ReligiousText_DivinePatterns_code_part1

April 11, 2021

[1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patheffects as PathEffects
from gensim.models.fasttext import FastText
from gensim.models.fasttext import load_facebook_model
import nltk
from nltk.cluster import KMeansClusterer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
%matplotlib inline
```

[2]:

```
import seaborn as sns
sns.set_style('darkgrid')
sns.set_palette('muted')
sns.set_context("notebook", font_scale=1.5,
                rc={"lines.linewidth": 2.5})
RS = 123
```

[3]:

```
df = pd.read_csv('AllBooks_baseline_DTM_Labelled.csv')
df.rename(columns={'Unnamed: 0': 'Books'}, inplace=True)
df.head()
```

[3]:

```
          Books  foolishness  hath  wholesome  takest  feelings  anger  \
0  Buddhism_Ch1            0     0          0       0         0      0
1  Buddhism_Ch2            0     0          0       0         0      0
2  Buddhism_Ch3            0     0          0       0         0      0
3  Buddhism_Ch4            0     0          0       0         0      0
4  Buddhism_Ch5            0     0          0       0         0      0

   vaivaswata  matrix  kindled  …  erred  thinkest  modern  reigned  \
0           0       0        0  …      0         0       0        0
1           0       0        0  …      0         0       0        0
2           0       0        0  …      0         0       0        0
3           0       0        0  …      0         0       0        0
4           0       0        0  …      0         0       0        0
```

```
   sparingly  visual  thoughts  illumines  attire  explains
0          0       0         0          0       0         0
1          0       0         0          0       0         0
2          0       0         0          0       0         0
3          0       0         0          0       0         0
4          0       0         0          0       0         0

[5 rows x 8267 columns]
```

[4]:

```python
words = np.array(df.columns[1:])
word_count = pd.DataFrame(df.sum()[1:], columns=['Count'])
```

[5]:

```python
words_clean = []
for w in words:
    if len(w) > 2 and len(w) < 15:
        words_clean.append(w)
words_clean = np.array(words_clean)
df_clean = df[words_clean]

df_clean.drop(columns=words_clean[np.where(df_clean.sum().values < 5)],␣
 ↪inplace=True)
df_clean['Books'] = df['Books'].values
df = df_clean
cols = df.columns.tolist()
cols = cols[-1:] + cols[:-1]
df = df[cols]

words = np.array(df.columns[1:])
```

```
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-
packages/pandas/core/frame.py:4308: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
<ipython-input-5-816d8997bfff>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean['Books'] = df['Books'].values
```

[6]:

```python
df.head()
```

[6]:

```
          Books  hath  feelings  anger  open  rage  looketh  illumination  \
0  Buddhism_Ch1     0         0      0     1     0        0             0
1  Buddhism_Ch2     0         0      0     0     0        0             0
2  Buddhism_Ch3     0         0      0     0     0        0             0
3  Buddhism_Ch4     0         0      0     0     0        0             1
4  Buddhism_Ch5     0         0      0     0     0        0             0

   tell  build  …  glad  needs  well  state  production  developed  \
0     0      0  …     0      0     0      0           0          0
1     0      0  …     0      0     0      0           0          0
2     1      0  …     0      0     0      0           0          0
3     0      0  …     0      0     0      0           0          2
4     0      0  …     0      0     0      0           0          0

   regarded  taketh  thoughts  illumines
0         0       0         0          0
1         0       0         0          0
2         0       0         0          0
3         0       0         0          0
4         0       0         0          0

[5 rows x 2190 columns]
```

[7]:

```python
buddhism = df[df['Books'].str.contains('Buddhism')]
buddhism = pd.DataFrame(buddhism.sum()[1:], columns=['Count'])

taoteching = df[df['Books'].str.contains('TaoTeChing')]
taoteching = pd.DataFrame(taoteching.sum()[1:], columns=['Count'])

upanishad = df[df['Books'].str.contains('Upanishad')]
upanishad = pd.DataFrame(upanishad.sum()[1:], columns=['Count'])

yogasutra = df[df['Books'].str.contains('YogaSutra')]
yogasutra = pd.DataFrame(yogasutra.sum()[1:], columns=['Count'])

proverb = df[df['Books'].str.contains('Proverb')]
proverb = pd.DataFrame(proverb.sum()[1:], columns=['Count'])

