Login:

Cd

copy all files under directory:

scp -r jl10919@dumbo.hpc.nyu.edu:~jl10919/hw2\_py.zip .

upload file to dumbo (put py code in /src)

scp template1.py jl10919@dumbo.hpc.nyu.edu:~jl10919/hw3

get data:

hfs -get /user/hc2660/hw3data/data\_cleaning.csv .

put data:

hfs -put data\_cleaning.csv

run

spark-submit --conf spark.pyspark.python=/share/apps/python/3.6.5/bin/python /home/jl10919/hw3/template1.py /user/hc2660/hw3data/data\_cleaning.csv

start pyspark:

module load python/gnu/3.6.5

module load spark/2.4.0

pyspark

hfs -rm -r hw3-task2-\*-1.out

hfs -rm -r hw3-task2-e.out

hfs -rm -r hw3-task2-f.out

hfs -getmerge hw3-task1-a.out hw3-task1-a.out

hfs -getmerge hw3-task1-b.out hw3-task1-b.out

hfs -getmerge hw3-task1-c.out hw3-task1-c.out

hfs -getmerge hw3-task1-d1.out hw3-task1-d1.out

hfs -getmerge hw3-task1-d2.out hw3-task1-d2.out

hfs -getmerge hw3-task2-c.out hw3-task2-c.out

hfs -getmerge hw3-task2-d.out hw3-task2-d.out

hfs -getmerge hw3-task2-e.out hw3-task2-e.out

hfs -getmerge hw3-task2-f.out hw3-task2-f.out

hfs -getmerge hw3-task2-h.out hw3-task2-h.out

hfs -getmerge 2f-all.out 2f-all.out

hfs -getmerge 2e-all.out 2e-all.out

hfs -getmerge 2f-1.out 2f-1.out

hfs -getmerge 2f-2.out 2f-2.out

hfs -getmerge 2f-3.out 2f-3.out

hfs -getmerge 2f-4.out 2f-4.out

hfs -getmerge 2f-5.out 2f-5.out

hfs -getmerge hw3-task2-c-1.out hw3-task2-c-1.out

hfs -getmerge hw3-task2-d-1.out hw3-task2-d-1.out

hfs -getmerge hw3-task2-e-1.out hw3-task2-e-1.out

hfs -getmerge hw3-task2-f-1.out hw3-task2-f-1.out

vim hw3-task2-c-1.out

diff /home/jl10919/hw3/hw3-task1-c.out hw3-task1-c.out

debug:

可以进入pyspark

line1 = sc.textFile('/user/hc2660/hw3data/data\_cleaning.csv')

一句句输入命令 用

for row in line1.take(10): print(row)

看RDD是否对

sql-debug

from pyspark.sql import SparkSession  
from pyspark.sql.functions import \*  
from pyspark import SparkContext

parking\_df = spark.read.format('csv').options(header = 'true', inferschema = 'true').load("data\_cleaning.csv")

parking\_df.createOrReplaceTempView("df")

df.show(5)

A picture containing screenshot, monitor, screen

Description automatically generated

Task1-a

SQL Constraints:

<https://www.w3schools.com/sql/sql_constraints.asp>

输出表中行数，return int： 用.count()

repeat\_sn.count()

Task 1-b

task1b\_result.coalesce(1).rdd.map(lambda x: x[0] + ',' + str(x[1])).saveAsTextFile("hw3-task1-b.out")

看输出文件是否包含了999:

grep -i "999" hw3-task1-b.out

>>> 999,274

Task1-c

使用函数去处理select后的表单

name = 'plate\_type'  
udf = UserDefinedFunction(lambda x: 'NULL' if x== '999' else x)

task1c\_result = task1c\_tmp.select(\*[udf(column).alias(name) if column == name else column for column in task1c\_tmp.columns])

Task1-d

A screenshot of a cell phone

Description automatically generated

空值是null

**PySpark︱DataFrame操作指南：增/删/改/查/合并/统计与数据处理**

https://blog.csdn.net/sinat\_26917383/article/details/80500349

Task2

A screenshot of a cell phone

Description automatically generated

Get rdd

import sys  
from pyspark.sql import SparkSession  
from pyspark.sql.functions import \*  
from pyspark import SparkContext

from csv import reader

sc = SparkContext()

line1 = sc.textFile ('data\_cleaning.csv')

line1 = line1.mapPartitions(lambda x: reader(x))

plate\_id\_rdd = line1.map(lambda x: x[-7])

for row in plate\_id\_rdd.take(10): print(row)

street\_name\_rdd = line1.map(lambda x:x[-5])

for row in street\_name\_rdd.take(10): print(row)

>>> for row in plate\_id\_rdd.take(10): print(row)

...

plate\_type

PAS

PAS

COM

PAS

COM

COM

PAS

PAS

PAS

>>> for row in street\_name\_rdd.take(10): print(row)

...

street\_name

N/S WARREN ST

PHILPS

POST AVE

C/O E 53

W 43 STREET

W 30 ST

DYCKMAN ST

WHITE PLAINS ROAD

OCEAN AVE

Python decode() 方法以 *encoding* 指定的编码格式解码字符串。默认编码为字符串编码

str.decode(encoding='UTF-8',errors='strict')

