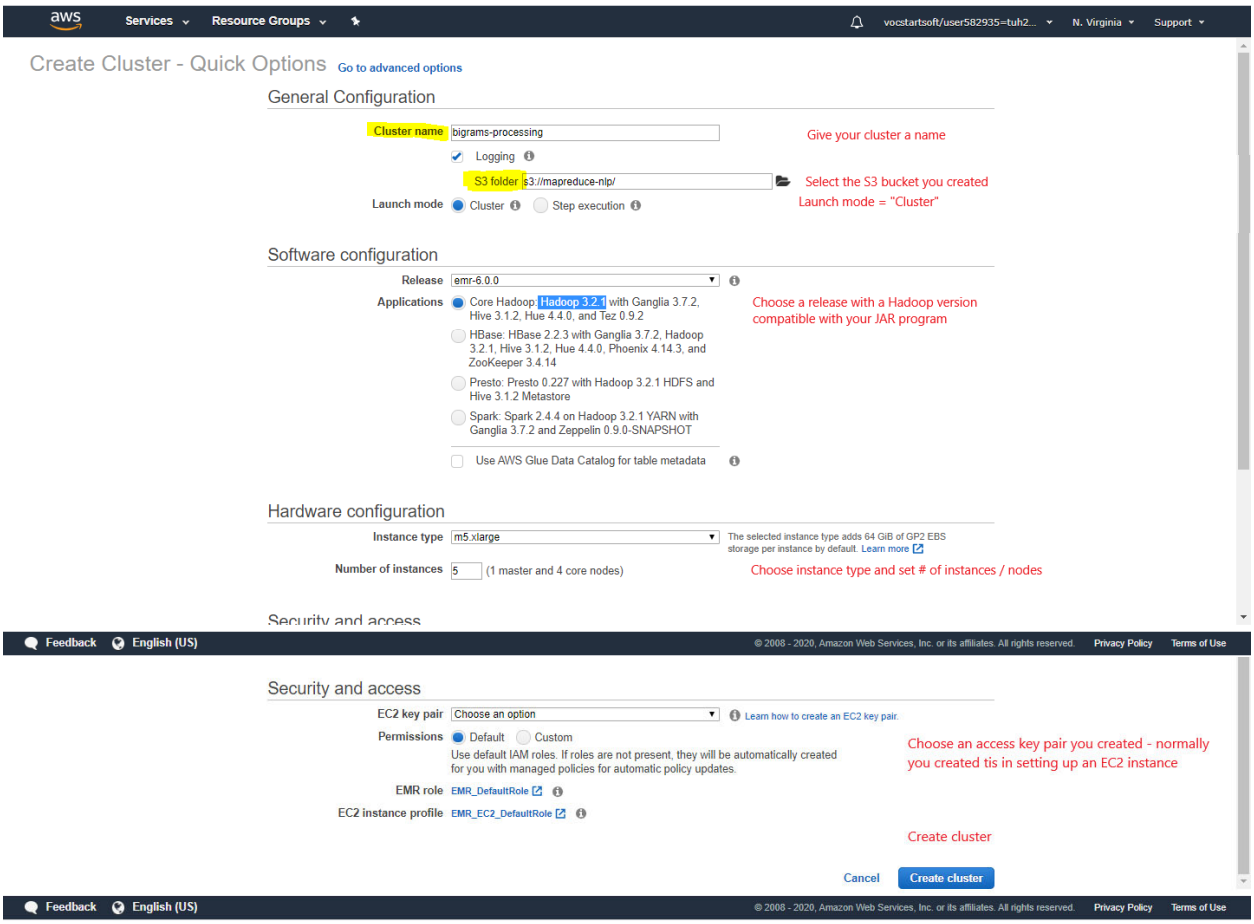
a. Setup S3



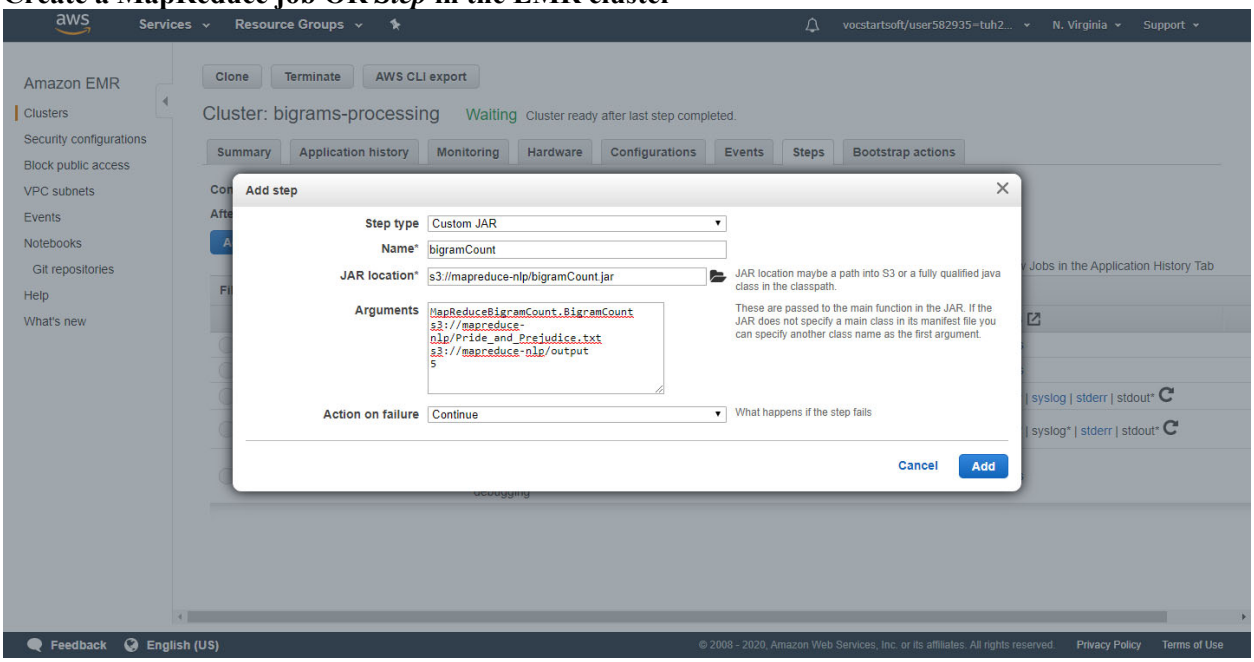
Upload the *document to be processed* and *the executable*



