Lab 3: Record Linking Product Listings

In this lab exercise, you will build on HW 3 to practice more with record linking techniques. **This lab assumes you have attempted HW 3**, if you have not, you may need to review / refer to HW 3 as you work on the lab. You should work in a group of 4-5 students of your choice; you will only need to submit once as a group (see submission instructions below). Remember that labs are graded for sincere effort: if you do not finish the entire lab assignment before the end of class, turn in what you have so far.

Name and NetID

Nathan Kim - njk24 Eugene Kim - yk171

The Data

This lab will use two files: `amazon.csv` and `google.csv` . Both contain product listings including five columns: product id (of course, the same ids are not used in both files), title, description, manufacturer, and price. As in HW 3, your task will be to compute the set of pairs of ids that should be linked. The datasets in this assignment are substantially larger (a few thousand entries each, and some of the titles and descriptions are long strings) and messier (some of price values, for example, have text in the column that should be split off, some values are missing, etc) than in HW 3, so you will need to be careful to (a) spend more time exploring and cleaning the data and (b) being careful about running time (computing all pairwise edit distances will probably take a long time, for example).

In [10]:
```
import pandas as pd

df_a = pd.read_csv('amazon.csv')
df_b = pd.read_csv('google.csv')
df_a.head()
```

Out[10]:

	id	title	description	manufacturer	price
0	0	clickart 950 000 - premier image pack (dvd-rom)	NaN	broderbund	0.00
1	1	ca international - arcserve lap/desktop oem 30pk	oem arcserve backup v11.1 win 30u for laptops ...	computer associates	0.00
2	2	noah's ark activity center (jewel case ages 3-8)	NaN	victory multimedia	0.00
3	3	peachtree by sage premium accounting for nonpr...	peachtree premium accounting for nonprofits 20...	sage software	599.99
4	4	singing coach unlimited	singing coach unlimited - electronic learning ...	carry-a-tune technologies	99.99

As in HW 3, we provide an implementation of edit distance as well as quantifications of error. Make sure that you downloaded `official_matches.csv` into your working directory, and then run the next cell before calling any of these functions. You can refer to HW 3 for example uses, and you are also welcome to use any of your own code from HW 3 to help you in the task for this lab.

In [2]:
```
import lab3_utils
from lab3_utils import edit_dist, precision, recall, f1_score
```

The Problem

You will compute an overall matching of records that should be merged between the two datasets. **You should format your answer as a *set* of pairs of ids**, one from `df_a` and another from `df_g` , such that you predict the corresponding records should be merged. For example, `{(661, 801), (228, 388), (304, 735), (101, 102)}` identifies that the records with ids 661 and 801 should be merged, 228 and 388 should be merged, 304 and 735 should be merged, and 101 and 102 should be merged. The ids are unique between the datasets, so you don't have to worry about a given id appearing in both. You also don't have to worry about the order of the pairs: `(661, 801)` will be treated equivalently as `(801, 661)` for the purpose of quantifying error (and if you include both orders, the duplicate will be removed when quantifying error).

The instructors will maintain a leaderboard of the teams that have achieved the highest F1 scores on the whiteboard. If your team breaks one of the records on the board, raise your hand for an instructor. We will ask for a team name and add your team name and score to the board. The three leading teams by the end of the class will receive extra credit for today's lab (note, as in the homework, you will not receive credit for hard coding any of the official matches).

Your final answer when you turn in this notebook should contain your code below to compute your match, and should then print the F1 score achieved (you do not need to print the entire match itself). After that, you should describe in a few sentences what your record linkage code is doing.

In [20]:
```
def generate_trigrams(s):
    output = set()
    if(len(s)<=3):
        for x in range(len(s)):
            output.add(s[0:x+1])
        for y in range(len(s)):
            output.add(s[y:])
        return output
    else:
        for x in range(3):
            output.add(s[0:x+1])
        for i in range(len(s)-4):
            output.add(s[i:i+3])
        for i in range(len(s)-4):
            output.add(s[i:i+3])
        output.add(s[len(s)-4:len(s)-1])
        output.add(s[len(s)-3:len(s)])
        output.add(s[len(s)-2:])
        output.add(s[len(s)-1])
        return output

print(generate_trigrams("Hello!"))
```

```
{'lo!', 'He', 'ell', 'llo', 'o!', '!', 'H', 'Hel'}
```

In [22]:
```
def jaccsim(a, b):
    intersect = generate_trigrams(a) & generate_trigrams(b)
    union = generate_trigrams(a) | generate_trigrams(b)
    return(len(intersect)/len(union))

print(jaccsim("hello!", "hallo!"))
```

```
0.45454545454545453
```

In [27]:
```
def cleanuppricea(a):
    for x in a['price']:
        strip(x,side)
    return a
def cleapuppriceb(b):
    for x in b['price']:
        strip(x,side)
    return b
def compareprices(a, b):
    listofsims=[]
    cleanuppricea(a)
    cleanuppriceb(b)
    for x in a['price']:
        for y in b['price']:
            if(abs(float(x) - float(y))<3):
                listofsims.append(x)
                listofsims.append(y)
    return listofsims

print(compareprices(df_a, df_b))
```

```
---------------------------------------------------------------------------
----
NameError                                 Traceback (most recent call l
ast)
<ipython-input-27-f7565ad3fa5e> in <module>
     18     return listofsims
     19
---> 20 print(compareprices(df_a, df_b))

<ipython-input-27-f7565ad3fa5e> in compareprices(a, b)
      9 def compareprices(a, b):
     10     listofsims=[]
---> 11     cleanuppricea(a)
     12     cleanuppriceb(b)
     13     for x in a['price']:

<ipython-input-27-f7565ad3fa5e> in cleanuppricea(a)
      1 def cleanuppricea(a):
      2     for x in a['price']:
----> 3         strip(x,side)
      4     return a
      5 def cleapuppriceb(b):

NameError: name 'strip' is not defined
```