* encoding -- 要使用的编码，如"UTF-8"。
* errors -- 设置不同错误的处理方案。默认为 'strict',意为编码错误引起一个UnicodeError。 其他可能得值有 'ignore', 'replace', 'xmlcharrefreplace', 'backslashreplace' 以及通过 codecs.register\_error() 注册的任何值。

str = "this is string example....wow!!!"; str = str.encode('base64','strict'); print "Encoded String: " + str; print "Decoded String: " + str.decode('base64','strict')

Encoded String: dGhpcyBpcyBzdHJpbmcgZXhhbXBsZS4uLi53b3chISE=

Decoded String: this is string example....wow!!!

python中，我们使用decode()和encode()来进行解码和编码

在python中，使用unicode类型作为编码的基础类型。即

     decode              encode

str ---------> unicode --------->str

unicodedata.**normalize**(*form*, *unistr*)

Return the normal form *form* for the Unicode string *unistr*. Valid values for *form* are ‘NFC’, ‘NFKC’, ‘NFD’, and ‘NFKD’.

The normal form KD (NFKD) will apply the compatibility decomposition, i.e. replace all compatibility characters with their equivalents.

data=[ ('fkorstyz', 'Krzysztof'), ('fkorstyz', 'Kryzysztof'), ('fkorstyz', 'Krzystof')]

distData = sc.parallelize(data)

for row in distData: print(row)

[jl10919@login-1-1 out]$ head hw3-task2-c.out

999,9,9,9

app,A,P,P

cmb,C,M,B

com,C,O,M

csp,C,S,P

ham,H,A,M

his,H,I,S

mot,M,O,T

nys,N,Y,S

omf,O,M,F

[jl10919@login-1-1 out]$ head hw3-task2-d.out

9,9,9,9

ahm,H,A,M

ap,A,P,P

aps,P,A,S

bcm,C,M,B

cmo,C,O,M

cor,O,R,C

cps,C,S,P

dps,P,S,D

fmo,O,M,F

[jl10919@login-1-1 out]$ head hw3-task2-e.out

,

06 merrick,0,6, ,M,E,R,R,I,C,K

08 jfk tern van wyck,T,E,R,N, ,0,8, ,V,A,N, ,W,Y,C,K, ,J,F,K

1,1

1 10,1, ,1,0

1 10 42 s t,1, ,1, ,1,0, ,4,2, ,S, ,T

1 17 street west,W,E,S,T, ,1, ,1,7, ,S,T,R,E,E,T

1 airport hanger lga,L,G,A, ,A,I,R,P,O,R,T, ,H,A,N,G,E,R, ,1

1 american hanger,A,M,E,R,I,C,A,N, ,H,A,N,G,E,R, ,1

1 ave,1, ,A,V,E

[jl10919@login-1-1 out]$ head hw3-task2-f.out

,

01,1, ,1,0

01235fmort,5,0, ,F,T, ,F,R,O,M, ,2,1,3

01238eforst,3,0, ,F,E,E,T, ,E, ,O,F, ,1,8,2, ,S,T,R

01248aefhorstw,R,E,A,R, ,O,F, ,4,2,0, ,W, ,1,8,T,H, ,S

01249orstw,R,/,O, ,4,2,0, ,W, ,1,9, ,S,T

0124st,1, ,1, ,1,0, ,4,2, ,S, ,T

0125orstw,R,/,O, ,1,5,0, ,W, ,2,2,5, ,S,T

0125orw,R,/,O, ,W, ,1,5,0, ,W, ,2,2,5

0126bdefiost,6,2, ,S,T, ,S,I,D,E, ,O,F, ,2,0,1, ,B,E

可以打20行

直接reducebykey(add)的话是直接x[1]为一个元素



再map成[x[1]]之后可以，注意不可以是list(x[1]),

Eg. List(‘abc’) = [‘a’,’b’,’c’],但[‘abc’]=[‘abc’]

.map(lambda x: (x[0],[x[1]])).reduceByKey(lambda x,y: x+y)

A picture containing white

Description automatically generated

用distinct去重

当n=3，street name:

A screenshot of a cell phone

Description automatically generated

n-1

6867

('01aehtv', ['10TH AVE', '110th Ave', '100th Ave', '10th Ave'])

n-2

8888

n-3

8897

n-4

8823 clusters

n-5

8539

('huntintantavntspointpoinspoitavetspounts', ['Hunts Point Ave', 'HUNTS POINT AVE', 'HUNTSPOINT AVE']

对于fingerprint 可以cluster掉大小写不同，重复输入同样字段的情况，

但是对于一些比较相似的情况不能cluster 比如

('108 e ss', ['S/S E 108'])

('108 e ss st', ['S/S E 108 ST'])

('108 e st', ['E 108 ST'])

('108 e street', ['E 108 STREET'])

('108 east st', ['EAST 108 ST'])

('108 east street', ['EAST 108 STREET'])