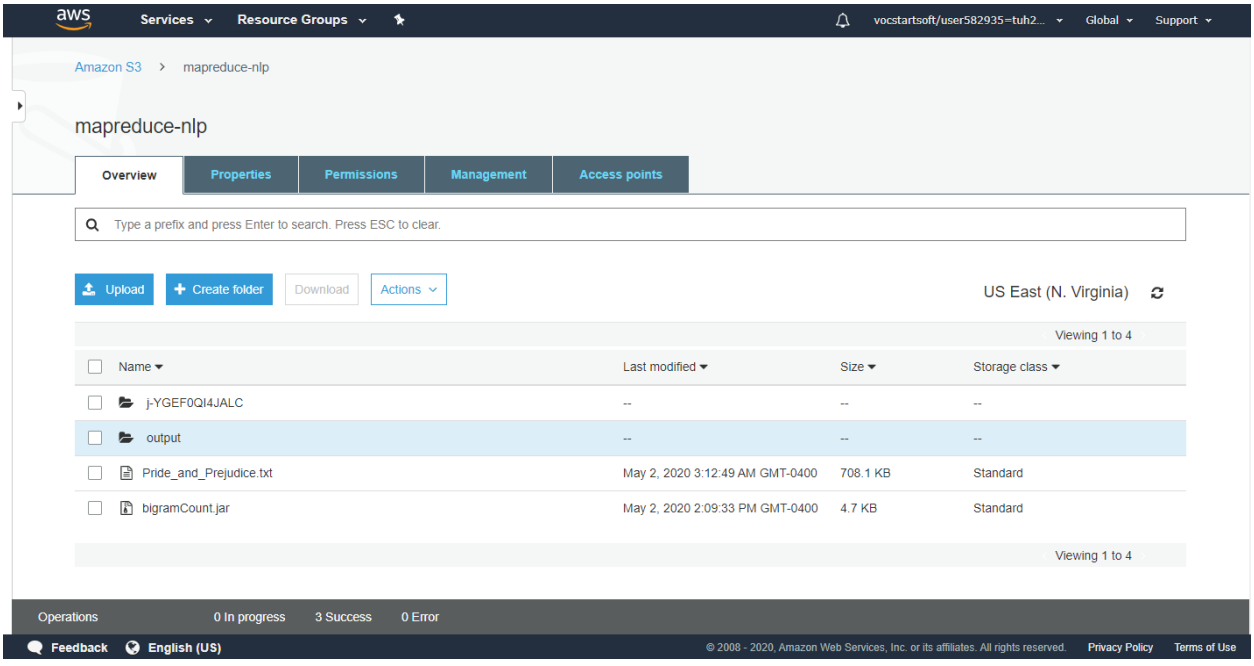
b. Setup EMR - Amazon Elastic MapReduce

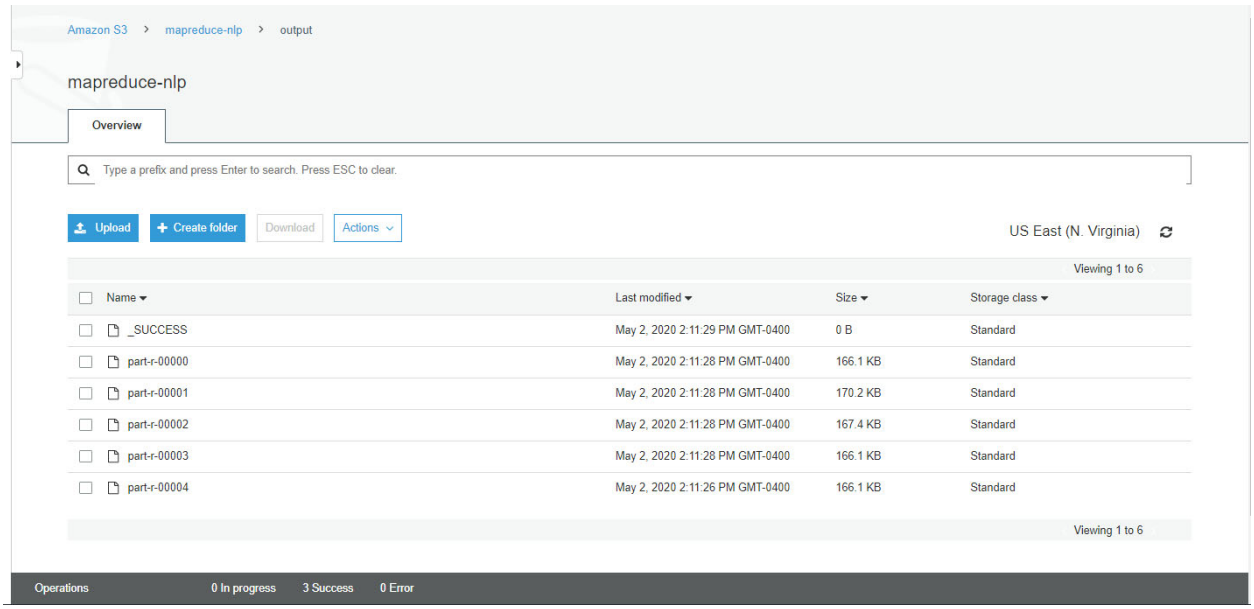


c. Create a MapReduce job OR *Step* in the EMR cluster

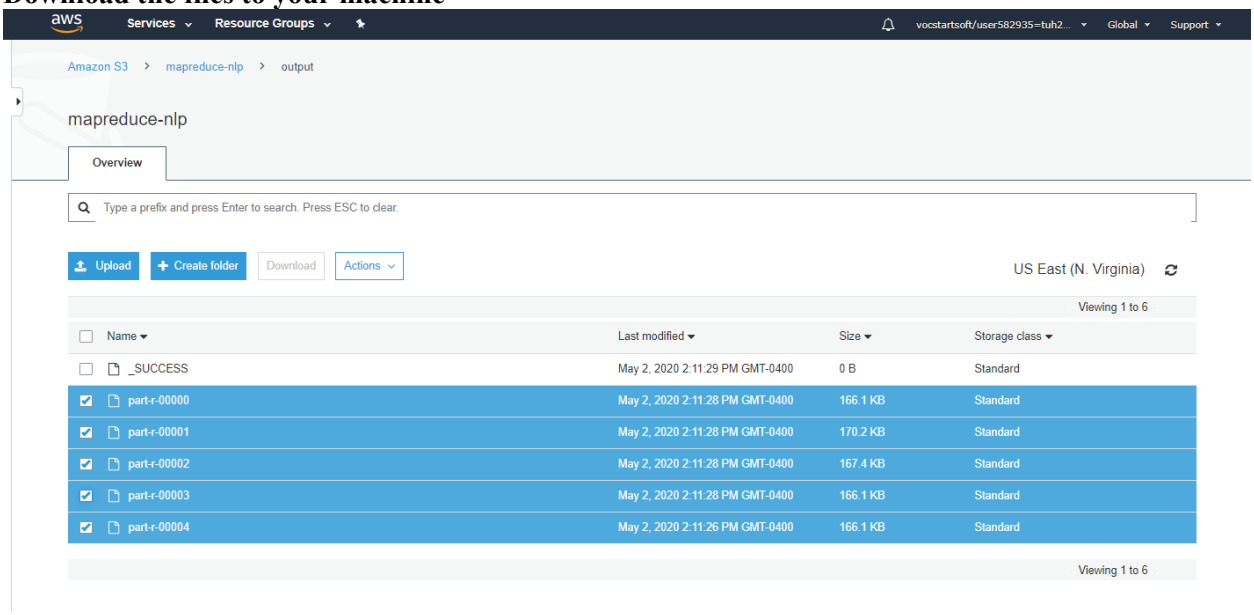