In [23]:
```
def jaccsimlist(a, b):
    listofsims=[]
    for x in a['title']:
        for y in b['title']:
            if (0.75 <= jaccsim(x,y) < 1):
                listofsims.append(x)
                listofsims.append(y)
    return listofsims

print(jaccsimlist(df_a, df_b))
```

```
---------------------------------------------------------------------------
----
KeyboardInterrupt                         Traceback (most recent call l
ast)
<ipython-input-23-7a1ea62199a6> in <module>
      8     return listofsims
      9
---> 10 print(jaccsimlist(df_a, df_b))

<ipython-input-23-7a1ea62199a6> in jaccsimlist(a, b)
      3     for x in a['title']:
      4         for y in b['title']:
----> 5             if (0.75 <= jaccsim(x,y) < 1):
      6                 listofsims.append(x)
      7                 listofsims.append(y)

<ipython-input-22-1b041ae79dcc> in jaccsim(a, b)
      1 def jaccsim(a, b):
----> 2     intersect = generate_trigrams(a) & generate_trigrams(b)
      3     union = generate_trigrams(a) | generate_trigrams(b)
      4     return(len(intersect)/len(union))
      5

<ipython-input-20-2f1cfff02ec6> in generate_trigrams(s)
     11             output.add(s[0:x+1])
     12         for i in range(len(s)-4):
---> 13             output.add(s[i:i+3])
     14         for i in range(len(s)-4):
     15             output.add(s[i:i+3])

KeyboardInterrupt:
```

In [ ]:
```
def setofpairs(df_a,df_b):
    listofmatches=jaccsimlist(df_a, df_b)
    setofids=set()
    for x in range(len(listofmatches)):
        if x % 2==0:
            for a in range(len(df_a['title'])):
                if (listofmatches[x]==df_a['title'][a]):
                    for b in range(len(df_b['title'])):
                        if(listofmatches[x+1]==df_b['title'][b]):
                            setofids.add((df_a['id'][a],df_b['id'][b]))
    return(setofids)
print(setofpairs(df_a,df_b))
```

In [ ]:

```
from lab3_utils import edit_dist, precision, recall, f1_score

def jaccsimtitlelist(a, b):
    listofsims=[]
    for x in a['title']:
        for y in b['title']:
            if (0.5 <= jaccsim(x,y) <= 1):
                listofsims.append(x)
                listofsims.append(y)
    return listofsims


def jaccsimdescriptionlist(a, b):
    listofsims=[]
    for x in a['description']:
        for y in b['description']:
            if (0.95<jaccsim(x,y)<= 1):
                listofsims.append(x)
                listofsims.append(y)
    return listofsims

def setoftitlepairs(df_a,df_b):
    listofmatches=jaccsimtitlelist(df_a, df_b)
    setofids=set()
    for x in range(len(listofmatches)):
        if x % 2==0:
            for a in range(len(df_a['title'])):
                if (listofmatches[x]==df_a['title'][a]):
                    for b in range(len(df_b['title'])):
                        if(listofmatches[x+1]==df_b['title'][b]):
                            setofids.add((df_a['id'][a],df_b['id'][b]))
    return(setofids)

def setofdescriptionpairs(df_a,df_b):
    listofmatches=jaccsimdescriptionlist(df_a, df_b)
    setofids=set()
    for x in range(len(listofmatches)):
        if x % 2==0:
            for a in range(len(df_a['description'])):
                if (listofmatches[x]==df_a['description'][a]):
                    for b in range(len(df_b['description'])):
                        if(listofmatches[x+1]==df_b['description'][b]):
                            setofids.add((df_a['id'][a],df_b['id'][b]))
    return(setofids)
our_match = setoftitlepairs(df_a,df_b)|setofdescriptionpairs(df_a,df_b)


print("Precision: ", precision(our_match))
print("Recall: ", recall(our_match))
print("F1 score: ", f1_score(our_match))
```

Write your written response here

Submitting Lab 3

1. Double check that you have written all of your answers along with your supporting work in this notebook. Make sure you save the complete notebook.
2. Double check that your entire notebook runs correctly and generates the expected output. To do so, you can simply select Kernel -> Restart and Run All.
3. You will download two versions of your notebook to submit, a .pdf and a .py. To create a PDF, we reccomend that you select File --> Download as --> HTML (.html). Open the downloaded .html file; it should open in your web broser. Double check that it looks like your notebook, then print a .pdf using your web browser (you should be able to select to print to a pdf on most major web browsers and operating systems). Check your .pdf for readability: If some long cells are being cut off, go back to your notebook and split them into multiple smaller cells. Also, make sure that it is a reasonable length; print statements which are truncated inside of the notebook may come to many pages in the pdf. To get the .py file from your notebook, simply select File -> Download as -> Python (.py).
4. Upload the .pdf to gradescope under lab 3 report and the .py to gradescope under lab 3 code. Only submit once per group, but be sure to add your partner using the group feature on gradescope (https://www.gradescope.com/help#help-center-item-student-group-members).

In [ ]: