utility

July 26, 2022

1 Utility Function Example

1.1 Dependency Package

[ ]:
```
import string
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords

stopwords = stopwords.words('english')
```

1.2 read_file

[ ]:
```
def read_file(path, sep=',', orient=None):

    """
    Read CSV file into DataFrame. Also supports reading JSON file, if orient is␣
↪provided.

    Parameters
    ----------
    path : str, path object, or file-like object
        The file path to access file to read from current directory. The string␣
↪could also be URL.
    sep : str, default ','
        Delimiter to use.
    orient : str , default None
        Indication of expected JSON string format. The set of possible orients␣
↪is:
        'split' : dict like {index -> [index], columns -> [columns], data ->␣
↪[values]}
        'records' : list like [{column -> value}, ... , {column -> value}]
        'index' : dict like {index -> {column -> value}}
        'columns' : dict like {column -> {index -> value}}
        'values' : just the values array
```

```
    Returns
    -------
     : DataFrame or Series
        The DataFrame or Series contains the data in the input file.
    """

    if path[-4:]=='.csv':
        return pd.read_csv(path, sep)
    elif path[-5:]=='.json':
        return pd.read_json(path, orient)
    else:
        raise ValueError("Unrecognized file type")
```

[ ]:
```
df=read_file('../../data/data.csv')
df.head()
```

[ ]:
```
  entity_composite_figi  entity_country entity_exch_code  \
0          BBG000BT1715   United States               UN
1          BBG000BT1715   United States               UN
2          BBG000BT1715   United States               UN
3          BBG000C3J4X4  United Kingdom               LN
4          BBG000C3J4X4  United Kingdom               LN

         entity_exchange   entity_figi     entity_industry  \
0                   NYSE  BBG000BT1984      Packaged Foods
1                   NYSE  BBG000BT1984      Packaged Foods
2                   NYSE  BBG000BT1984      Packaged Foods
3  London Stock Exchange  BBG000C3J543  Travel and Leisure
4  London Stock Exchange  BBG000C3J543  Travel and Leisure

                  entity_name entity_region  entity_relevance  \
0  J.M. Smucker Company (The)      Americas             100.0
1  J.M. Smucker Company (The)      Americas             100.0
2  J.M. Smucker Company (The)      Americas             100.0
3               CARNIVAL PLC        Europe              90.0
4               CARNIVAL PLC        Europe              90.0

           entity_sector  …  new_story_group  \
0  Consumer Non-Durables  …                t
1  Consumer Non-Durables  …                f
2  Consumer Non-Durables  …                f
3               Services  …                t
4               Services  …                t

                             signal_id story_group_count  \
0  ab2020be-ad74-4287-913a-1992b1ccaf11                1
```

```
1  6d76730f-90d1-4a91-bfc6-4d282dd0fa63                 2
2  ae96aa0d-a2d5-4450-a237-e2ebbeb4f375                 3
3  840d8a99-c05a-4678-a581-249596939ef7                 1
4  30e1a37e-c797-46ff-982d-a9f78e08b2a7                 1

                        story_group_id story_group_sentiment_avg  \
0  ee170b18-96cf-4f8a-bcb0-981ba2c3e9a6                     -63.9
1  ee170b18-96cf-4f8a-bcb0-981ba2c3e9a6                     -63.9
2  ee170b18-96cf-4f8a-bcb0-981ba2c3e9a6                     -63.9
3  4c4e5b22-18f6-4d2f-b278-fd79d1d983f7                      91.7
4  0c3ba15c-17eb-48d3-b095-b64d2a76dff0                      91.7

  story_group_sentiment_stdev                 story_id  story_sentiment  \
0                         0.0  5bbfc3e733d3e90001ec8897            -63.9
1                         0.0  5bbfc3ffabb782000109e669            -63.9
2                         0.0  5bbfc40ba656f22bc1efaa05            -63.9
3                         0.0  5bc72782abb78200011a3da8             91.7
4                         0.0  5bc727b3a656f22bc1fec70e             91.7

  story_source story_type
0   agweek.com       news
1   agweek.com       news
2   agweek.com       news
3   agweek.com       news
4   agweek.com       news

[5 rows x 29 columns]
```

[ ]:
```
df=read_file('../../data/partial_data.json', orient='records')
df.head()
```

[ ]:
```
  entity_composite_figi  entity_country entity_exch_code  \
0          BBG000BT1715   United States               UN
1          BBG000BT1715   United States               UN
2          BBG000BT1715   United States               UN
3          BBG000C3J4X4  United Kingdom               LN
4          BBG000C3J4X4  United Kingdom               LN

         entity_exchange   entity_figi     entity_industry  \
0                   NYSE  BBG000BT1984      Packaged Foods
1                   NYSE  BBG000BT1984      Packaged Foods
2                   NYSE  BBG000BT1984      Packaged Foods
3  London Stock Exchange  BBG000C3J543  Travel and Leisure
4  London Stock Exchange  BBG000C3J543  Travel and Leisure

                  entity_name entity_region  entity_relevance  \
0  J.M. Smucker Company (The)      Americas             100.0
```

```
1  J.M. Smucker Company (The)      Americas             100.0
2  J.M. Smucker Company (The)      Americas             100.0
3                CARNIVAL PLC        Europe              90.0
4                CARNIVAL PLC        Europe              90.0

          entity_sector  …  new_story_group  \
0  Consumer Non-Durables  …                t
1  Consumer Non-Durables  …                f
2  Consumer Non-Durables  …                f
3              Services  …                t
4              Services  …                t

                             signal_id story_group_count  \
0  ab2020be-ad74-4287-913a-1992b1ccaf11                1
1  6d76730f-90d1-4a91-bfc6-4d282dd0fa63                2
2  ae96aa0d-a2d5-4450-a237-e2ebbeb4f375                3
3  840d8a99-c05a-4678-a581-249596939ef7                1
4  30e1a37e-c797-46ff-982d-a9f78e08b2a7                1

                        story_group_id story_group_sentiment_avg  \
0  ee170b18-96cf-4f8a-bcb0-981ba2c3e9a6                    -63.9
1  ee170b18-96cf-4f8a-bcb0-981ba2c3e9a6                    -63.9
2  ee170b18-96cf-4f8a-bcb0-981ba2c3e9a6                    -63.9
3  4c4e5b22-18f6-4d2f-b278-fd79d1d983f7                     91.7
4  0c3ba15c-17eb-48d3-b095-b64d2a76dff0                     91.7

  story_group_sentiment_stdev                 story_id  story_sentiment  \
0                         0.0  5bbfc3e733d3e90001ec8897            -63.9
1                         0.0  5bbfc3ffabb782000109e669            -63.9
2                         0.0  5bbfc40ba656f22bc1efaa05            -63.9
3                         0.0  5bc72782abb78200011a3da8             91.7
4                         0.0  5bc727b3a656f22bc1fec70e             91.7

  story_source story_type
0   agweek.com       news
1   agweek.com       news
2   agweek.com       news
3   agweek.com       news
4   agweek.com       news

[5 rows x 29 columns]
```

1.3 com_sim_cat

[ ]:
```
def clean_string(text):

    """
    Clean the punctuation and stopwords in the input string and turn the string␣
 ↪to lower case.

    Parameters
    ----------
    text : str
        The input string.

    Returns
    -------
    text : str
        The cleaned string.

    """

    text = ''.join([char for char in text if char not in string.punctuation])
    text = text.lower()
    text = ' '.join([word for word in text.split() if word not in stopwords])
    return text

def cosine_sim_vectors(vec1, vec2):

    """
    Calculate the cosine similarity between two vectorized strings.
    Two inputs should have the same number of element.

    Parameters
    ----------
    vec1 : array_like
        The first vectorized string.
    vec2 : array_like
        The second vectorized string.

    Returns
    -------
     : float
        The cosine similarity score of the two given vectorized strings.
    """

    vec1 = vec1.reshape(1,-1)
    vec2 = vec2.reshape(1,-1)
    return cosine_similarity(vec1, vec2)[0]
```

```
def com_sim_cat(df, column, sim_threshold=0.4):

    """
    Given a categorical column in a DataFrame, calculate the pairwise cosine␣
↪similarity score
    in the alphabetic-ordered series of categories. If the similarity score is␣
↪higher than the sim_threshold,
    compare the string length of these two category names. The longer␣
↪category-name values in the column
    will be replaced by the similar shorter category-name values.

    Parameters
    ----------
    df : DataFrame
        The DataFrame contains the target categorical column.
    column : str
        The target categorical column name.
    sim_threshold : float, default 0.4
        The threshold of cosines similarity score. If a pair has score higher␣
↪than the threshold, do the replacement.
        The score range is [0, 1].

    Returns
    -------
    : Series
        The target categorical column after combining the similar categories.
    repl_dict : dict
        The dictionary the contains the keys that are replaced by the␣
↪corresponding values.
    """

    # clean the target column
    df[column] = df[column].apply(clean_string)
    # initialize the dict to store the value to replace
    repl_dict = {}
    # get the target column categories as list
    phrases = list(df[column].value_counts().sort_index().index)
    # vectorize the target column categories
    vectorizer = CountVectorizer().fit_transform(phrases)
    vectors = vectorizer.toarray()
    for i in range(0, len(vectors)-1):
        sim = cosine_sim_vectors(vectors[i], vectors[i+1])
        if sim >= sim_threshold:
            if (len(phrases[i])<=len(phrases[i+1])):
                repl_dict[phrases[i+1]] = phrases[i]
```

```
                phrases[i+1] = phrases[i]
            else:
                repl_dict[phrases[i]] = phrases[i+1]
                phrases[i] = phrases[i+1]
        else:
            continue
    return df[column].replace(repl_dict), repl_dict
```

[ ]:
```
df['entity_industry'].value_counts().sort_index()
```

[ ]:
```
Agricultural Chemicals                                  7
Auto Manufacturing                                     14
Auto Parts:O.E.M.                                       2
Beverages                                               3
Beverages (Production/Distribution)                     8
Broadcasting                                            3
Building Materials                                      3
Business Services                                       3
Catalog/Specialty Distribution                          8
Computer Manufacturing                                 11
Computer Software: Prepackaged Software                11
Computer Software: Programming, Data Processing         2
Consumer Electronics/Appliances                         8
Consumer Electronics/Video Chains                       3
Department/Specialty Retail Stores                      1
Departmental Stores                                     2
Electric Utilities: Central                            15
Health Care Equipment and Services                      8
Homebuilding                                            2
Hotels/Resorts                                          2
Industrial Machinery/Components                        12
Integrated oil Companies                                4
Major Pharmaceuticals                                  18
Meat/Poultry/Fish                                       3
Medical Specialities                                    1
Medical/Nursing Services                                4
Military/Government/Technical                           1
Office Equipment/Supplies/Services                      2
Oil Equipment Services and Distribution                 1
Oil and Gas                                             6
Oil and gas                                             1
Oilfield Services/Equipment                             4
Other Consumer Services                                 1
Packaged Foods                                         19
Pharmaceuticals and Biotechnology                       1
Power Generation                                       10
Property-Casualty Insurers                             12
```

```
Radio And Television Broadcasting And Communications Equipment     1
Recreational Products/Toys                                         1
Restaurants                                                        7
Semiconductors                                                     1
Software and Computer Services                                     4
Steel/Iron Ore                                                    10
Television Services                                                6
Travel and Leisure                                                 3
Name: entity_industry, dtype: int64
```

[ ]:
```
test_df, repl = com_sim_cat(df, 'entity_industry')
```

[ ]:
```
test_df.value_counts().sort_index()
```

[ ]:
```
agricultural chemicals                                  7
auto partsoem                                          16
beverages                                              11
broadcasting                                            3
building materials                                      3
business services                                       3
catalogspecialty distribution                           8
computer manufacturing                                 11
computer software prepackaged software                 13
consumer electronicsappliances                          3
consumer services                                       9
departmental stores                                     3
electric utilities central                             15
health care equipment services                          8
homebuilding                                            2
hotelsresorts                                           2
industrial machinerycomponents                         12
integrated oil companies                                4
major pharmaceuticals                                  18
meatpoultryfish                                         3
medical specialities                                    1
medicalnursing services                                 4
militarygovernmenttechnical                             1
office equipmentsuppliesservices                        2
oil equipment services distribution                     1
oil gas                                                 7
oilfield servicesequipment                              4
packaged foods                                         19
pharmaceuticals biotechnology                           1
power generation                                       10
propertycasualty insurers                              12
radio television broadcasting communications equipment  1
recreational productstoys                               1
```

```
restaurants                                    7
semiconductors                                 1
software computer services                     4
steeliron ore                                 10
television services                            6
travel leisure                                 3
Name: entity_industry, dtype: int64
```

[ ]:
```
repl
```

[ ]:
```
{'auto manufacturing': 'auto partsoem',
 'beverages productiondistribution': 'beverages',
 'computer software programming data processing': 'computer software prepackaged
software',
 'consumer electronicsvideo chains': 'consumer electronicsappliances',
 'consumer electronicsappliances': 'consumer services',
 'departmentspecialty retail stores': 'departmental stores'}
```

1.4 comp_key

[ ]:
```
def comp_key(df, column1, column2, key_name, concat_sign=':'):

    """
    Create a new column in the given DataFrame that contains concatenation of␣
↪two given columns as composite key.

    Parameters
    ----------
    df: DataFrame
        The DataFrame contains column1 and column2 and in which a new column of␣
↪composite keys would be created.
    column1: str
        The column name of the first column to be concatenated.
    column2: str
        The column name of the second column to be concatenated.
    key_name: str
        The name of the new composite key column.
    concat_sign: str, default ':
        The sign to concat the filed of the first column and the field of the␣
↪second column.

    Returns
    -------
     : Series
        The column of the composite keys.
    """
```

```
    df[key_name] = df[column1] + concat_sign + df[column2]
    return df[key_name]
```

[ ]:
```
comp_key(df, "entity_exch_code", "entity_ticker", "composite_ticker")
```

[ ]:
```
0      UN:SJM
1      UN:SJM
2      UN:SJM
3      LN:CCL
4      LN:CCL
        …
244    UN:NUE
245    UN:NUE
246    UN:NUE
247    UN:NUE
248    UN:NUE
Name: composite_ticker, Length: 249, dtype: object
```

[ ]:
```

```