d. Go back to the S3 console to check the result

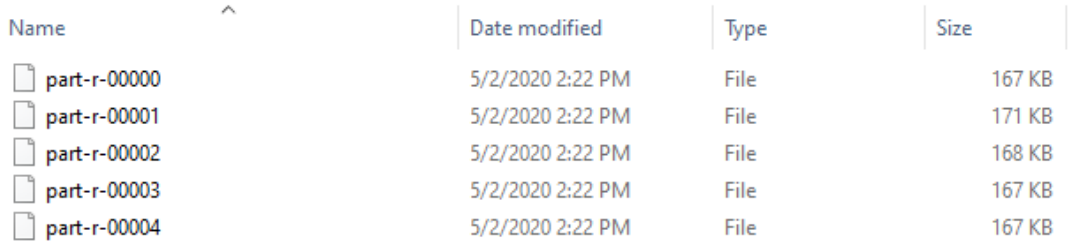




e. Download the files to your machine



On machine:



f. Combine and sort

$ copy part-r-* temp_combined.txt

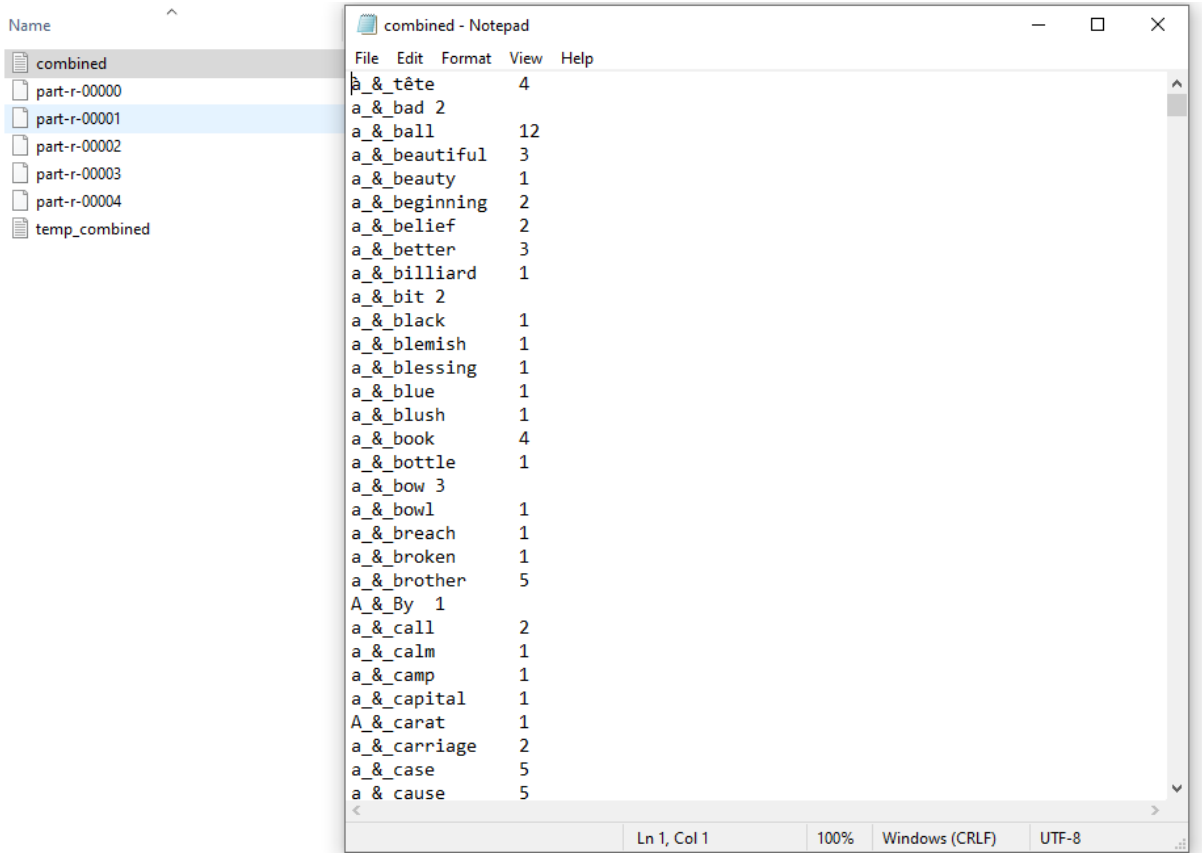
```
C:\Users\KhaiNguyen\Documents\CS_4517\Project\Project_5\BigramCount\output_EMR>copy part-r-* temp_combined.txt
part-r-00000
part-r-00001
part-r-00002
part-r-00003
part-r-00004
        1 file(s) copied.
```

