In [3]:
```
%%javascript
IPython.OutputArea.prototype._should_scroll = function(lines) {
    return false;
}
```

```
<IPython.core.display.Javascript object>
```

Raul G. Martinez

PID: A12461871

UCSD MAS DSE Cohort 6

DSE 203 - Fall 2020

--

Instructions

Assignment 1

Due Oct 19 by 11:59pm

Points 30

Submitting a file upload Available Oct 4 at 12am - Oct 19 at 11:59pm 16 days

Data: Ingredients_beef.csv (in Canvas/Files)

There are 3 columns in the data -- id, recipe_name, ingredients Each row stands for a specific ingredient used in a specific recipe; id is the recipe identifier, recipe_name is the name of the recipe. Since one recipe uses multiple ingredients, there are multiple rows for a single recipe.

Your task is the following.

- (a) pick 20 random elements from the set of ingredients (using SQL). For every ingredient, create a list of recipe ids using different similarity measures and determine which measures are most effective.
- (b) find ingredient names that appear in the recipe titles (approximately).
- (c) Users are looking for recipe titles which have the following ingredients (one query per item below, spelling errors are intentional):
  - (i) spinutch
  - (ii) onions and garlic
  - (iii) hot water
  - (iv) seasoning (Italian)
  - (v) roasted beef

Your submission should include the python scripts (includes SQL scripts) + results

Note:

(1) import the data into database first. I recommend using IDEs (e.g. DataGrip - database management tools;PyCharm -here is a " database" on the sidebar; etc.)

(2) you can select random samples (e.g, N=100) using SQL by the following command:

select * from XXX order by random() limit 100

(3) references for how to connect python to postgresql database

- using Jupyter notebook

https://towardsdatascience.com/jupyter-magics-with-sql-921370099589 (https://towardsdatascience.com/jupyter-magics-with-sql-921370099589) (Links to an external site.)

- using python IDE:

https://pynative.com/python-postgresql-tutorial/ (https://pynative.com/python-postgresql-tutorial/) (Links to an external site.)

Solution

(a) pick 20 random elements from the set of ingredients (using SQL). For every ingredient, create a list of recipe ids using different similarity measures and determine which measures are most effective.

In [13]:
```
#First create table schema and import 'Ingredients_beef.csv' into pgAdmin 4 PostgreSQL

""" SQL Commands:

CREATE TABLE Ingredients (
  id INTEGER,
  recipe_name VARCHAR(150),
  ingredients VARCHAR(150)
);

COPY Ingredients(id, recipe_name, ingredients)
FROM 'C:\Users\rmartinez4\Box\UCSD MAS DSE\2019-rgm001\DSE203\Data for Assignments\Ingredients_beef.csv'
DELIMITER ','
CSV HEADER;

SELECT * FROM Ingredients;

"""
```

In [5]:
```
# install required libraries
# !pip install SQLAlchemy
# !pip install psycopg2
```

In [6]:
```
# import libraries
import pandas as pd
from sqlalchemy.engine import create_engine

# Postgres username, password, and database name
POSTGRES_ADDRESS = 'localhost'
POSTGRES_PORT = '5432'
POSTGRES_USERNAME = 'postgres'
POSTGRES_PASSWORD = 'password'
POSTGRES_DBNAME = 'DSE203_Assignment_1'

# A long string that contains the necessary Postgres login information
postgres_str = ('postgresql://{username}:{password}@{ipaddress}:{port}/{dbname}'
        .format(
            username=POSTGRES_USERNAME,
            password=POSTGRES_PASSWORD,
            ipaddress=POSTGRES_ADDRESS,
            port=POSTGRES_PORT,
            dbname=POSTGRES_DBNAME))

# Create the connection
cnx = create_engine(postgres_str)

# read all data
df = pd.read_sql('SELECT * FROM Ingredients', cnx)
df.shape
```

Out[6]:
```
(189059, 3)
```

In [7]:

```
# read query as dataframe, pick 20 random elements from the set of ingredients
df_Rand20 = pd.read_sql('SELECT * FROM Ingredients ORDER BY random() limit 20', cnx)
df_Rand20
```

Out[7]:

	id	recipe_name	ingredients
0	211063	my favorite chili con carne	'chili powder'
1	22604	spicy tofu casserole with pork	'beef stock'
2	191079	slow cooked spaghetti and meatballs	'green pepper'
3	26338	upside down pizza	'milk'
4	59160	mushroom stuffed meatloaf	'extra lean ground beef'
5	97760	mini meatballs	'dry breadcrumbs'
6	144909	rachael ray s tuscan chicken	'salt'
7	266744	zesty meatballs	'breadcrumbs'
8	315850	maggie s crescent roll casserole	'ground beef'
9	370163	bell peppers stove top beef stuffed red or...	'onion'
10	91717	california burgers	'bacon'
11	274396	golden bee cornish pasties with garlic aioli	'fresh ground black pepper'
12	371423	mom s spinach meatballs	'frozen chopped spinach'
13	330942	flatlander chili	'garlic powder'
14	14234	beef ribs bbq sokalbi koo ee	'short rib of beef'
15	74601	texas crock pot roast	'paprika'
16	10180	mama vaun s meatloaf	'breadcrumbs'
17	387714	ropa vieja shredded beef	'beef bouillon granules'
18	144217	beef and rice skillet	'salt'
19	218911	kittencal s meatball sandwich	'black pepper'

In [8]:
```python
# import libraries
import py_stringmatching as sm
import py_stringsimjoin as ssj
import collections

# create an alphabetical tokenizer, it helps to handle unwanted single quotation characters
alphabet_tok = sm.AlphabeticTokenizer(return_set=True)

# profile attributes dataframe
ssj.profile_table_for_join(df)
```

Out[8]:

Attribute	**Unique values**	**Missing values**	**Comments**
id	17104 (9.05%)	0 (0.0%)	
recipe_name	17022 (9.0%)	0 (0.0%)	
ingredients	5399 (2.86%)	0 (0.0%)	

In [9]:
```python
# define dictionary to store data, format is {threshold: similarity measure: ingredient: list(recipe IDs)}
thres_SimMeas_Ingred_RecIDs_dict = collections.defaultdict()

# compute every threshold, similarity measure, and ingredient
for threshold in [0.2, 0.4, 0.6]:

    # temporal dict to store list of recipe IDs
    SimMeas_Ingred_RecIDs_dict = collections.defaultdict(lambda: collections.defaultdict(list))

    # define token lists, make sure the strings are unique
    x_tokens = df_Rand20['ingredients'].apply(alphabet_tok.tokenize).tolist() # from random limit 20
    y_tokens = df['ingredients'].apply(alphabet_tok.tokenize).tolist() # all data

    # create similarity objects and names, use set-based measures: cosine, Dice, Jaccard, overlap coefficient, Tversky Index
    sim_objs = [sm.Cosine(),sm.Dice(),sm.Jaccard(),sm.OverlapCoefficient(),sm.TverskyIndex()]
    sim_objs_names = ['Cosine','Dice','Jaccard','OverlapCoeff','TverskyIndex']

    # find recipe ID matches for each element x (random 20 ingredient) with all ingredients from database
    for s_name, s in zip(sim_objs_names, sim_objs):

        print('\nProcessing Threshold {} and Similarity Measure \'{}\'...'.format(threshold, s_name))
        count = 0

        for i, x in zip(df_Rand20['ingredients'], x_tokens): # loop over each 20 random ingredient

            i_name = i[1:-1] # get ingredient name

            # compute matches and save lists in dictionary
            df_matches = df[[s.get_sim_score(x,y)>threshold for y in y_tokens]]
            matches_id = list(set(df_matches['id'])) # add as list unique matches
            SimMeas_Ingred_RecIDs_dict[s_name][i_name] += matches_id # substring ingredients to avoid getting single quotes

            try: # show some ingrendient match examples, use try statement in case there are no matches
                if count>2: continue
                print('    Sample matches for {} -> {}'.format(i, list(set(df_matches['ingredients']))[:2]))
                count += 1
            except:
                pass

    # assign to master dictionary with thresholds as keys
    thres_SimMeas_Ingred_RecIDs_dict[threshold] = SimMeas_Ingred_RecIDs_dict
```

```
Processing Threshold 0.2 and Similarity Measure 'Cosine'...
   Sample matches for 'chili powder' -> ["'ginger powder'", "'sesame powder'"]
   Sample matches for 'beef stock' -> ["'reduced-sodium beef broth'", "'beef ribs'"]
   Sample matches for 'green pepper' -> ["'black pepper'", "'dried hot pepper'"]

Processing Threshold 0.2 and Similarity Measure 'Dice'...
   Sample matches for 'chili powder' -> ["'ginger powder'", "'sesame powder'"]
   Sample matches for 'beef stock' -> ["'reduced-sodium beef broth'", "'beef ribs'"]
   Sample matches for 'green pepper' -> ["'black pepper'", "'dried hot pepper'"]

Processing Threshold 0.2 and Similarity Measure 'Jaccard'...
   Sample matches for 'chili powder' -> ["'ginger powder'", "'sesame powder'"]
   Sample matches for 'beef stock' -> ["'beef sirloin steak'", "'beef ribs'"]
   Sample matches for 'green pepper' -> ["'black pepper'", "'dried hot pepper'"]

Processing Threshold 0.2 and Similarity Measure 'OverlapCoeff'...
   Sample matches for 'chili powder' -> ["'ginger powder'", "'sesame powder'"]
   Sample matches for 'beef stock' -> ["'reduced-sodium beef broth'", "'beef ribs'"]
   Sample matches for 'green pepper' -> ["'black pepper'", "'dried hot pepper'"]

Processing Threshold 0.2 and Similarity Measure 'TverskyIndex'...
   Sample matches for 'chili powder' -> ["'ginger powder'", "'sesame powder'"]
   Sample matches for 'beef stock' -> ["'reduced-sodium beef broth'", "'beef ribs'"]
   Sample matches for 'green pepper' -> ["'black pepper'", "'dried hot pepper'"]

Processing Threshold 0.4 and Similarity Measure 'Cosine'...
   Sample matches for 'chili powder' -> ["'ginger powder'", "'sesame powder'"]
   Sample matches for 'beef stock' -> ["'beef sirloin steak'", "'beef ribs'"]
   Sample matches for 'green pepper' -> ["'black pepper'", "'dried hot pepper'"]

Processing Threshold 0.4 and Similarity Measure 'Dice'...
   Sample matches for 'chili powder' -> ["'serrano chili'", "'ginger powder'"]
   Sample matches for 'beef stock' -> ["'beef ribs'", "'soup stock'"]
   Sample matches for 'green pepper' -> ["'black pepper'", "'chili pepper'"]

Processing Threshold 0.4 and Similarity Measure 'Jaccard'...
   Sample matches for 'chili powder' -> ["'hot chili powder'", "'dark chili powder'"]
   Sample matches for 'beef stock' -> ["'beef stock powder'", "'beef stock'"]
   Sample matches for 'green pepper' -> ["'hot green chili pepper'", "'green chili pepper'"]

Processing Threshold 0.4 and Similarity Measure 'OverlapCoeff'...
   Sample matches for 'chili powder' -> ["'ginger powder'", "'sesame powder'"]
   Sample matches for 'beef stock' -> ["'reduced-sodium beef broth'", "'beef ribs'"]
   Sample matches for 'green pepper' -> ["'black pepper'", "'dried hot pepper'"]

Processing Threshold 0.4 and Similarity Measure 'TverskyIndex'...
   Sample matches for 'chili powder' -> ["'serrano chili'", "'ginger powder'"]
   Sample matches for 'beef stock' -> ["'beef ribs'", "'soup stock'"]
   Sample matches for 'green pepper' -> ["'black pepper'", "'chili pepper'"]

Processing Threshold 0.6 and Similarity Measure 'Cosine'...
   Sample matches for 'chili powder' -> ["'hot chili powder'", "'dark chili powder'"]
   Sample matches for 'beef stock' -> ["'beef stock powder'", "'beef stock'"]
   Sample matches for 'green pepper' -> ["'hot green chili pepper'", "'green chili pepper'"]

Processing Threshold 0.6 and Similarity Measure 'Dice'...
   Sample matches for 'chili powder' -> ["'hot chili powder'", "'dark chili powder'"]
   Sample matches for 'beef stock' -> ["'beef stock powder'", "'beef stock'"]
   Sample matches for 'green pepper' -> ["'hot green chili pepper'", "'green chili pepper'"]

Processing Threshold 0.6 and Similarity Measure 'Jaccard'...
   Sample matches for 'chili powder' -> ["'hot chili powder'", "'dark chili powder'"]
   Sample matches for 'beef stock' -> ["'beef stock powder'", "'beef stock'"]
   Sample matches for 'green pepper' -> ["'green chili pepper'", "'green pepper flakes'"]

Processing Threshold 0.6 and Similarity Measure 'OverlapCoeff'...
   Sample matches for 'chili powder' -> ["'hot chili powder'", "'dark chili powder'"]
   Sample matches for 'beef stock' -> ["'beef stock powder'", "'beef stock'"]