对于n-gram,同样可以去掉大小写，重复的情况，但是可以去掉没有空格时重复的情况，如

('00041044aiinmansnysttny1yn', ['MAIN ST NY NY 10044', 'MAIN ST NYNY 10044'])

n-gram fingerprint：一样key

('00041044aiinmansnysttny1yn', 'MAIN ST NY NY 10044')

('00041044aiinmansnysttny1yn', 'MAIN ST NYNY 10044')

Fingerprint：不一样key

('10044 main ny st', 'MAIN ST NY NY 10044')

('10044 main nyny st', 'MAIN ST NYNY 10044')

当n=4时

('129t29th9thse129thst', ['E 129th St'])

('129s29ste129', ['E 129 ST'])

('12av2ave', ['12 AVE'])

('12av2aveavenenuevenu', ['12 AVENUE'])

太多被归到null key 不合适

('', ['E 51', '', '9 ST', 'CPW', 'W 53', 'E 96', 'E 94', '6 ST', '8 ST', '5 ST', 'E 80', 'AV', 'E 2', 'CIA', 'E 34', 'E 37', 'E 52', 'E 49', '1 10', 'E 70', '252', 'W 12', 'WEA', 'W 83', 'W 75', 'GCP', '2ND', '45', 'WPR', 'LEE', '158', '217', 'E 44', 'E 48', '98', '112', '1', 'RSD', '1 ST', '1ST', 'E 86', 'E 84', '174', 'O P', 'ACP', '9ST', 'E 77', 'E 89', '70', 'W 73', 'W 78', 'W 68', 'W 96', '4 ST', '7 ST', 'A/C', 'C/O', '2 ST', '5TH', 'E 87', 'E 23', 'E 21', 'E 46', 'E 24', 'E 69', 'RBB', 'E 31', 'E 32', 'E 35', 'E 55', 'E 72', 'AVE', 'E 36', '195', 'W 10', 'W 77', 'R/S', 'E 54', '105', '99', 'E 60', 'E 79', 'E 12', '3RD', 'E 7', 'BBA', 'E', 'E 4', '76', '62', 'LEX', 'AV M', 'E 38', 'BPE', '8 ST'])

观查可得 很多地址缩写导致不一样

<http://maf.directory/zp4/abbrev.html>

east = e

west = w

north = n

south = s

street=st

Avenue = ave = av

Number+th = number

Place = pl

Road = rd

Lane = la

Lane = ln

Plaza = plz

Drive = dr

Square = sq

Parkway = pkwy

('purdy', ['PURDY'])

('purdyrdysturdys', ['Purdy St', 'PURDY ST'])

<http://bcdcspatial.blogspot.com/2012/09/normalize-to-usps-street-abbreviations.html>

H:

所有街道名称改成缩写。

Eg.可以去掉影响的符号和大小写 ST.JOHNS = St Johns

同时place = pl

hnjonsohplspsttj,ST.JOHNS PL,ST . JOHNS PL,St Johns Pl,ST JOHNS PL,ST JOHN S PLACE,ST .JOHNS PL

eg. Road -> rd 而且连在一起没有分开的词也可以辨别到 (FRESHPOND=FRESH POND)

dresfrhpndonpordresh,FRESH POND RD,FRESHPOND RD,FRESH POND ROAD,FRESHPOND ROAD,Fresh Pond Rd

所有东南西北改成缩写

Eg. East = E

emhokwlumoolospkshupwy,E MOSHOLU PKWY,E Mosholu Pky,EAST MOSHOLU PARKWAY

所有numberth改成number

Eg, 69 = 69th

699sst,69 STREET,69th St,69 ST,69TH STREET

6860 cluster

对plate id

Fingerprint():

>>> task2c\_result.filter(lambda x: len(x[1])>=2).collect()

[('jnp981', ['JNP981&', 'JNP981']), ('l21687', ['L.21687', 'L21687']), ('ns', ['N/S', 'NS'])]

n-gram fingerprint():

[('088088e8fegf', ['GFE8808', 'GFE8088']), ('155155gvt5vt', ['GVT5515', 'GVT5155']), ('16216887l2', ['L.21687', 'L21687']), ('17576571777ct6', ['T657177C', 'T657717C']), ('177177gzr7zr', ['GZR7177', 'GZR7717']), ('222662b2gxxb', ['GXB2262', 'GXB2622']), ('222772e2gzze', ['GZE2272', 'GZE2722']), ('222882cthct2', ['HCT2822', 'HCT2282']), ('222882d2fdhf', ['HFD2282', 'HFD2822']), ('222882gzh2zh', ['GZH2282', 'GZH2822']), ('222992csgcs2', ['GCS2292', 'GCS2922']), ('333773gkkuu3', ['GKU3373', 'GKU3733']), ('444554dthdt4', ['HDT4454', 'HDT4544']), ('47646776777ct6', ['T676477C', 'T647767C']), ('477477gssww7', ['GSW7747', 'GSW7477']), ('5055axhax5', ['HAX550', 'HAX5550']), ('677997999ct6', ['T679979C', 'T679799C']), ('777997gpk7pk', ['GPK7977', 'GPK7797']), ('8198jnnpp9', ['JNP981&', 'JNP981']), ('888998gtp8tp', ['GTP8988', 'GTP8898']), ('ns', ['N/S', 'NS'])]

debug

module load python/gnu/3.6.5

module load spark/2.4.0

pyspark

import re  
import sys  
from pyspark.sql import SparkSession  
from pyspark.sql.functions import \*  
from pyspark import SparkContext  
from csv import reader  
from operator import add

sc = spark.sparkContext

line1 = sc.textFile('/user/hc2660/hw3data/data\_cleaning.csv', 1)

head = "summons\_number"  
def delhead(line):  
 if head in line:  
 return 1  
 return 0

line1 = line1.mapPartitions(lambda x: reader(x))  
line1 = line1.filter(lambda line: delhead(line) != 1)

plate\_id\_rdd = line1.map(lambda x: x[-8])  
street\_name\_rdd = line1.map(lambda x:x[-5])

print(street\_name\_rdd.distinct().count())

print(street\_name\_rdd.count())

def fingerprint(value):  
 # remove leading and trailing whitespace  
 # change all characters to their lowercase representation  
 s1 = value.strip().lower()  
 # remove all punctuation and control characters  
 s2 = s1.translate(str.maketrans('', '', string.punctuation))  
 # normalize extended western characters to their ASCII representation (for example "gödel" → "godel")  
 s3 = unicodedata.normalize(u'NFKD', s2).encode('ascii', 'ignore').decode('utf8')  
 # split the string into whitespace-separated tokens  
 s4 = s3.split()  
 # sort the tokens and remove duplicates  
 s5 = sorted(list(set(s4)))  
 # join the tokens back together  
 s6 = ' '.join(s5)  
 key = s6  
 return (key,value)

def ngram\_fingerprint(value, n = 1): #choose your n-gram  
 #change all characters to their lowercase representation  
 s1 = value.strip().lower()  
 #remove all punctuation, control characters and whitespace  
 s2 = s1.translate(str.maketrans('', '', string.punctuation))  
 s3 = s2.replace(" ", "")  
 #normalize extended western characters to their ASCII representation  
 s4 = unicodedata.normalize(u'NFKD', s3).encode('ascii', 'ignore').decode('utf8')  
 #obtain all the string n-grams  
 s5 = [s4[i:i+n] for i in range(len(s4)-n+1)]  
 #sort the n-grams and remove duplicates  
 s6= sorted(list(set(s5)))  
 #join the sorted n-grams back together  
 s7 = ''.join(s6)  
 result = s7  
 return (result,value)

task2c\_result = plate\_id\_rdd.map(lambda line: fingerprint(str(line))).distinct().map(lambda x: (x[0],[x[1]])).reduceByKey(lambda x,y: x+y).sortByKey()

task2d\_result = plate\_id\_rdd.map(lambda line: ngram\_fingerprint(str(line),n=2)).distinct().map(lambda x: (x[0],[x[1]])).reduceByKey(lambda x,y: x+y).sortByKey()

t2c=task2c\_result.map(lambda x: x[0] + ',' + ','.join(x[1]))   
t2d=task2d\_result.map(lambda x: x[0] + ',' + ','.join(x[1]))

task2d\_result.filter(lambda x: len(x[1])>=2).collect()

>>> task2d\_result.filter(lambda x: len(x[1])>=2).collect()

[('088088e8fegf', ['GFE8808', 'GFE8088']), ('155155gvt5vt', ['GVT5515', 'GVT5155']), ('16216887l2', ['L.21687', 'L21687']), ('17576571777ct6', ['T657177C', 'T657717C']), ('177177gzr7zr', ['GZR7177', 'GZR7717']), ('222662b2gxxb', ['GXB2262', 'GXB2622']), ('222772e2gzze', ['GZE2272', 'GZE2722']), ('222882cthct2', ['HCT2822', 'HCT2282']), ('222882d2fdhf', ['HFD2282', 'HFD2822']), ('222882gzh2zh', ['GZH2282', 'GZH2822']), ('222992csgcs2', ['GCS2292', 'GCS2922']), ('333773gkkuu3', ['GKU3373', 'GKU3733']), ('444554dthdt4', ['HDT4454', 'HDT4544']), ('47646776777ct6', ['T676477C', 'T647767C']), ('477477gssww7', ['GSW7747', 'GSW7477']), ('5055axhax5', ['HAX550', 'HAX5550']), ('677997999ct6', ['T679979C', 'T679799C']), ('777997gpk7pk', ['GPK7977', 'GPK7797']), ('8198jnnpp9', ['JNP981&', 'JNP981']), ('888998gtp8tp', ['GTP8988', 'GTP8898']), ('ns', ['N/S', 'NS'])]

task2c\_result.filter(lambda x: len(x[1])>=2).collect()

>>> task2c\_result.filter(lambda x: len(x[1])>=2).collect()

[('jnp981', ['JNP981&', 'JNP981']), ('l21687', ['L.21687', 'L21687']), ('ns', ['N/S', 'NS'])]

def normalizeStreetSuffixes(inputValue):  
 *'''  
 Use common abbreviations -> USPS standardized abbreviation to replace common street suffixes  
  