eccleasiasticus = df[df['Books'].str.contains('Eccleasiasticus')]
eccleasiasticus = pd.DataFrame(eccleasiasticus.sum()[1:], columns=['Count'])

wisdom = df[df['Books'].str.contains('Wisdom')]
wisdom = pd.DataFrame(wisdom.sum()[1:], columns=['Count'])

words = np.array(df.columns[1:])
```

[8]:
```
eastern = pd.concat([buddhism.T, taoteching.T])
eastern.index = ['Buddhism', 'Tao']
eastern = eastern.drop(columns=words[np.where(eastern.sum() == 0)])
eastern
```

[8]:
```
          feelings  anger  open  rage  illumination  tell  neither  soft  mentally  \
Buddhism        19      0     2     0             1    14       15     0         5
Tao              0      1     2     1             0     1        1     7         0

          land  …  business  red  grows  needs  well  state  production  developed  \
Buddhism     4  …         0    3      3      0     5      0           4         10
Tao          2  …         1    0      0      2     3     28           1          0

          regarded  thoughts
Buddhism         0         9
Tao              3         0

[2 rows x 1279 columns]
```

[9]:
```
eastern_words = np.array(eastern.columns)
eastern_words
```

[9]:
```
array(['feelings', 'anger', 'open', …, 'developed', 'regarded',
       'thoughts'], dtype=object)
```

[10]:
```
# Creating pretrained fasttext model from Crawl EN
fb_model = load_facebook_model('crawl-300d-2M-subword.bin')
```

[11]:
```
fb_model.build_vocab(eastern_words, update = True)
fb_model.train(eastern_words, total_examples = fb_model.corpus_count, epochs =␣
 ↪100)
```

[11]:
```
(22232, 812400)
```

[12]:
```
print(fb_model.wv.similarity('soul', 'mind'))
```

```
0.5782816
```

[13]:
```
X = fb_model.wv[eastern_words]
```

[14]:
```
X.shape
```

[14]:
```
(1279, 300)
```

[15]:
```
# Utility function to visualize the outputs of PCA and t-SNE

def fashion_scatter(x, colors, data):
    # choose a color palette with seaborn.
    num_classes = len(np.unique(colors))
    palette = np.array(sns.color_palette("hls", num_classes))

    # create a scatter plot.
    f = plt.figure(figsize=(16, 16))
    ax = plt.subplot(aspect='equal')
    sc = ax.scatter(x[:,0], x[:,1], lw=0, s=40, c=palette[np.array(colors).
 ↪astype(int)])
    plt.xlim(-25, 25)
    plt.ylim(-25, 25)
    ax.axis('off')
    ax.axis('tight')

    # add the labels for each digit corresponding to the label
    txts = []

    for i in range(num_classes):

        # Position of each label at median of data points.
        x_temp = data[data['Y'] == i]
        xtext = np.median(x_temp['pca1'])
        ytext = np.median(x_temp['pca2'])
        txt = ax.text(xtext, ytext, str(i), fontsize=24)
        txt.set_path_effects([
            PathEffects.Stroke(linewidth=5, foreground="w"),
            PathEffects.Normal()])
        txts.append(txt)

    return f, ax, sc, txts
```

[300]:

```
NUM_CLUSTERS = 9
kclusterer = KMeansClusterer(NUM_CLUSTERS, distance = nltk.cluster.util.
 ↪cosine_distance, repeats = 25)
Y = kclusterer.cluster(X, assign_clusters = True)
```

[301]:

```
pca_50 = PCA(n_components=50)
pca_result_50 = pca_50.fit_transform(X)
```

[302]:

```
pca_tsne = TSNE(random_state = RS).fit_transform(pca_result_50)
```

[303]:

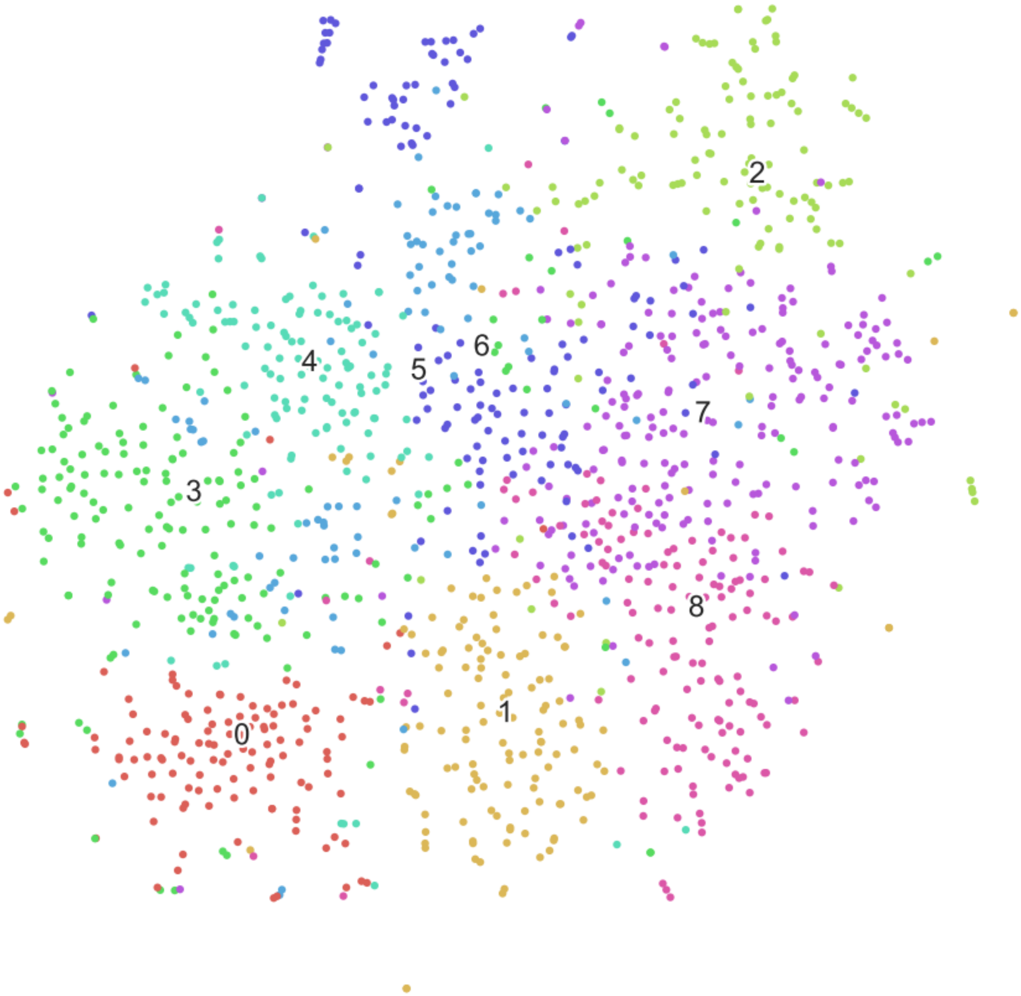
```
tsne_df = pd.DataFrame(columns = ['pca1','pca2'])
tsne_df['pca1'] = pca_tsne[:,0]
tsne_df['pca2'] = pca_tsne[:,1]
```

[304]:

```
tsne_labelled = tsne_df.copy()
tsne_labelled['Y'] = Y
```

```
[305]: fashion_scatter(tsne_df.values, Y, tsne_labelled)
```

```
[305]: (<Figure size 1152x1152 with 1 Axes>,
        <AxesSubplot:>,
        <matplotlib.collections.PathCollection at 0x7fe55016acd0>,
        [Text(-23.538057, -22.648022, '0'),
         Text(0.5663059, -20.755497, '1'),
         Text(23.645163, 24.58831, '2'),
         Text(-27.93565, -2.1824245, '3'),
         Text(-17.315462, 8.744926, '4'),
         Text(-7.355224, 8.059377, '5'),
         Text(-1.5707259, 10.055416, '6'),
         Text(18.67114, 4.4432774, '7'),
         Text(18.08054, -11.8905115, '8')])
```



[137]:
```
tsne_final = tsne_labelled.copy()
tsne_final['Word'] = eastern_words
tsne_final.rename(columns={'pca1': 'x', 'pca2': 'y', 'Y': 'Category'},␣
 ↪inplace=True)