   Sample matches for 'green pepper' -> ["'green pepper strips'", "'green hot pepper sauce'"]

Processing Threshold 0.6 and Similarity Measure 'TverskyIndex'...
   Sample matches for 'chili powder' -> ["'hot chili powder'", "'dark chili powder'"]
   Sample matches for 'beef stock' -> ["'beef stock powder'", "'beef stock'"]
   Sample matches for 'green pepper' -> ["'hot green chili pepper'", "'green chili pepper'"]
```

```
In [10]: # report comparison for the number of matches found
         print('Comparing The Number of Recipe ID Matches Found on Each Randomly Selected 20 Ingredients Across Similarity Measures:\n')
         for KEY,VALUE in thres_SimMeas_Ingred_RecIDs_dict.items():

             # find number of recipe ID matches for each ingredient and similarity measure
             meas_ingred_recIDsCount_dict = {k2:{k1:len(v1) for k1,v1 in v2.items()} for k2,v2 in VALUE.items()}

             # report results
             print('********** Threshold = {} **********'.format(KEY))
             print(pd.DataFrame(meas_ingred_recIDsCount_dict))
             print('\nTotal Count in Each Similarity Measure:')
             print(pd.DataFrame(meas_ingred_recIDsCount_dict).sum().sort_values(ascending=False))
             print('\n')
```

```
Comparing The Number of Recipe ID Matches Found on Each Randomly Selected 20 Ingredients Across Similarity Measures:

********** Threshold = 0.2 **********
                           Cosine   Dice  Jaccard  OverlapCoeff  TverskyIndex
chili powder                 4415   4415     4304          4415          4415
beef stock                  17016  17014    16017         17016         17014
green pepper                11926  11925    10945         11926         11925
milk                         1830   1830     1828          1830          1830
extra lean ground beef      17062  17040     9796         17065         17040
dry breadcrumbs              2616   2616     2403          2616          2616
salt                        20432  20432    19934         20432         20432
breadcrumbs                  2496   2496     2496          2496          2496
ground beef                 17052  17051    16487         17052         17051
onion                       10497  10496    10438         10497         10496
bacon                         961    961      955           961           961
fresh ground black pepper   15003  15003     7355         15004         15003
frozen chopped spinach       1445   1442      917          1445          1442
garlic powder                9548   9545     9477          9548          9545
short rib of beef           17018  16989      897         17022         16989
paprika                      1007   1007     1007          1007          1007
beef bouillon granules      17016  17012    12166         17016         17012
black pepper                11327  11327    11118         11327         11327

Total Count in Each Similarity Measure:
OverlapCoeff    178675
Cosine          178667
```

```
TverskyIndex    178601
Dice            178601
Jaccard         138540
dtype: int64


********** Threshold = 0.4 **********
                           Cosine   Dice  Jaccard  OverlapCoeff  TverskyIndex
chili powder                 4304   4006     1899          4415          4006
beef stock                  16017  11549     1621         17016         11549
green pepper                10945   8428     4796         11926          8428
milk                         1830   1812     1721          1830          1812
extra lean ground beef       9796   9202     8973         14816          9202
dry breadcrumbs              2403   1613      783          2616          1613
salt                        20430  19610    16442         20432         19610
breadcrumbs                  2496   2496     2216          2496          2496
ground beef                 16487  14730     9554         17052         14730
onion                       10492  10197     9751         10497         10197
bacon                         961    951      934           961           951
fresh ground black pepper    7355   3869     3582         13073          3869
frozen chopped spinach        917    207      148           917           207
garlic powder                9477   9261     3706          9548          9261
short rib of beef             897    287       75         11754           287
paprika                      1007   1007      984          1007          1007
beef bouillon granules      12166   1766     1148         12166          1766
black pepper                11118   8888     6487         11327          8888

Total Count in Each Similarity Measure:
OverlapCoeff    163849
Cosine          139098
TverskyIndex    109879
Dice            109879
Jaccard          74820
dtype: int64


********** Threshold = 0.6 **********
                           Cosine   Dice  Jaccard  OverlapCoeff  TverskyIndex
chili powder                 1899   1899     1862          1900          1899
beef stock                   1623   1621     1051          1623          1621
green pepper                 4796   4796     1610          4803          4796
milk                         1721   1721     1441          1830          1721
extra lean ground beef       8973   8973     3069          9787          8973
```

```
dry breadcrumbs              783    783     237          783          783
salt                       16442  16442   13332        20432        16442
breadcrumbs                 2216   2216    1092         2496         2216
ground beef                 9556   9554    8607         9556         9554
onion                       9751   9751    7742        10497         9751
bacon                        934    934     762          961          934
fresh ground black pepper   3582   3561    1508         7248         3561
frozen chopped spinach       148    148     133          207          148
garlic powder               3706   3706    1654         3706         3706
short rib of beef             75     75      70          852           75
paprika                      984    984     825         1007          984
beef bouillon granules      1148   1148     417         1766         1148
black pepper                6801   6487    2268         6822         6487

Total Count in Each Similarity Measure:
OverlapCoeff    86276
Cosine          75138
TverskyIndex    74799
Dice            74799
Jaccard         47680
dtype: int64
```