 Obtains list from https://www.usps.com/send/official-abbreviations.htm  
 '''* usps\_street\_abbreviations = {'trpk': 'tpke', 'forges': 'frgs', 'bypas': 'byp', 'mnr': 'mnr', 'viaduct': 'via',  
 'mnt': 'mt',  
 'lndng': 'lndg', 'vill': 'vlg', 'aly': 'aly', 'mill': 'ml', 'pts': 'pts',  
 'centers': 'ctrs', 'row': 'row', 'cnter': 'ctr',  
 'hrbor': 'hbr', 'tr': 'trl', 'lndg': 'lndg', 'passage': 'psge', 'walks': 'walk',  
 'frks': 'frks', 'crest': 'crst', 'meadows': 'mdws',  
 'freewy': 'fwy', 'garden': 'gdn', 'bluffs': 'blfs', 'vlg': 'vlg', 'vly': 'vly',  
 'fall': 'fall', 'trk': 'trak', 'squares': 'sqs',  
 'trl': 'trl', 'harbor': 'hbr', 'frry': 'fry', 'div': 'dv', 'straven': 'stra',  
 'cmp': 'cp', 'grdns': 'gdns', 'villg': 'vlg',  
 'meadow': 'mdw', 'trails': 'trl', 'streets': 'sts', 'prairie': 'pr', 'hts': 'hts',  
 'crescent': 'cres', 'pass': 'pass',  
 'ter': 'ter', 'port': 'prt', 'bluf': 'blf', 'avnue': 'ave', 'lights': 'lgts',  
 'rpds': 'rpds', 'harbors': 'hbrs',  
 'mews': 'mews', 'lodg': 'ldg', 'plz': 'plz', 'tracks': 'trak', 'path': 'path',  
 'pkway': 'pkwy', 'gln': 'gln',  
 'bot': 'btm', 'drv': 'dr', 'rdg': 'rdg', 'fwy': 'fwy', 'hbr': 'hbr', 'via': 'via',  
 'divide': 'dv', 'inlt': 'inlt',  
 'fords': 'frds', 'avenu': 'ave', 'vis': 'vis', 'brk': 'brk', 'rivr': 'riv',  
 'oval': 'oval', 'gateway': 'gtwy',  
 'stream': 'strm', 'bayoo': 'byu', 'msn': 'msn', 'knoll': 'knl',  
 'expressway': 'expy', 'sprng': 'spg',  
 'flat': 'flt', 'holw': 'holw', 'grden': 'gdn', 'trail': 'trl', 'jctns': 'jcts',  
 'rdgs': 'rdgs',  
 'tunnel': 'tunl', 'ml': 'ml', 'fls': 'fls', 'flt': 'flt', 'lks': 'lks', 'mt': 'mt',  
 'groves': 'grvs',  
 'vally': 'vly', 'ferry': 'fry', 'parkway': 'pkwy', 'radiel': 'radl',  
 'strvnue': 'stra', 'fld': 'fld',  
 'overpass': 'opas', 'plaza': 'plz', 'estate': 'est', 'mntn': 'mtn', 'lock': 'lck',  
 'orchrd': 'orch',  
 'strvn': 'stra', 'locks': 'lcks', 'bend': 'bnd', 'kys': 'kys', 'junctions': 'jcts',  
 'mountin': 'mtn',  
 'burgs': 'bgs', 'pine': 'pne', 'ldge': 'ldg', 'causway': 'cswy', 'spg': 'spg',  
 'beach': 'bch', 'ft': 'ft',  
 'crse': 'crse', 'motorway': 'mtwy', 'bluff': 'blf', 'court': 'ct', 'grov': 'grv',  
 'sprngs': 'spgs',  
 'ovl': 'oval', 'villag': 'vlg', 'vdct': 'via', 'neck': 'nck', 'orchard': 'orch',  
 'light': 'lgt',  
 'sq': 'sq', 'pkwy': 'pkwy', 'shore': 'shr', 'green': 'grn', 'strm': 'strm',  
 'islnd': 'is',  
 'turnpike': 'tpke', 'stra': 'stra', 'mission': 'msn', 'spngs': 'spgs',  
 'course': 'crse',  
 'trafficway': 'trfy', 'terrace': 'ter', 'hway': 'hwy', 'avenue': 'ave',  
 'glen': 'gln',  
 'boul': 'blvd', 'inlet': 'inlt', 'la': 'ln', 'ln': 'ln', 'frst': 'frst',  
 'clf': 'clf',  
 'cres': 'cres', 'brook': 'brk', 'lk': 'lk', 'byp': 'byp', 'shoar': 'shr',  
 'bypass': 'byp',  
 'mtin': 'mtn', 'ally': 'aly', 'forest': 'frst', 'junction': 'jct', 'views': 'vws',  
 'wells': 'wls', 'cen': 'ctr',  
 'exts': 'exts', 'crt': 'ct', 'corners': 'cors', 'trak': 'trak', 'frway': 'fwy',  
 'prarie': 'pr', 'crossing': 'xing',  
 'extn': 'ext', 'cliffs': 'clfs', 'manors': 'mnrs', 'ports': 'prts',  
 'gatewy': 'gtwy', 'square': 'sq', 'hls': 'hls',  
 'harb': 'hbr', 'loops': 'loop', 'mdw': 'mdw', 'smt': 'smt', 'rd': 'rd',  