$ sort temp_combined.txt > combined.txt

```
C:\Users\KhaiNguyen\Documents\CS_4517\Project\Project_5\BigramCount\output_EMR>sort temp_combined.txt > combined.txt

C:\Users\KhaiNguyen\Documents\CS_4517\Project\Project_5\BigramCount\output_EMR>
```

g. Check the result



(50 points) Part 2: Pandas
For this problem use the data crawled from your Project 2.

1. (5 points) Describe the that you use to solve this problem. (Hint: get it from your report.)
BeautifulSoup4 : https://pypi.org/project/beautifulsoup4/
python tika : https://pypi.org/project/tika/
Pandas: https://pypi.org/project/pandas/
IDE: VSCode and Jupyter Notebook

2. (5 points) Import the Pandas package.
$ pip3 install pandas

3. (5 points) Load data into separate Data Frames. (All of you have collected multiple files. Work with at least 2 files for this problem.)
a. Use csv functions to load data from csv files if your data is in csv.
b. Use json functions to load data from json documents if your data is in json.

```
In [25]: # a) Import the Pandas package.
         import pandas as pd
         import os

         # b) Load data into separate DataFrames. Work with at least 2 files for this problem.
         territory_data_1 = pd.read_csv(os.path.join("./csv","2020-04-11_Switzerland.csv"))
         territory_data_2 = pd.read_csv(os.path.join("./csv","US", "2020-04-11_California.csv"))
```

4. (5 points) Check the data-type of each of column by outputting the dttypes attribute of your DataFrame.

```
In [17]: territory_data_1["Grocery & pharmacy"].dtype
Out[17]: dtype('O')
```

```
In [18]: territory_data_1.dtypes
Out[18]: Territory             object
         Retail & recreation   object
         Grocery & pharmacy    object
         Parks                 object
         Transit stations      object
         Workplace             object
         Residential           object
         dtype: object
```

5. (5 points) Show an example of sorting one of your DataFrames by a column. Give the top-15 entries in descending order.

Here we sort by the “Parks” column

```
In [31]: territory_data_1.sort_values(by="Parks", ascending=False).head(15)
```

Out[31]:

	Territory	Retail & recreation	Grocery & pharmacy	Parks	Transit stations	Workplace	Residential
17	Schaffhausen	-63%	+30%	Not enough data for this date	-34%	-39%	+2%
2	Appenzell_Ausserrhoden	-62%	+13%	Not enough data for this date	-21%	-33%	Not
3	Appenzell_Innerrhoden	-80%	Not enough data for this date	Not enough data for this date	Not enough data for this date	-84%	Not
23	Uri	-74%	+0%	Not enough data for this date	-54%	-48%	Not
7	Canton_of_Zug	-68%	+46%	Not enough data for this date	-40%	-44%	+22%
10	Glarus	-87%	-6%	Not enough data for this date	+5%	-35%	Not
12	Jura	-82%	+12%	Not enough data for this date	-34%	-40%	+14%
15	Nidwalden	-62%	+0%	-5%	-50%	-46%	Not
16	Obwalden	-83%	-4%	-49%	-40%	-40%	Not
24	Valais	-85%	-8%	-35%	-60%	-52%	+18%
11	Grisons	-85%	-15%	-35%	-65%	-49%	+14%
14	Neuchâtel	-70%	+8%	-18%	-37%	-47%	+23%
22	Ticino	-83%	-21%	-18%	-50%	-50%	+20%
1	Aargau	-74%	+19%	+99%	-32%	-36%	+15%
26	Zurich	-78%	+20%	+96%	-55%	-43%	+15%

6. (10 point) Give an example of using filtering.

a. Give an example for vertical filtering/slicing where you select a subset of the columns. This corresponds to a projection in a SELECT statement.

```
In [57]: # Filtering

         # a) Give an example for vertical filtering/slicing where you select a subset of the
         # columns. This corresponds to a projection in a SELECT statement.
         vertical_slice_1 = territory_data_1[['Transit stations', 'Workplace']]
         vertical_slice_1.head(15)
```