Part (a) Analysis

In the cells above there are two pieces of data displayed, one being an example output for the target ingredients (from the random 20 list) with the matched list of ingredients coming from all the data (only a few are displayed) while the other one is a table of count for Recipe IDs matched, both display data across the different similarity measures tested including Cosine, Dice, Jaccard, Overlap Coefficients, and Tversky Index with threshold values of 0.2, 0.4, and 0.6.

By visual inspection is very clear that the similarity measures are working by comparing the list of ingrents matched with the target ingredient, however, is hard to review and validate manually thousands of string matches. For this reason, I took the approach of basically counting the number of Recipe IDs matched and compare across the threshold parameters and similarity measures. The tables above showed that as we increase the threshold value the number of matches decreases, this is obviously expected, but what is interesting is that some ingredients keep relatively similar number of matches such as "sauerkraut", "onion", and "carrots". On another observation, the order of the number of matches across similarity measures is consistently preserved where Overlap Coefficient is the highest and Jaccard is the lowest, this is true for all threshold values.

Is hard to determine in a detail manner which measure is the most effective without labeling the data, and therefore calculting accuracy with metrics such as Precision and Recall. But for simplicity and ease of analysis we can make two arguments, one being that the similarity measurement with the greatest number of matches is likely performing the best, which in this case is Overlap Coefficient. In fact, all measures have very similar number of matches with the exception of Jaccard; this applies to all threshold values tested. On the other hand, we can argue that Jaccard has better accuracy for true positives than the other measurements and for this reason is expressing less matches.

```
In [ ]:
```

(b) find ingredient names that appear in the recipe titles (approximately).

```
In [11]: threshold = 0.6 # define threshold

         # create similarity objects and names, use set-based measures: cosine, Dice, Jaccard, overlap coefficient, Tversky Index
         sim_objs = [sm.Cosine(),sm.Dice(),sm.Jaccard(),sm.OverlapCoefficient(),sm.TverskyIndex()]
         sim_objs_names = ['Cosine','Dice','Jaccard','OverlapCoeff','TverskyIndex']

         # loop over every similarity measure
         for s_name, s in zip(sim_objs_names, sim_objs):

             x_tokens = df['recipe_name'].apply(alphabet_tok.tokenize).tolist() # recipe name tokens
             y_tokens = df['ingredients'].apply(alphabet_tok.tokenize).tolist() # ingredients tokens

             # find matches filter, similarity measure used is "Overlap Coefficient"
             matches = [s.get_sim_score(x,y)>threshold for x,y in zip(x_tokens, y_tokens)]

             # apply to original dataframe
             df_ingred_recipe_matches = df[matches]

             # report findings
             print('\n\n********** Similarity Measure = \'{}\' **********'.format(s_name))
             print('Number of ingredient matches with recipe title (rows) = {}'.format(df_ingred_recipe_matches.shape[0]))
             print('DataFrame Dimensions: {}, the top 10 are shown below:\n'.format(df_ingred_recipe_matches.shape))
             print(df_ingred_recipe_matches[:10].to_string())
```

```


********** Similarity Measure = 'Cosine' **********
Number of ingredient matches with recipe title (rows) = 1221
DataFrame Dimensions: (1221, 3), the top 10 are shown below:

        id                    recipe_name     ingredients
251   1075                   beef crumble          'beef'
459   2626         beef stuffed acorn squash  'acorn squash'
1059  3980      beef stuffed acorn squash ii  'acorn squash'
1092  4028               apple juice roast   'apple juice'
1155  4191              chipped beef dip ii  'chipped beef'
1323  4692               beef liver creole    'beef liver'
1364  4821              roast beef poor boys    'beef roast'
1594  6451              rotkohl   red cabbage   'red cabbage'
1627  6495   succulent sour cream pot roast    'sour cream'
1836  6817               sour cream burgers    'sour cream'
```

```
********** Similarity Measure = 'Dice' **********
Number of ingredient matches with recipe title (rows) = 848
DataFrame Dimensions: (848, 3), the top 10 are shown below:

        id                recipe_name         ingredients
251   1075               beef crumble              'beef'
459   2626  beef stuffed acorn squash      'acorn squash'
1092  4028          apple juice roast       'apple juice'
1155  4191        chipped beef dip ii      'chipped beef'
1323  4692          beef liver creole        'beef liver'
1364  4821       roast beef poor boys        'beef roast'
1594  6451       rotkohl  red cabbage       'red cabbage'
1836  6817         sour cream burgers        'sour cream'
1937  7057          ranch round steak  'beef round steak'
2045  7232                savory rice              'rice'


********** Similarity Measure = 'Jaccard' **********
Number of ingredient matches with recipe title (rows) = 183
DataFrame Dimensions: (183, 3), the top 10 are shown below:

          id                recipe_name       ingredients
1092    4028          apple juice roast     'apple juice'
1323    4692          beef liver creole      'beef liver'
1594    6451       rotkohl  red cabbage     'red cabbage'
1836    6817         sour cream burgers      'sour cream'
2199    7808         manitoba wild rice       'wild rice'
3802   10623            brisket of beef    'beef brisket'
5544   14137       simmered lamb shanks     'lamb shanks'
5736   14442     ground beef  picadillo     'ground beef'
11275  24429            sour cream soup      'sour cream'
11661  25418  smoked sausage jambalaya  'smoked sausage'


********** Similarity Measure = 'OverlapCoeff' **********
Number of ingredient matches with recipe title (rows) = 5129
DataFrame Dimensions: (5129, 3), the top 10 are shown below:

      id                                 recipe_name            ingredients
23   274                 meatballs in cheese pastry               'cheese'
50   503               curried beef and chicken satay                 'beef'
```

```
51   503              curried beef and chicken satay              'chicken'
67   579             garlic meatballs in lemon sauce               'garlic'
138  628  filet mignon with sweet potato shoestrings  'filet mignon steaks'
156  630                  country style zucchini soup             'zucchini'
178  650                            curried beef loaf                 'beef'
187  749       zucchini lasagna  lasagne    low carb             'zucchini'
213  809         irish beef stew with guinness stout       'beef stew meat'
220  809         irish beef stew with guinness stout       'guinness stout'


********** Similarity Measure = 'TverskyIndex' **********
Number of ingredient matches with recipe title (rows) = 848
DataFrame Dimensions: (848, 3), the top 10 are shown below:

        id                recipe_name         ingredients
251   1075                beef crumble              'beef'
459   2626  beef stuffed acorn squash      'acorn squash'
1092  4028           apple juice roast       'apple juice'
1155  4191          chipped beef dip ii      'chipped beef'
1323  4692           beef liver creole        'beef liver'
1364  4821         roast beef poor boys        'beef roast'
1594  6451         rotkohl  red cabbage       'red cabbage'
1836  6817           sour cream burgers        'sour cream'
1937  7057           ranch round steak  'beef round steak'
2045  7232                 savory rice              'rice'
```

Part (b) Analysis

Analysis for part (b) follows a similar order as compared to part (a), where the masure "Overlap Coefficient" has the highest number of matches with 5129 followed by "Cosine" with 1221, then "Dice" and "TverskyIndex" tied with 848, and lastly "Jaccard" with 183. According to the top 10 matches shown above they all seem to make sense by comparing the strings, however, similar to part (a) is hard to determine which method is performing the best with this dataset and we need to have labeled data to start calculating performance metrics.

```
In [ ]:
```

(c) Users are looking for recipe titles which have the following ingredients (one query per item below, spelling errors are intentional):

- (i) spinutch
- (ii) onions and garlic
- (iii) hot water
- (iv) seasoning (Italian)