 'hill': 'hl', 'blf': 'blf',  
 'highway': 'hwy', 'walk': 'walk', 'clfs': 'clfs', 'brooks': 'brks', 'brnch': 'br',  
 'aven': 'ave',  
 'shores': 'shrs', 'iss': 'iss', 'route': 'rte', 'wls': 'wls', 'place': 'pl',  
 'sumit': 'smt', 'pines': 'pnes',  
 'trks': 'trak', 'shoal': 'shl', 'strt': 'st', 'frwy': 'fwy', 'heights': 'hts',  
 'ranches': 'rnch',  
 'boulevard': 'blvd', 'extnsn': 'ext', 'mdws': 'mdws', 'hollows': 'holw',  
 'vsta': 'vis', 'plains': 'plns',  
 'station': 'sta', 'circl': 'cir', 'mntns': 'mtns', 'prts': 'prts', 'shls': 'shls',  
 'villages': 'vlgs',  
 'park': 'park', 'nck': 'nck', 'rst': 'rst', 'haven': 'hvn', 'turnpk': 'tpke',  
 'expy': 'expy', 'sta': 'sta',  
 'expr': 'expy', 'stn': 'sta', 'expw': 'expy', 'street': 'st', 'str': 'st',  
 'spurs': 'spur', 'crecent': 'cres',  
 'rad': 'radl', 'ranch': 'rnch', 'well': 'wl', 'shoals': 'shls', 'alley': 'aly',  
 'plza': 'plz', 'medows': 'mdws',  
 'allee': 'aly', 'knls': 'knls', 'ests': 'ests', 'st': 'st', 'anx': 'anx',  
 'havn': 'hvn', 'paths': 'path', 'bypa': 'byp',  
 'spgs': 'spgs', 'mills': 'mls', 'parks': 'park', 'byps': 'byp', 'flts': 'flts',  
 'tunnels': 'tunl', 'club': 'clb', 'sqrs': 'sqs',  
 'hllw': 'holw', 'manor': 'mnr', 'centre': 'ctr', 'track': 'trak', 'hgts': 'hts',  
 'rnch': 'rnch', 'crcle': 'cir', 'falls': 'fls',  
 'landing': 'lndg', 'plaines': 'plns', 'viadct': 'via', 'gdns': 'gdns',  
 'gtwy': 'gtwy', 'grove': 'grv', 'camp': 'cp', 'tpk': 'tpke',  
 'drive': 'dr', 'freeway': 'fwy', 'ext': 'ext', 'points': 'pts', 'exp': 'expy',  
 'ky': 'ky', 'courts': 'cts', 'pky': 'pkwy', 'corner': 'cor',  
 'crssing': 'xing', 'mnrs': 'mnrs', 'unions': 'uns', 'cyn': 'cyn', 'lodge': 'ldg',  
 'trfy': 'trfy', 'circle': 'cir', 'bridge': 'brg',  
 'dl': 'dl', 'dm': 'dm', 'express': 'expy', 'tunls': 'tunl', 'dv': 'dv', 'dr': 'dr',  
 'shr': 'shr', 'knolls': 'knls', 'greens': 'grns',  
 'tunel': 'tunl', 'fields': 'flds', 'common': 'cmn', 'orch': 'orch', 'crk': 'crk',  
 'river': 'riv', 'shl': 'shl', 'view': 'vw',  
 'crsent': 'cres', 'rnchs': 'rnch', 'crscnt': 'cres', 'arc': 'arc', 'btm': 'btm',  
 'blvd': 'blvd', 'ways': 'ways', 'radl': 'radl',  
 'rdge': 'rdg', 'causeway': 'cswy', 'parkwy': 'pkwy', 'juncton': 'jct',  
 'statn': 'sta', 'gardn': 'gdn', 'mntain': 'mtn',  
 'crssng': 'xing', 'rapid': 'rpd', 'key': 'ky', 'plns': 'plns', 'wy': 'way',  
 'cor': 'cor', 'ramp': 'ramp', 'throughway': 'trwy',  
 'estates': 'ests', 'ck': 'crk', 'loaf': 'lf', 'hvn': 'hvn', 'wall': 'wall',  
 'hollow': 'holw', 'canyon': 'cyn', 'clb': 'clb',  
 'cswy': 'cswy', 'village': 'vlg', 'cr': 'crk', 'trce': 'trce', 'cp': 'cp',  
 'cv': 'cv', 'ct': 'cts', 'pr': 'pr', 'frg': 'frg',  
 'jction': 'jct', 'pt': 'pt', 'mssn': 'msn', 'frk': 'frk', 'brdge': 'brg',  
 'cent': 'ctr', 'spur': 'spur', 'frt': 'ft', 'pk': 'park',  
 'fry': 'fry', 'pl': 'pl', 'lanes': 'ln', 'gtway': 'gtwy', 'prk': 'park',  
 'vws': 'vws', 'stravenue': 'stra', 'lgt': 'lgt',  
 'hiway': 'hwy', 'ctr': 'ctr', 'prt': 'prt', 'ville': 'vl', 'plain': 'pln',  
 'mount': 'mt', 'mls': 'mls', 'loop': 'loop',  
 'riv': 'riv', 'centr': 'ctr', 'is': 'is', 'prr': 'pr', 'vl': 'vl', 'avn': 'ave',  
 'vw': 'vw', 'ave': 'ave', 'spng': 'spg',  
 'hiwy': 'hwy', 'dam': 'dm', 'isle': 'isle', 'crcl': 'cir', 'sqre': 'sq',  
 'jct': 'jct', 'jctn': 'jct', 'mountain': 'mtn',  
 'keys': 'kys', 'parkways': 'pkwy', 'drives': 'drs', 'tunl': 'tunl', 'jcts': 'jcts',  
 'knl': 'knl', 'center': 'ctr',  
 'driv': 'dr', 'tpke': 'tpke', 'sumitt': 'smt', 'canyn': 'cyn', 'ldg': 'ldg',  
 'harbr': 'hbr', 'rest': 'rst', 'shoars': 'shrs',  
 'vist': 'vis', 'gdn': 'gdn', 'islnds': 'iss', 'hills': 'hls', 'cresent': 'cres',  
 'point': 'pt', 'lake': 'lk', 'vlly': 'vly',  
 'strav': 'stra', 'crossroad': 'xrd', 'bnd': 'bnd', 'strave': 'stra',  
 'stravn': 'stra', 'knol': 'knl', 'vlgs': 'vlgs',  
 'forge': 'frg', 'cntr': 'ctr', 'cape': 'cpe', 'height': 'hts', 'lck': 'lck',  
 'highwy': 'hwy', 'trnpk': 'tpke', 'rpd': 'rpd',  
 'boulv': 'blvd', 'circles': 'cirs', 'valleys': 'vlys', 'vst': 'vis',  
 'creek': 'crk', 'mall': 'mall', 'spring': 'spg',  
 'brg': 'brg', 'holws': 'holw', 'lf': 'lf', 'est': 'est', 'xing': 'xing',  
 'trace': 'trce', 'bottom': 'btm',  
 'streme': 'strm', 'isles': 'isle', 'circ': 'cir', 'forks': 'frks', 'burg': 'bg',  
 'run': 'run', 'trls': 'trl',  
 'radial': 'radl', 'lakes': 'lks', 'rue': 'rue', 'vlys': 'vlys', 'br': 'br',  
 'cors': 'cors', 'pln': 'pln',  
 'pike': 'pike', 'extension': 'ext', 'island': 'is', 'frd': 'frd', 'lcks': 'lcks',  
 'terr': 'ter',  
 'union': 'un', 'extensions': 'exts', 'pkwys': 'pkwy', 'islands': 'iss',  
 'road': 'rd', 'shrs': 'shrs',  
 'roads': 'rds', 'glens': 'glns', 'springs': 'spgs', 'missn': 'msn', 'ridge': 'rdg',  
 'arcade': 'arc',  
 'bayou': 'byu', 'crsnt': 'cres', 'junctn': 'jct', 'way': 'way', 'valley': 'vly',  
 'fork': 'frk',  
 'mountains': 'mtns', 'bottm': 'btm', 'forg': 'frg', 'ht': 'hts', 'ford': 'frd',  
 'hl': 'hl',  
 'grdn': 'gdn', 'fort': 'ft', 'traces': 'trce', 'cnyn': 'cyn', 'cir': 'cir',  
 'un': 'un', 'mtn': 'mtn',  
 'flats': 'flts', 'anex': 'anx', 'gatway': 'gtwy', 'rapids': 'rpds',  
 'villiage': 'vlg', 'flds': 'flds',  
 'coves': 'cvs', 'rvr': 'riv', 'av': 'ave', 'pikes': 'pike', 'grv': 'grv',  
 'vista': 'vis', 'pnes': 'pnes',  
 'forests': 'frst', 'field': 'fld', 'branch': 'br', 'grn': 'grn', 'dale': 'dl',  
 'rds': 'rds', 'annex': 'anx',  
 'sqr': 'sq', 'cove': 'cv', 'squ': 'sq', 'skyway': 'skwy', 'ridges': 'rdgs',  
 'hwy': 'hwy', 'tunnl': 'tunl',  
 'underpass': 'upas', 'cliff': 'clf', 'lane': 'ln', 'land': 'land', 'bch': 'bch',  
 'dvd': 'dv', 'curve': 'curv',  
 'cpe': 'cpe', 'summit': 'smt', 'gardens': 'gdns'}  
 orient\_abbreviations = {'west': 'w', 'east': 'e', 'north': 'n', 'south':'s'}  
 words = inputValue.split()  
 for w in words:  
 # change all the street special noun to the abbreviation. eg. change street to st  
 if w in usps\_street\_abbreviations.keys():  
 inputValue = inputValue.replace(w, usps\_street\_abbreviations[w])  
 # change all cardinal direction to abbreviation. eg. change west to w  
 if w in orient\_abbreviations.keys():  
 inputValue = inputValue.replace(w, orient\_abbreviations[w])  
 # change the ordinal number to cardinal number. eg. change 100th to 100  
 if re.search(r'[0-9]+th', w, re.I):  
 # delte the "th"  
 rep\_w = re.sub(r'th', "", w, re.I)  
 inputValue = inputValue.replace(w, rep\_w)  
 return inputValue