bud_t = buddhism.T
buddhist_words = np.array(bud_t.columns)
buddhist_words = np.array(bud_t.drop(columns=buddhist_words[np.where(bud_t.
 ↪sum()==0)]).columns)
tao_t = taoteching.T
tao_words = np.array(tao_t.columns)
tao_words = np.array(tao_t.drop(columns=tao_words[np.where(tao_t.sum()==0)]).
 ↪columns)

books = []
for i in range(len(eastern_words)):
    if eastern_words[i] in buddhist_words and eastern_words[i] in tao_words:
        books.append('Both')
    elif eastern_words[i] in buddhist_words:
        books.append('Buddhist')
    elif eastern_words[i] in tao_words:
        books.append('Tao')
tsne_final['Book'] = books
```

[311]:
```
concept = pd.read_csv('chai.csv')
```

[312]:
```
concept
```

[312]:
```
      Unnamed: 0          x          y  Category         Word      Book
0              0 -18.182987 -21.067402         7     feelings  Buddhist
1              1  23.062016 -26.539265         7        anger       Tao
2              2  -6.889199   2.987535         4         open      Both
3              3  23.199852 -26.584093         7         rage       Tao
4              4   1.920885 -13.001882         9  illumination  Buddhist
…            …          …          …         …           …         …
1274        1274  -8.441429 -11.421192         8        state       Tao
1275        1275  -6.095231  -9.274641         8   production      Both
1276        1276  -2.047643  30.918629         0    developed  Buddhist
1277        1277  -0.478214  29.996874         0     regarded       Tao
1278        1278 -18.891659 -20.615662         8     thoughts  Buddhist

[1279 rows x 6 columns]
```

[ ]:
```

```

ReligiousText_DivinePatterns_code_part2

April 11, 2021

[290]:
```
from ibm_watson import ToneAnalyzerV3
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
import pandas as pd
import numpy as np
import json
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

[409]:
```
#load in dataset
df = pd.read_csv('AllBooks_baseline_DTM_Labelled.csv')
df.rename(columns={'Unnamed: 0': 'Books'}, inplace=True)
words = np.array(df.columns[1:])
```

[410]:
```
df.drop(columns=['s'], inplace=True)
```

[294]:
```
df.head()
```

[294]:
```
          Books  foolishness  hath  wholesome  takest  feelings  anger  \
0  Buddhism_Ch1            0     0          0       0         0      0
1  Buddhism_Ch2            0     0          0       0         0      0
2  Buddhism_Ch3            0     0          0       0         0      0
3  Buddhism_Ch4            0     0          0       0         0      0
4  Buddhism_Ch5            0     0          0       0         0      0

   vaivaswata  matrix  kindled  …  erred  thinkest  modern  reigned  \
0           0       0        0  …      0         0       0        0
1           0       0        0  …      0         0       0        0
2           0       0        0  …      0         0       0        0
3           0       0        0  …      0         0       0        0
4           0       0        0  …      0         0       0        0

   sparingly  visual  thoughts  illumines  attire  explains
0          0       0         0          0       0         0
1          0       0         0          0       0         0
2          0       0         0          0       0         0
```

```
3          0       0         0          0       0         0
4          0       0         0          0       0         0

[5 rows x 8267 columns]
```

[411]:

```
#split data by book
buddhism = df[df['Books'].str.contains('Buddhism')]
taoteching = df[df['Books'].str.contains('TaoTeChing')]
upanishad = df[df['Books'].str.contains('Upanishad')]
yogasutra = df[df['Books'].str.contains('YogaSutra')]
proverb = df[df['Books'].str.contains('Proverb')]
eccleasiasticus = df[df['Books'].str.contains('Eccleasiasticus')]
wisdom = df[df['Books'].str.contains('Wisdom')]
```

[296]:

```
buddhism.head()
```

[296]:

```
          Books  foolishness  hath  wholesome  takest  feelings  anger  \
0  Buddhism_Ch1            0     0          0       0         0      0
1  Buddhism_Ch2            0     0          0       0         0      0
2  Buddhism_Ch3            0     0          0       0         0      0
3  Buddhism_Ch4            0     0          0       0         0      0
4  Buddhism_Ch5            0     0          0       0         0      0

   vaivaswata  matrix  kindled  …  erred  thinkest  modern  reigned  \
0           0       0        0  …      0         0       0        0
1           0       0        0  …      0         0       0        0
2           0       0        0  …      0         0       0        0
3           0       0        0  …      0         0       0        0
4           0       0        0  …      0         0       0        0

   sparingly  visual  thoughts  illumines  attire  explains
0          0       0         0          0       0         0
1          0       0         0          0       0         0
2          0       0         0          0       0         0
3          0       0         0          0       0         0
4          0       0         0          0       0         0

[5 rows x 8267 columns]
```