- (v) roasted beef

```
In [12]: # dictionary to save data, with format {query: list of recipe titles}
query_recipeList_dict = collections.defaultdict(list)

# define token lists, make sure the strings are unique
user_queries = ['spinutch', 'onions and garlic', 'hot water', 'seasoning (Italian)', 'roasted beef']
y_tokens = df['ingredients'].apply(alphabet_tok.tokenize).tolist() # tokenize ingredients all data

threshold = 0.6 # define threshold

# create similarity objects and names, use set-based measures: cosine, Dice, Jaccard, overlap coefficient, Tversky Index
sim_objs = [sm.Cosine(),sm.Dice(),sm.Jaccard(),sm.OverlapCoefficient(),sm.TverskyIndex()]
sim_objs_names = ['Cosine','Dice','Jaccard','OverlapCoeff','TverskyIndex']

# loop over every similarity measure
for s_name, s in zip(sim_objs_names, sim_objs):
    print('\n\n********** Similarity Measure = \'{}\' **********'.format(s_name))

    for uq in user_queries:

        x =  alphabet_tok.tokenize(uq) # tokenize user queries

        # compute matches and save list of unique recipe titles, similarity measure used is "Overlap Coefficient"
        df_matches = df[[s.get_sim_score(x,y)>threshold for y in y_tokens]]
        matches_recipeName = list(set(df_matches['recipe_name'])) # add as list unique matches
        query_recipeList_dict[uq] = matches_recipeName

        # report a sample of the findings
        print('\tUser query for ingredient = \'{}\''.format(uq))
        print('Total number of unique recipe title matches found = {}'.format(len(matches_recipeName)))
        print('Sample of Recipe Titles Found: {}'.format(matches_recipeName[:5]))
```

```


********** Similarity Measure = 'Cosine' **********
        User query for ingredient = 'spinutch'
Total number of unique recipe title matches found = 0
Sample of Recipe Titles Found: []
        User query for ingredient = 'onions and garlic'
Total number of unique recipe title matches found = 1
Sample of Recipe Titles Found: ['beef and pasta confetti soup']
        User query for ingredient = 'hot water'
```

```
Total number of unique recipe title matches found = 4161
Sample of Recipe Titles Found: ['mock chow mein', 'kibbee nyi  raw kibbee', 'oven bag rump roast', 'johnny m casserole', 'pampered chef taco lasa
gna']
        User query for ingredient = 'seasoning (Italian)'
Total number of unique recipe title matches found = 89
Sample of Recipe Titles Found: ['hamburger helper style beef with noodles', 'bobby s goulash', 'grilled boneless sirloin and vidalia onion skewer
s', 'crock pot beefy tomato stroganoff', 'rooz ma lahem  rice with meat']
        User query for ingredient = 'roasted beef'
Total number of unique recipe title matches found = 604
Sample of Recipe Titles Found: ['authentic 1840 texas chili', 'my mincemeat', 'jean s canned brunswick stew', 'levantine beef with cherries', 'ka
ma meshi  yokokawa style rice casserole']


********** Similarity Measure = 'Dice' **********
        User query for ingredient = 'spinutch'
Total number of unique recipe title matches found = 0
Sample of Recipe Titles Found: []
        User query for ingredient = 'onions and garlic'
Total number of unique recipe title matches found = 0
Sample of Recipe Titles Found: []
        User query for ingredient = 'hot water'
Total number of unique recipe title matches found = 4161
Sample of Recipe Titles Found: ['mock chow mein', 'kibbee nyi  raw kibbee', 'oven bag rump roast', 'johnny m casserole', 'pampered chef taco lasa
gna']
        User query for ingredient = 'seasoning (Italian)'
Total number of unique recipe title matches found = 89
Sample of Recipe Titles Found: ['hamburger helper style beef with noodles', 'bobby s goulash', 'grilled boneless sirloin and vidalia onion skewer
s', 'crock pot beefy tomato stroganoff', 'rooz ma lahem  rice with meat']
        User query for ingredient = 'roasted beef'
Total number of unique recipe title matches found = 604
Sample of Recipe Titles Found: ['authentic 1840 texas chili', 'my mincemeat', 'jean s canned brunswick stew', 'levantine beef with cherries', 'ka
ma meshi  yokokawa style rice casserole']


********** Similarity Measure = 'Jaccard' **********
        User query for ingredient = 'spinutch'
Total number of unique recipe title matches found = 0
Sample of Recipe Titles Found: []
        User query for ingredient = 'onions and garlic'
Total number of unique recipe title matches found = 0
Sample of Recipe Titles Found: []
        User query for ingredient = 'hot water'
Total number of unique recipe title matches found = 199
```

```