def abbre\_fingerprint(value):  
 # remove leading and trailing whitespace  
 # change all characters to their lowercase representation  
 s1 = value.strip().lower()  
 # remove all punctuation and control characters  
 s2 = s1.translate(str.maketrans('', '', string.punctuation))  
 # normalize extended western characters to their ASCII representation (for example "gödel" → "godel")  
 s3 = unicodedata.normalize(u'NFKD', s2).encode('ascii', 'ignore').decode('utf8')  
 # change to the aabbreviation  
 s3 = normalizeStreetSuffixes(s3)  
 # split the string into whitespace-separated tokens  
 s4 = s3.split()  
 # sort the tokens and remove duplicates  
 s5 = sorted(list(set(s4)))  
 # join the tokens back together  
 s6 = ' '.join(s5)  
 key = s6  
 return (key,value)

task2h\_result = street\_name\_rdd.map(lambda line: abbre\_fingerprint(str(line))).distinct().map(lambda x: (x[0],[x[1]])).reduceByKey(lambda x,y: x+y).sortByKey()

task2h\_result.filter(lambda x: len(x[1])>=8).collect()

>>> task2h\_result.filter(lambda x: len(x[1])>=8).collect()

[('170 e st', ['EAST 170 STREET', 'E 170 ST', 'E 170TH ST', 'E 170 STREET', 'E 170TH STREET', 'EAST 170TH STREET', 'EAST 170 ST', 'E 170th St']), ('18 ave', ['18 AVE', '18TH AVE', '18 AV', '18 AVENUE', '18TH AVENUE', '18TH AVENUE', '18 AVE+', '18th Ave']), ('4 e st', ['E 4TH ST', 'E 4 ST', 'EAST 4 STREET', 'EAST 4TH STREET', 'EAST 4 ST', 'E 4 STREET', 'EAST 4TH ST', 'E 4th St']), ('8 st w', ['W 8TH STREET', 'WEST 8TH ST', 'W 8 ST', 'WEST 8 ST', 'W 8TH ST', 'WEST 8 STREET', 'WEST 8TH STREET', 'W 8th St'])]

print(task2h\_result.count())

//6938

t2e\_tmp = street\_name\_rdd.map(lambda line: fingerprint(str(line))).distinct().map(lambda x: (x[0],[x[1]])).reduceByKey(lambda x,y: x+y).sortByKey()

t2e\_tmp.filter(lambda x: len(x[1])>=4).collect()

[('blvd springfield', ['SPRINGFIELD BLVD', 'SPRINGFIELD BLVD', ';SPRINGFIELD BLVD', 'Springfield Blvd']), ('johns pl st', ['ST JOHNS PL', 'ST . JOHNS PL', 'ST .JOHNS PL', 'St Johns Pl'])]

t2f\_tmp = street\_name\_rdd.map(lambda line: ngram\_fingerprint(str(line),n=2)).distinct().map(lambda x: (x[0],[x[1]])).reduceByKey(lambda x,y: x+y).sortByKey()

t2f\_tmp.filter(lambda x: len(x[1])>=4).collect()

[('111thsstth', ['111TH ST', '11TH ST', '111th St', '11th St']), ('174t74hsstthw1', ['W 174TH ST', 'W .174TH ST', 'W.174 TH ST', 'W 174th St']), ('222ndse2ndst', ['E 222ND ST', 'E 22ND ST', 'E 222nd St', 'E 22nd St']), ('acalcddogalsmaoustug', ['MAC DOUGAL ST', 'MACDOUGAL ST', 'Macdougal St', 'Mac Dougal St']), ('amandamsndstva', ['VANDAM ST', 'VAN DAM ST', 'Van Dam St', 'Vandam St']), ('avcheresherasttctevewe', ['WESTCHESTER AVE', 'WESTCHESTER AVE', 'WEST CHESTER AVE', 'Westchester Ave']), ('blcicoitlvoooppctyvdyb', ['CO-OP CITYBLVD', 'CO-OP CITY BLVD', 'COOP CITY BLVD', 'Co Op City Blvd']), ('bldbelfigfieinldlvngprrispvd', ['SPRINGFIELD BLVD', 'SPRINGFIELD BLVD', ';SPRINGFIELD BLVD', 'Springfield Blvd']), ('dedrederidivrirssive', ['RIVERSIDE DRIVE', 'RIVERSIDE DR', 'RIVERSIDE DR', 'Riverside Dr']), ('hnjonsohplspsttj', ['ST JOHNS PL', 'ST.JOHNS PL', 'ST . JOHNS PL', 'ST .JOHNS PL', 'St Johns Pl'])]

>>> t2f\_tmp.filter(lambda x: len(x[1])>=5).collect()

[('hnjonsohplspsttj', ['ST JOHNS PL', 'ST.JOHNS PL', 'ST . JOHNS PL', 'ST .JOHNS PL', 'St Johns Pl'])]