[412]:

```
#get the sum of all word counts for each book
buddhism_combined = pd.DataFrame(buddhism.sum()[1:], columns=['Count'])
taoteching_combined = pd.DataFrame(taoteching.sum()[1:], columns=['Count'])
upanishad_combined = pd.DataFrame(upanishad.sum()[1:], columns=['Count'])
yogasutra_combined = pd.DataFrame(yogasutra.sum()[1:], columns=['Count'])
proverb_combined = pd.DataFrame(proverb.sum()[1:], columns=['Count'])
eccleasiasticus_combined = pd.DataFrame(eccleasiasticus.sum()[1:],␣
 ↪columns=['Count'])
```

```
wisdom_combined = pd.DataFrame(wisdom.sum()[1:], columns=['Count'])
words_combined = pd.DataFrame(df.sum()[1:], columns=['Count'])
```

```
[297]: buddhism_combined.head()
```

```
[297]:              Count
       foolishness     0
       hath            0
       wholesome       0
       takest          0
       feelings       19
```

```
[413]: #flatten the dataframes in order to combine them
       bud_flat = buddhism_combined.T.rename(index={'Count': 'Buddhism'})
       tao_flat = taoteching_combined.T.rename(index={'Count': 'TaoTeChing'})
       up_flat = upanishad_combined.T.rename(index={'Count': 'Upanishad'})
       yoga_flat = yogasutra_combined.T.rename(index={'Count': 'YogaSutra'})
       proverb_flat = proverb_combined.T.rename(index={'Count': 'Proverb'})
       eccl_flat = eccleasiasticus_combined.T.rename(index={'Count':␣
        ↪'Eccleasiasticus'})
       wisdom_flat = wisdom_combined.T.rename(index={'Count': 'Wisdom'})
```

```
[300]: bud_flat.head()
```

```
[300]:          foolishness hath wholesome takest feelings anger vaivaswata matrix  \
       Buddhism           0    0         0      0       19     0          0      0

                kindled convict  … erred thinkest modern reigned sparingly visual  \
       Buddhism       0       0  …     0        0      0       0         0      0

                thoughts illumines attire explains
       Buddhism        9         0      0        0

       [1 rows x 8266 columns]
```

```
[414]: #combine the flattened dataframes
       df_flat = pd.concat([bud_flat, tao_flat, up_flat, yoga_flat, proverb_flat,␣
        ↪eccl_flat, wisdom_flat])
```

```
[302]: df_flat.head()
```

```
[302]:            foolishness hath wholesome takest feelings anger vaivaswata matrix  \
       Buddhism              0    0         0      0       19     0          0      0
       TaoTeChing            0    0         0      0        0     1          0      0
       Upanishad             0    0         0      0        0     3          1      0
       YogaSutra             0    2         1      0        0     0          0      1
       Proverb               2   65         0      0        0    11          0      0
```

```
           kindled convict  … erred thinkest modern reigned sparingly  \
Buddhism         0       0  …     0        0      0       0         0
TaoTeChing       0       0  …     0        0      0       0         0
Upanishad        1       0  …     0        3      0       0         0
YogaSutra        0       0  …     0        0      2       0         0
Proverb          0       0  …     0        0      0       0         0

           visual thoughts illumines attire explains
Buddhism        0        9         0      0        0
TaoTeChing      0        0         0      0        0
Upanishad       0        2         1      0        1
YogaSutra       1       14         4      0        1
Proverb         0        8         0      1        0

[5 rows x 8266 columns]
```

[415]:

```
#standardize the word counts for better analysis
df_norm = pd.DataFrame()
for i in range(7):
    temp = (df_flat.iloc[i].values - np.mean(df_flat.iloc[i].values)) / np.
 ↪std(df_flat.iloc[i].values)
    df_norm = pd.concat([df_norm, pd.DataFrame(temp, index=df_flat.columns,␣
 ↪columns=[df_flat.index[i]]).T])
```

