SSIE 500: Homework 2 Working Code

September 11, 2020

[ ]:
```
import pandas as pd
import string
import numpy as np
from collections import Counter, OrderedDict
import matplotlib.pyplot as plt
import copy
import requests as r
import seaborn as sns
%matplotlib inline

url = 'https://raw.githubusercontent.com/grantaguinaldo/mcmc/master/
 ↪pride_prejudice.txt'
```

[ ]:
```
def load_data(url):
    string_punctuation = '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~‘êàé-’”“'
    url = url
    data = r.get(url)
    f = data.text
    print('Text File Has Been Downloaded')
    remove_bom = f.replace('\ufeff', '###')
    comma_delimit = remove_bom.replace('\n', ',').strip().lower().replace('\r',␣
 ↪'').split(',')
    clean_text = [each for each in comma_delimit if (str.rstrip(each) != '') or\
                  (str.rstrip(each) not in string_punctuation)]
    return pd.DataFrame({'text': clean_text})

def clean(s):
    '''
    Remove punctuation, numeric values and all extra spaces from string.
    '''
    string_punctuation = '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~‘êàé-’”“'
    # remove punctuation
    no_punc = s.translate(str.maketrans('', '', string_punctuation))
    # remove num
    no_num = ''.join([each for each in no_punc if not each.isdigit()])
    # remove extra spaces
    return ' '.join(no_num.split())
```

```
def count_alpha(x):
    return Counter(x)

def count(s):
    count_a = s.count('a')
    count_b = s.count('b')
    count_c = s.count('c')
    count_d = s.count('d')
    count_e = s.count('e')
    count_f = s.count('f')
    count_g = s.count('g')
    count_h = s.count('h')
    count_i = s.count('i')
    count_j = s.count('j')
    count_k = s.count('k')
    count_l = s.count('l')
    count_m = s.count('m')
    count_n = s.count('n')
    count_o = s.count('o')
    count_p = s.count('p')
    count_q = s.count('q')
    count_r = s.count('r')
    count_s = s.count('s')
    count_t = s.count('t')
    count_u = s.count('u')
    count_v = s.count('v')
    count_w = s.count('w')
    count_x = s.count('x')
    count_y = s.count('y')
    count_z = s.count('z')
    count_space = s.count(' ')

    return {'a': count_a, 'b': count_b, 'c': count_c, 'd': count_d, 'e': count_e,
            'f': count_f, 'g': count_g, 'h': count_h, 'i': count_i, 'j': count_j,
            'k': count_k, 'l': count_l, 'm': count_m, 'n': count_n, 'o': count_o,
            'p': count_p, 'q': count_q, 'r': count_r, 's': count_s, 't': count_t,
            'u': count_u, 'v': count_v, 'w': count_w, 'x': count_x, 'y': count_y,
            'z': count_z, 'space': count_space}

def markov(s):
    return markov_pred_dict[s]

def markov_sampler(char_init, n_iter, markov_dict):
    char_now = char_init
    markov_str = []
```

```
    n_iter = n_iter
    for i in range(n_iter):
        char_now = markov(char_now)
        markov_str.append(char_now)
    return ''.join(markov_str)

def generate_kgram(s, n):
    return Counter([s[i:i+n] for i in range(0, len(s), 1)])

def graph(x, y, data, ylabel, xlabel, title):
    sns.set(rc={'figure.figsize':(15,5)})
    sns.barplot(x=x, y=y, data=data)
    plt.ylabel(ylabel, fontsize=16)
    plt.xlabel(xlabel, fontsize=16)
    plt.ylim(0, 0.180, 0.025)
    plt.xticks(fontsize=15)
    plt.yticks(fontsize=15)
    plt.title(title, fontsize=17)
    return plt.show()

idx_list = ['space', 'a', 'b', 'c',
            'd', 'e', 'f', 'g', 'h',
            'i', 'j', 'k', 'l', 'm',
            'n', 'o', 'p', 'q', 'r',
            's', 't', 'u', 'v', 'w',
            'x', 'y', 'z']

new_col_list = ['first_pos', ' ', 'a', 'b', 'c',
                'd', 'e', 'f', 'g', 'h', 'i',
                'j', 'k', 'l', 'm', 'n', 'o',
                'p', 'q', 'r', 's', 't', 'u',
                'v', 'w', 'x', 'y', 'z']

idx_list_2 = ['first_pos', 'a', 'b', 'c',
            'd', 'e', 'f', 'g', 'h',
            'i', 'j', 'k', 'l', 'm',
            'n', 'o', 'p', 'q', 'r',
            's', 't', 'u', 'v', 'w',
            'x', 'y', 'z']

df = load_data(url)
df.shape
```

[ ]:
```
df_clean = copy.deepcopy(df)
df_clean.loc[:, 'clean_string'] = df_clean['text'].apply(clean)
df_clean.replace('', np.nan, inplace=True)
df_clean.describe()
```

[ ]:
```
df_clean = df_clean[~df_clean['clean_string'].isna()]
df_clean.describe()
```

