Stage 4

Dan Howe and Keith Funkhouser
CS838: Data Science
April 16, 2017

In this project stage, we continued to use the movies/songs/tracks dataset which we used in Stage 3. In that stage, we developed a matcher which matched songs and tracks. In this stage, we merged the two downsampled versions of each of those tables into a single table.

The songs table had the following attributes:

`title, artist_name, year`

The tracks table had the following attributes:

`title, year, episode, song, artists`

“`title`” in the songs table corresponded to "`song`” in the tracks table; “`artist_name`” and “`artists`” corresponded; the two “`year`” attributes corresponded. We therefore needed to develop rules to merge these three attribute pairs. “`episode`” and “`title`” from the tracks table were simply copied over to the new table.

To merge “`artist_name`” and “`artists`”, we simply took “`artists`”. The “`artists`” attribute was effectively a list of artists delimited by a ‘+’ character. In order for a pair to match in the first place, the value of the “`artist_name`” attribute needed to be be similar to one the values in this list, so simply taking the “`artists`” attribute from the tracks table was appropriate.

To merge the two “`year`” attributes, we took the value from the tracks table. The songs table had many missing values (indicated by a year 0), while the tracks table had no missing value. The resulting "`year`" column was densely populated, indicating that this was an effective approach.

To merge the (songs table) “`title`” with “`song`”, we took the shorter of the two strings. If two tuples matched then the names of the songs should have been similar. We reasoned that the longer string probably had additional information appended like “(album version)”, “(radio edit)”, etc.

The schema of the new table, and a few rows to demonstrate the data merging:

	id	title	episode	title_song_merged	artist_name_artists_merged	year_year_merged
0	270943	the surreal life		too little too late	steven page+ed robertson+barenaked ladies	2003
1	575882	phil collins: live at perkins palace		i dont care anymore	phil collins	1983
2	231528	tv land confidential	music (#2.6)	where did our love go	lamont dozier+brian holland+eddie holland+the supremes	2005
3	82676	enie backt	verliebt verlobt verheiratet (#4.6)	l-o-v-e	joss stone	2012

There are 6182 total tuples in this table, 372 of which correspond to matched pairs. The rest are unmatched tuples from the songs and tracks tables.

data_merging.py:

```
import pandas

def merge(fname1, # name of first file
          fname2, # name of second file
          cnames1, # column names in table 1
          cnames2, # column names in table 2
          merging_fn): #functions to merge columns

    f1 = pandas.read_csv(fname1)
    f2 = pandas.read_csv(fname2)

    f1_cols = list(f1.columns.values)
    f2_cols = list(f2.columns.values)

    # check that the input complies with our expectations:
    # 1) the lists of column names must match in length
    assert len(cnames1) == len(cnames2) and len(cnames2) == len(merging_fn), \
        "ERROR: Length of corresponding column lists must match: %d != %d." % \
        (len(cnames1), len(cnames2))
    # 2) the list of merging functions must be the same length
    assert all([col in f1_cols for col in cnames1]), \
        "ERROR: All columns given must be in list of actual column names: %s != %s" % \
        (str(cnames1), str(f1_cols))
    # 3) the input tables must be the same length (1:1 correspondence)
    assert len(f1) == len(f2), \
        "ERROR: Two data frames must have the same number of (corresponding) rows"

    # this is the resulting table that we will populate
    result = pandas.DataFrame()

    # loop over the columns in both tables which do NOT need to be merged, and populate
them
    # in the resulting table
    for col in f1_cols:
        if col not in cnames1:
            result[col] = f1[col]
    for col in f2_cols:
        if col not in cnames2:
            result[col] = f2[col]

    # now, loop over all the columns that must be merged and do so
    for i in range(len(cnames1)):
        col1 = cnames1[i]
        col2 = cnames2[i]
        f = merging_fn[i]

        # construct the tuple pairs to be merged
        e = pandas.concat([f1[col1], f2[col2]], axis=1)

        # perform the merging
```

```
        new_col = "%s_%s_merged" % (col1, col2)

        # insert into the table
        result[new_col] = e.apply(f, axis=1)

    return result

# These are the merging functions:
# 1) merge year with year:
#    simply return the tracks year, as songs has many missing values, while tracks
#    has very few if any
def merge_years(x):
    songs_year = x[0]
    tracks_year = x[1]

    return tracks_year

# 2) merge artist with artists
#    simply return tracks_artist, since, if the pairs are a match, the song artist
#    ought to match one of the artists in the tracks artist, which is actually a
#    list of artists delimited by +
def merge_artists(x):
    songs_artist = x[0]
    tracks_artist = x[1]

    return tracks_artist

# 3) merge title with name
#    return the shorter of the two song names, a longer song name probably has something
#    like a version of that song specified
def merge_names(x):
    songs_name = x[0]
    tracks_name = x[1]

    if len(songs_name) < len(tracks_name):
        return songs_name
    return tracks_name


# Perform the merging
result = merge('datasets/songs_for_merging.csv', 'datasets/tracks_for_merging.csv',
      ['title', 'artist_name', 'year'],
      ['song', 'artists', 'year'],
      [merge_names, merge_artists, merge_years]
)

# Write the file to disk
result.to_csv("E.csv")
```