Sample of Recipe Titles Found: ['quick italian ground beef dinner', 'mom s ground beef and vegetable soup', 'browning sauce  substitute for kitch
en bouquet  or gravy master', 'hamburger veggie soup', 'melanie s roast']
        User query for ingredient = 'seasoning (Italian)'
Total number of unique recipe title matches found = 0
Sample of Recipe Titles Found: []
        User query for ingredient = 'roasted beef'
Total number of unique recipe title matches found = 0
Sample of Recipe Titles Found: []


********** Similarity Measure = 'OverlapCoeff' **********
        User query for ingredient = 'spinutch'
Total number of unique recipe title matches found = 0
Sample of Recipe Titles Found: []
        User query for ingredient = 'onions and garlic'
Total number of unique recipe title matches found = 3899
Sample of Recipe Titles Found: ['beef and veggies stir fry', 'goulash triestino', 'big juicy stuffed hamburgers', 'japanese gyoza', 'oven bag rum
p roast']
        User query for ingredient = 'hot water'
Total number of unique recipe title matches found = 4161
Sample of Recipe Titles Found: ['mock chow mein', 'kibbee nyi  raw kibbee', 'oven bag rump roast', 'johnny m casserole', 'pampered chef taco lasa
gna']
        User query for ingredient = 'seasoning (Italian)'
Total number of unique recipe title matches found = 89
Sample of Recipe Titles Found: ['hamburger helper style beef with noodles', 'bobby s goulash', 'grilled boneless sirloin and vidalia onion skewer
s', 'crock pot beefy tomato stroganoff', 'rooz ma lahem  rice with meat']
        User query for ingredient = 'roasted beef'
Total number of unique recipe title matches found = 604
Sample of Recipe Titles Found: ['authentic 1840 texas chili', 'my mincemeat', 'jean s canned brunswick stew', 'levantine beef with cherries', 'ka
ma meshi  yokokawa style rice casserole']


********** Similarity Measure = 'TverskyIndex' **********
        User query for ingredient = 'spinutch'
Total number of unique recipe title matches found = 0
Sample of Recipe Titles Found: []
        User query for ingredient = 'onions and garlic'
Total number of unique recipe title matches found = 0
Sample of Recipe Titles Found: []
        User query for ingredient = 'hot water'
Total number of unique recipe title matches found = 4161
Sample of Recipe Titles Found: ['mock chow mein', 'kibbee nyi  raw kibbee', 'oven bag rump roast', 'johnny m casserole', 'pampered chef taco lasa
gna']
```

```
        User query for ingredient = 'seasoning (Italian)'
Total number of unique recipe title matches found = 89
Sample of Recipe Titles Found: ['hamburger helper style beef with noodles', 'bobby s goulash', 'grilled boneless sirloin and vidalia onion skewer
s', 'crock pot beefy tomato stroganoff', 'rooz ma lahem  rice with meat']
        User query for ingredient = 'roasted beef'
Total number of unique recipe title matches found = 604
Sample of Recipe Titles Found: ['authentic 1840 texas chili', 'my mincemeat', 'jean s canned brunswick stew', 'levantine beef with cherries', 'ka
ma meshi  yokokawa style rice casserole']
```

Part (c) Analysis

The results from this part are quite interesting, they follow the similar order of match numbers ("Overlap Coefficient" being the highest while "Jaccard" the lowest) found on previous parts, but now it revealed more information about how each method performs with specific example words. First, the query for "spinutch" was not found anywhere and is likely because all of the methods tested above are set-based and we need something more granular like sequence-based measures to handle typos or any other variations when spelling. Second, the query "onions and garlic" had 3899 matches with the method "Overlap Coefficient" and 1 match "Cosine", and the rest had zero matches. Next, the queries for "hot water", "seasoning (Italian)", and "roasted beef" returned the same number of matches across all methods with 4161, 89, and 604 respectively with the exception of "Jaccard". As observed in parts (a) and (b) above "Jaccard" is the method with the lowest number of matches, and likewise in this part it only found matches for the query "hot water" with 199 while the others had zero matches. As mentioned before, without performance metrics and labels is hard to determine precisely which method is better but to generalize if we had to select a method I would go with "Overlap Coefficient" since it has the highest number of matches, I think is nice to have the extra information is collecting and perhaps filter or clean it with other techniques downstream.

```
In [ ]:
```