[ ]:
```
df_clean.loc[:, 'clean_string_count'] = df_clean['clean_string'].
 ↪apply(count_alpha)
df_clean.loc[:, 'clean_string_count_py'] = df_clean['clean_string'].apply(count)

list_dict = [dict(each) for each in df_clean.clean_string_count.tolist()]
final_dist = {}
for d in list_dict:
    for k in d.keys():
        final_dist[k] = final_dist.get(k, 0) + d[k]

list_dict_py = [dict(each) for each in df_clean.clean_string_count_py.tolist()]
final_dist_py = {}
for d in list_dict_py:
    for k in d.keys():
        final_dist_py[k] = final_dist_py.get(k, 0) + d[k]
```

[ ]:
```
# Create df_freq

df_freq = pd.DataFrame(final_dist.items(), columns=['letter', 'count'])
df_freq['freq'] = df_freq['count'] / df_freq['count'].sum()
df_freq.at[3, 'letter'] = 'space'
df_freq.sort_values(by='count', ascending=False, inplace=True)
df_freq.reset_index(drop=True, inplace=True)
df_freq['rank'] = df_freq.index + 1

# Create df_freq_py

df_freq_py = pd.DataFrame(final_dist_py.items(), columns=['letter', 'count'])
df_freq_py['freq'] = df_freq_py['count'] / df_freq_py['count'].sum()
#df_freq_py.at[3, 'letter'] = 'space'
df_freq_py.sort_values(by='count', ascending=False, inplace=True)
df_freq_py.reset_index(drop=True, inplace=True)
df_freq_py['rank'] = df_freq_py.index + 1

# Check if both df are the same
df_freq_py.equals(df_freq)
```

[ ]:
```
graph(x='letter',
      y='freq',
      data=df_freq,
      ylabel='Count \n Frequency',
      xlabel='Letter',
      title='Distrubution of Letters in Pride and Prejudice by \n Jane Austen␣
 ↪(Using Counter)')
```

[ ]:
```
df_clean.loc[:, 'kgrams'] = df_clean['clean_string'].apply(generate_kgram,␣
 ↪args=[2])
df_clean.head()
```

[ ]:
```
kgram_list_dict_py = [dict(each) for each in df_clean.kgrams.tolist()]
kgram_dist= {}
for d in kgram_list_dict_py:
    for k in d.keys():
        kgram_dist[k] = kgram_dist.get(k, 0) + d[k]

df_kgram = pd.DataFrame(kgram_dist.items(), columns=['kgram', 'count'])
df_kgram['kgram_len'] = df_kgram['kgram'].str.split(' ').str.len()
two_grams = df_kgram[df_kgram.kgram_len == 1]

two_grams = copy.deepcopy(two_grams)

two_grams.loc[:, 'first_pos'] = two_grams['kgram'].str[0]
two_grams.loc[:, 'second_pos'] = two_grams['kgram'].str[1]

df = two_grams[['kgram', 'count', 'first_pos', 'second_pos']]
df.head()
```

[ ]:
```
df_trans = df.pivot_table(index=['first_pos'], columns='second_pos',␣
 ↪values='count')
df_trans['first_pos'] = df_trans.index
df_trans.reset_index(drop=True, inplace=True)
df_reorder = df_trans.reindex(columns=idx_list_2)
df_reorder.head()
```

[ ]:
```
df_reorder['idxmax'] = df_reorder.iloc[:, 1:-1].idxmax(axis=1)
markov_pred_dict = dict(zip(df_reorder['first_pos'].tolist(),␣
 ↪df_reorder['idxmax'].tolist()))
markov_pred_dict
```

[ ]:
```
markov_string = markov_sampler(char_init='t', n_iter=3000,␣
 ↪markov_dict=markov_pred_dict)
markov_string
```

[ ]:
```
df_kgram = copy.deepcopy(df_kgram)
df_kgram.loc[:, 'first_pos'] = df_kgram['kgram'].str[0]
df_kgram.loc[:, 'second_pos'] = df_kgram['kgram'].str[1]
df_k = df_kgram[['kgram', 'count', 'first_pos', 'second_pos']]
df_ktrans = df_k.pivot_table(index=['first_pos'], columns='second_pos',␣
 ↪values='count')
df_ktrans['first_pos'] = df_ktrans.index
df_ktrans.reset_index(drop=True, inplace=True)
```

```
df_ktrans.fillna(0, inplace=True)

df_ktrans = df_ktrans.reindex(columns=new_col_list).fillna(0)
df_ktrans.rename(columns={' ': 'space'}, inplace=True)
df_ktrans.loc[0, 'first_pos'] = 'space'
df_ktrans['total'] = df_ktrans.iloc[:, 0:].sum(axis=1)
df_ktrans.head()
```

[ ]:
```
df_ktrans_freq = df_ktrans.iloc[:, 1:].div(df_ktrans['total'] , axis=0)
df_ktrans_freq_2 = df_ktrans_freq.iloc[:, 0:27]
df_ktrans_freq_2['idx'] = idx_list
df_ktrans_freq_2.set_index('idx', inplace=True)
```

[ ]:
```
sns.set(rc={'figure.figsize':(17,8)})
sns.heatmap(df_ktrans_freq_2.iloc[:, 0:27], linewidths=2, yticklabels=1,␣
 ↪cmap='Blues')
plt.ylabel('First Position', fontsize=14)
plt.xlabel('Second Position', fontsize=14)
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)
plt.title('Markov Transistion Frequency Between Two Letters \n', fontsize=14)
plt.show()
```