Out[57]:

	Transit stations	Workplace
0	-47%	-41%
1	-32%	-36%
2	-21%	-33%
3	Not enough data for this date	-84%
4	-63%	-37%
5	-33%	-37%
6	-50%	-36%
7	-40%	-44%
8	-37%	-41%
9	-74%	-49%
10	+5%	-35%
11	-65%	-49%
12	-34%	-40%
13	-47%	-39%
14	-37%	-47%

b. Give an example for **horizontal filtering/slicing** where you select a subset of the rows in your DataFramer according to some criteria. This corresponds to WHERE in SELECT statement.

In [58]:
```
# b) Give an example for vertical filtering/slicing where you select a subset of the
# rows in your DataFramer according to some criteria. This corresponds to WHERE
# in SELECT statement
numbers = ["-15%", "+30%"]

horizontal_slice_1 = territory_data_1[territory_data_1["Grocery & pharmacy"].isin(numbers)]
horizontal_slice_1.head()
```

Out[58]:

	Territory	Retail & recreation	Grocery & pharmacy	Parks	Transit stations	Workplace	Residential
11	Grisons	-85%	-15%	-35%	-65%	-49%	+14%
17	Schaffhausen	-63%	+30%	Not enough data for this date	-34%	-39%	+2%

7. (10 points) Show an example where you **merge two DataFrames**.

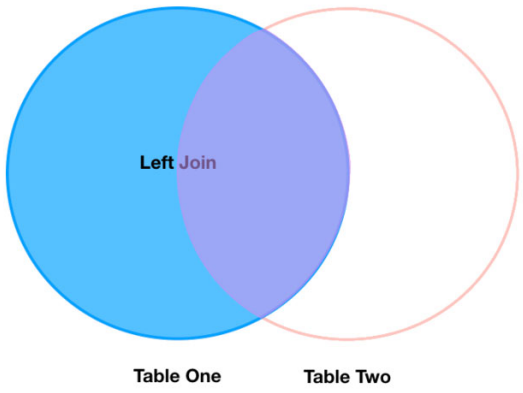
Perform a concat()

In [64]:
```
# 7) Merge 2 dataFrames
concat_territory_1 = pd.concat([vertical_slice_1, horizontal_slice_1], ignore_index=True)
concat_territory_1.tail(15)
```

Out[64]:

	Transit stations	Workplace	Territory	Retail & recreation	Grocery & pharmacy	Parks	Residential
14	-37%	-47%	NaN	NaN	NaN	NaN	NaN
15	-50%	-46%	NaN	NaN	NaN	NaN	NaN
16	-40%	-40%	NaN	NaN	NaN	NaN	NaN
17	-34%	-39%	NaN	NaN	NaN	NaN	NaN
18	-26%	-36%	NaN	NaN	NaN	NaN	NaN
19	-42%	-35%	NaN	NaN	NaN	NaN	NaN
20	-25%	-36%	NaN	NaN	NaN	NaN	NaN
21	-29%	-24%	NaN	NaN	NaN	NaN	NaN
22	-50%	-50%	NaN	NaN	NaN	NaN	NaN
23	-54%	-48%	NaN	NaN	NaN	NaN	NaN
24	-60%	-52%	NaN	NaN	NaN	NaN	NaN
25	-49%	-46%	NaN	NaN	NaN	NaN	NaN
26	-55%	-43%	NaN	NaN	NaN	NaN	NaN
27	-65%	-49%	Grisons	-85%	-15%	-35%	+14%
28	-34%	-39%	Schaffhausen	-63%	+30%	Not enough data for this date	+2%

Perform a join(), left_join() in particular:



```
In [91]: Texas_df = pd.read_csv(os.path.join("./csv","US", "2020-04-11_Texas.csv"))
         Cali_df = pd.read_csv(os.path.join("./csv","US", "2020-04-11_California.csv"))


         merged_inner = pd.merge(left=Texas_df, right=Cali_df, left_on='Grocery & pharmacy', right_on='Grocery & pharmacy')
```

```
In [98]: #Texas_df.head()
```

```
In [99]: #Cali_df.tail()
```

```
In [97]: merged_left = pd.merge(left=Texas_df, right=Cali_df, left_on='Grocery & pharmacy', right_on='Grocery & pharmacy', how='left')

         # The LEFT JOIN produces a complete set of records from DataFrame A (Left DataFrame), with the matching records (where available
         # here, Texas, California, Humboldt_County(CA) and Sonoma_County (CA) have Grocery & pharmacy value of -14%
         # same for Angelina_County(TX), Solano_County(CA), Stanislaus_County(CA), Sutter_County(CA) with value of -4%
         merged_left.head(15)
```

