create_merged_data

May 9, 2018

In [203]:
```
import pandas as pd
import numpy as np
import csv
```

In [204]:
```
M = pd.read_csv("predictions_all.csv")
left = pd.read_csv("ltable_amazon.csv",encoding = "ISO-8859-1")
right = pd.read_csv("rtable_walmart.csv",encoding = "ISO-8859-1")
```

In [206]:
```
# amazon->wamart matches
# wamart->amazon matches
amazon_to_walmart_mapper = {}
walmart_to_amazon_mapper = {}

for index, row in left.iterrows():
    left_id = row['id']
    #Find list of matches
    match = M.loc[(M['ltable_id'] == left_id)  & (M['predicted_labels'] == 1)]
    if match.empty:
        #No match exists for walmart ids add key-> emptylist
        amazon_to_walmart_mapper[left_id] = []
    else:
        # Match exist
        match_list= match.iloc[:, 2].tolist()
        amazon_to_walmart_mapper[left_id] = match_list

# Do the same thing for walmart dataset
for index, row in right.iterrows():
    right_id = row['id']
    #Find list of matches
    match = M.loc[(M['rtable_id'] == right_id) & (M['predicted_labels'] == 1)]
    if match.empty:
        #No match exists for walmart ids add key-> emptylist
        walmart_to_amazon_mapper[right_id] = []
    else:
        # Match exist
        match_list= match.iloc[:, 1].tolist()
        walmart_to_amazon_mapper[right_id] = match_list
```

In [207]:
```
class Cluster:
    wids = []
    aids = []

    def __init__(self, aids, wids):
        self.aids = aids
        self.wids = wids
```

In [208]:
```
#create list to manage duplicates
wids_touched = []
aids_touched = []
clusters = [] #List of clusters

for key, values in amazon_to_walmart_mapper.items():
    aids = []
    wids = []
    if key not in aids_touched:
        aids.append(key)
        aids_touched.append(key)
        wids.extend(values)
        wids_touched.extend(values)
        for w_id in values:
            if w_id in walmart_to_amazon_mapper:
                a_list = walmart_to_amazon_mapper[w_id]
                for val in a_list:
                    if val not in aids_touched:
                        aids_touched.append(val)
                        aids.append(val)
    clusters.append(Cluster(list(set(aids)),list(set(wids))))
```

In [211]:
```
# Now that we have the custers of matches, let's start merging
# Name - MaxLength
# Price - Exists/Max
# Category - Amazon Category
# Author - MaxLength
# ISBN - Exists/Amazon has preference
# Pages - Max
# Publisher - Exists/ Amazon has preference
# Language - exists/Amazon
# Dimensions - esists/Amazon has preference
# Weight - Amazon
# Rating - MinRating
#M.loc[(M['rtable_id'] == right_id) & (M['predicted_labels'] == 1)]
with open("merged_table.csv", "w", newline='') as csv_file:
    writer = csv.writer(csv_file, delimiter=',')
    writer.writerow(["Name","Sale Price","Category","Author","ISBN10","Pages",
                     "Publisher","Language","Dimensions","Weight","Rating"])
    for cluster in clusters:
```

```
max_name = ""
max_price = 0
category = ""
max_author = ""
isbn = ""
max_pages = 0
publisher = ""
language = ""
dimensions = ""
weight = ""
rating = 100
if cluster.aids and cluster.wids:

    for a_id in cluster.aids:
        if len(left['Name'][a_id]) > len(max_name):
            max_name = left['Name'][a_id]
        if np.isnan(left['Sale Price'][a_id]) == False and
        float(left['Sale Price'][a_id]) > float(max_price):
            max_price = float(left['Sale Price'][a_id])
        if len(left['Category'][a_id]) > len(category):
            category = left['Category'][a_id]
        if len(left['Author'][a_id]) > len(max_author):
            max_author = left['Author'][a_id]
        if len(isbn) == 0:
            isbn = left['ISBN10'][a_id]
        if np.isnan(left['Pages'][a_id]) == False and
        int(left['Pages'][a_id]) > int(max_pages):
            max_pages = int(left['Pages'][a_id])
        if len(publisher) == 0:
            publisher = left['Publisher'][a_id]
        if len(language) == 0:
            language = left['Language'][a_id]
        if len(dimensions) == 0 and left['Dimensions'][a_id] != "nan":
            dimensions = left['Dimensions'][a_id]
            if isinstance(dimensions, str) == False: dimensions = ""
        if len(weight) == 0:
            weight = left['Weight'][a_id]
        if np.isnan(left['Rating'][a_id]) == False and
        int(left['Rating'][a_id]) < int(rating):
            rating = int(left['Rating'][a_id])

    for w_id in cluster.wids:
        if len(right['Name'][w_id]) > len(max_name):
            max_name = right['Name'][w_id]
        if np.isnan(right['Sale Price'][w_id]) == False and
        float(right['Sale Price'][w_id]) > float(max_price):
            max_price = float(right['Sale Price'][w_id])
        if len(right['Author'][w_id]) > len(max_author):
```

```
            max_author = right['Author'][w_id]
        if len(isbn) == 0:
            isbn = right['ISBN10'][w_id]
        if np.isnan(right['Pages'][w_id]) == False and
        int(right['Pages'][w_id]) > int(max_pages):
            max_pages = int(right['Pages'][w_id])
        if len(publisher) == 0:
            publisher = right['Publisher'][w_id]
        if len(language) == 0:
            language = right['Language'][w_id]
        if len(dimensions) == 0 :
            dimensions = right['Dimensions'][w_id]
        if np.isnan(right['Rating'][w_id]) == False and
        int(right['Rating'][w_id]) < int(rating):
            rating = int(right['Rating'][w_id])
    if rating == 100:
        rating = math.nan
    if max_price == 0:
        max_price = math.nan
    if max_pages == 0:
        max_pages = math.nan
    writer.writerow([max_name, max_price, category, max_author, isbn,
                     max_pages, publisher, language, dimensions,
                     weight, rating])
```