Out[97]:

	Territory_x	Retail & recreation_x	Grocery & pharmacy	Parks_x	Transit stations_x	Workplace_x	Residential_x	Territory_y	Retail & recreation_y	Parks_y	Transit stations_y	Workplace
0	Texas	-45%	-14%	-54%	-49%	-38%	+16%	California	-48%	-46%	-53%	-4(
1	Texas	-45%	-14%	-54%	-49%	-38%	+16%	Humboldt_County	-45%	-18%	-55%	-3
2	Texas	-45%	-14%	-54%	-49%	-38%	+16%	Sonoma_County	-50%	-59%	-49%	-4
3	Anderson_County	-35%	-2%	Not enough data for this date	Not enough data for this date	-24%	+16%	Kern_County	-39%	-24%	-37%	-2
4	Anderson_County	-35%	-2%	Not enough data for this date	Not enough data for this date	-24%	+16%	Tulare_County	-36%	-47%	-32%	-2
5	Andrews_County	-45%	-40%	Not enough data for this date	-13%	-36%	Not	NaN	NaN	NaN	NaN	Ni
6	Angelina_County	-46%	-4%	-20%	-38%	-29%	+13%	Solano_County	-43%	-18%	-39%	-3
7	Angelina_County	-46%	-4%	-20%	-38%	-29%	+13%	Stanislaus_County	-42%	-15%	-27%	-3
8	Angelina_County	-46%	-4%	-20%	-38%	-29%	+13%	Sutter_County	-37%	-19%	-20%	-3
9	Aransas_County	-45%	-31%	-57%	Not enough data for	-27%	+10%	NaN	NaN	NaN	NaN	Ni

8. (5 points) Export the merged DataFrame to
a. A csv file if your input data is in json documents.
b. **A json file** if your **input data is in csv files.**

```
In [109]: merged_left.to_json(os.path.join("./json_test","2020-04-11_Texas_and_Cali.json"))
```

jupyter 2020-04-11_Texas_and_Cali.json a few seconds ago

Logout

File Edit View Language JSON


```
{"Territory_x":
{"0":"Texas","1":"Texas","2":"Texas","3":"Anderson_County","4":"Anderson_County","5":"Andrews_County","6":"Angelina_County","7":"Angelina_County","8":"Angelina_County","9":"Aransas_County","10":"Archer_County","11":"Atascosa_County","12":"Atascosa_County","13":"Atascosa_County","14":"Austin_County","15":"Bailey_County","16":"Bandera_County","17":"Bastrop_County","18":"Bee_County","19":"Bell_County","20":"Bell_County","21":"Bell_County","22":"Bexar_County","23":"Bexar_County","24":"Blanco_County","25":"Bosque_County","26":"Bowie_County","27":"Bowie_County","28":"Bowie_County","29":"Brazoria_County","30":"Brazos_County","31":"Brewster_County","32":"Brooks_County","33":"Brooks_County","34":"Brown_County","35":"Burleson_County","36":"Burleson_County","37":"Burnet_County","38":"Burnet_County","39":"Burnet_County","40":"Caldwell_County","41":"Calhoun_County","42":"Callahan_County","43":"Cameron_County","44":"Camp_County","45":"Camp_County","46":"Carson_County","47":"Cass_County","48":"Cass_County","49":"Cass_County","50":"Castro_County","51":"Chambers_County","52":"Chambers_County","53":"Cherokee_County","54":"Childress_County","55":"Clay_County","56":"Coke_County","57":"Coleman_County","58":"Coleman_County","59":"Collin_County","60":"Colorado_County","61":"Colorado_County","62":"Comal_County","63":"Comanche_County","64":"Cooke_County","65":"Coryell_County","66":"Crane_County","67":"Crockett_County","68":"Crosby_County","69":"Culberson_County","70":"Dallam_County","71":"Dallas_County","72":"Dallas_County","73":"Dallas_County","74":"Dawson_County","75":"Dawson_County","76":"DeWitt_County","77":"Deaf_Smith_County","78":"Deaf_Smith_County","79":"Deaf_Smith_County","80":"Delta_County","81":"Denton_County","82":"Dimmit_County","83":"Donley_County","84":"Duval_County","85":"Eastland_County","86":"Eastland_County","87":"Eastland_County","88":"Ector_County","89":"Ector_County","90":"El_Paso_County","91":"El_Paso_County","92":"El_Paso_County","93":"Ellis_County","94":"Ellis_County","95":"Ellis_County","96":"Erath_County","97":"Falls_County","98":"Falls_County","99":"Fannin_County","100":"Fannin_County","101":"Fayette_County","102":"Floyd_County","103":"Fort_Bend_County","104":"Fort_Bend_County","105":"Franklin_County","106":"Franklin_County","107":"Freestone_County","108":"Freestone_County","109":"Frio_County","110":"Gaines_County","111":"Galveston_County","112":"Galveston_County","113":"Garza_County","114":"Gillespie_County","115":"Goliad_County","116":"Gonzales_County","117":"Gray_County","118":"Grayson_County","119":"Grayson_County","120":"Gregg_County","121":"Gregg_County","122":"Grimes_County","123":"Guadalupe_County","124":"Guadalupe_County","125":"Guadalupe_County","126":"Hale_County","127":"Hall_County","128":"Hamilton_County","129":"Hardeman_County","130":"Hardin_County","131":"Harris_County","132":"Harris_County","133":"Harrison_County","134":"Hartley_County","135":"Haskell_County","136":"Hays_County","137":"Henderson_County","138":"Hidalgo_County","139":"Hill_County","140":"Hockley_County","141":"Hood_County","142":"Hopkins_County","143":"Houston_County","144":"Howard_County","145":"Hunt_County","146":"Hunt_County","147":"Hunt_County","148":"Hutchinson_County","149":"Jack_County","150":"Jackson_County","151":"Jasper_County","152":"Jefferson_County","153":"Jim_Hogg_County","154":"Jim_Wells_County","155":"Jim_Wells_County","156":"Johnson_County","157":"Johnson_County","158":"Johnson_County","159":"Jones_County","160":"Jones_County","161":"Karnes_County","162":"Kaufman_County","163":"Kaufman_County","164":"Kaufman_County","165":"Kendall_County","166":"Kendall_County","167":"Kerr_County","168":"Kerr_County","169":"Kerr_County","170":"Kimble_County","171":"Kleberg_County","172":"La_Salle_County","173":"La_Salle_County","174":"Lamar_County","175":"Lamb_County","176":"Lamb_County","177":"Lampasas_County","178":"Lavaca_County","179":"Lee_County","180":"Leon_County","181":"Liberty_County","182":"Limestone_County","183":"Live_Oak_County","184":"Llano_County","185":"Loving_County","186":"Lubbock_County","187":"Lubbock_County","188":"Lubbock_County","189":"Lynn_County","190":"Madison_County","191":"Marion_County","192":"Marion_County","193":"Marion_County","194":"Martin_County","195":"Martin_County","196":"Martin_County","197":"Matagorda_County","198":"Matagorda_County","199":"Maverick_County","200":"McCulloch_County","201":"McLennan_County","202":"McLennan_County","203":"McLennan_County","204":"McMullen_County","205":"Medina_County","206":"Midland_County","207":"Milam_County","208":"Mitchell_County","209":"Montague_County","210":"Montgomery_County","211":"Moore_County","212":"Morris_County","213":"Morris_County","214":"Nacogdoches_County","215":"Nacogdoches_County","216":"Nacogdoches_County","217":"Navarro_County","218":"Newton_County","219":"Newton_County","220":"Nolan_County","221":"Nolan_County","222":"Nolan_County","223":"Nueces_County","224":"Nueces_County","225":"Ochiltree_County","226":"Ochiltree_County","227":"Ochiltree_County","228":"Oldham_County","229":"Orange_County","230":"Palo_Pinto_County","231":"Palo_Pinto_County","232":"Palo_Pinto_County","233":"Panola_County","234":"Panola_County","235":"Parker_County","236":"Parmer_County","237":"Parmer_County","238":"Parmer_County","239":"Pecos_County","240":"Polk_County","241":"Potter_County","242":"Potter_County","243":"Potter_County","244":"Presidio_County","245":"Rains_County","246":"Randall_County","247":"Randall_County","248":"Randall_County","249":"Reagan_County","250":"Real_County","251":"Red_River_County","252":"Reeves_County","253":"Reeves_County","254":"Refugio_County","255":"Robertson_County","256":"Rockwall_County","257":"Rockwall_County","258":"Runnels_County","259":"Rusk_County","260":"Sabine_County","261":"Sabine_County","262":"San_Augustine_County","263":"San_Augustine_County","264":"San_Augustine_County","265":"San_Jacinto_County","266":"San_Jacinto_County","267":"San_Patricio_County","268":"San_Patricio_Coun
```