[305]:

```
df_norm.head()
```

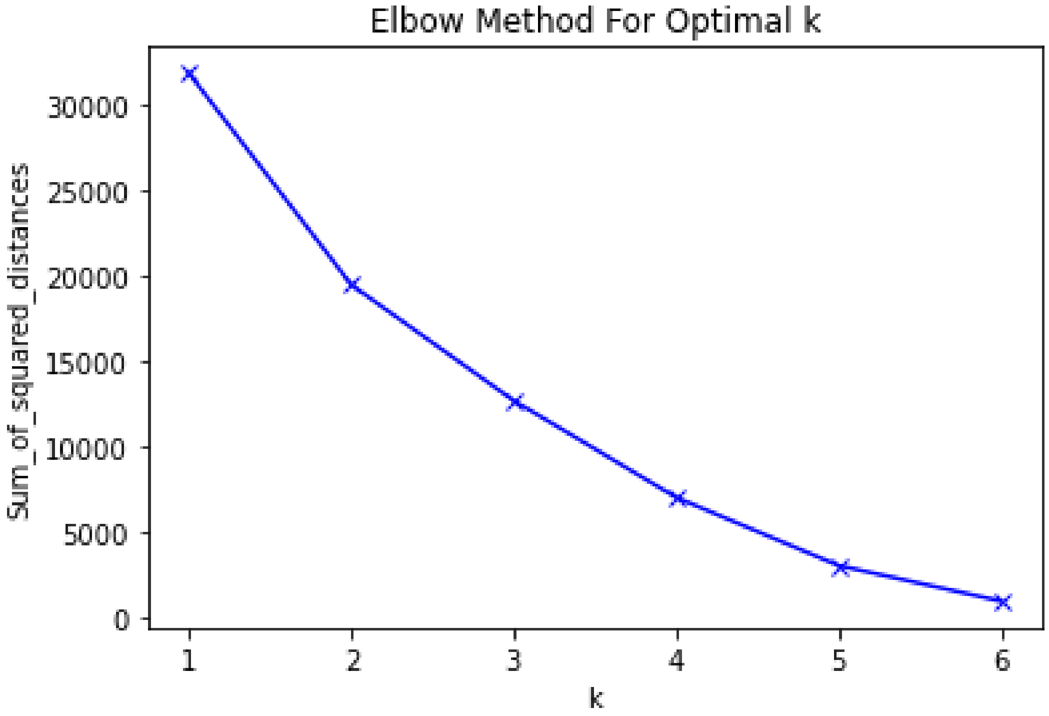
[305]:

```
           foolishness      hath wholesome    takest  feelings     anger  \
Buddhism             0         0         0         0    4.5396         0
TaoTeChing           0         0         0         0         0  0.425633
Upanishad            0         0         0         0         0  0.885231
YogaSutra    -0.149241  0.149241         0 -0.149241 -0.149241 -0.149241
Proverb       0.320025   10.4008         0         0         0   1.76014

           vaivaswata matrix   kindled   convict  …     erred  thinkest  \
Buddhism            0      0         0         0  …         0         0
TaoTeChing          0      0         0         0  …         0         0
Upanishad    0.295077      0  0.295077         0  …         0  0.885231
YogaSutra   -0.149241      0 -0.149241 -0.149241  … -0.149241 -0.149241
Proverb             0      0         0         0  …         0         0

              modern   reigned sparingly visual  thoughts illumines    attire  \
Buddhism           0         0         0      0   2.15034         0         0
TaoTeChing         0         0         0      0         0         0         0
Upanishad          0         0         0      0  0.590154  0.295077         0
YogaSutra   0.149241 -0.149241 -0.149241      0   1.94014  0.447724 -0.149241
Proverb            0         0         0      0    1.2801         0  0.160013
```

```
            explains
Buddhism           0
TaoTeChing         0
Upanishad   0.295077
YogaSutra          0
Proverb            0

[5 rows x 8266 columns]
```

[306]:

```
#prepare data for clustering algorithm
X = df_norm

#use elbow method to optimize number of clusters
Sum_of_squared_distances = []
K = range(1,7)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(X)
    Sum_of_squared_distances.append(km.inertia_)
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```

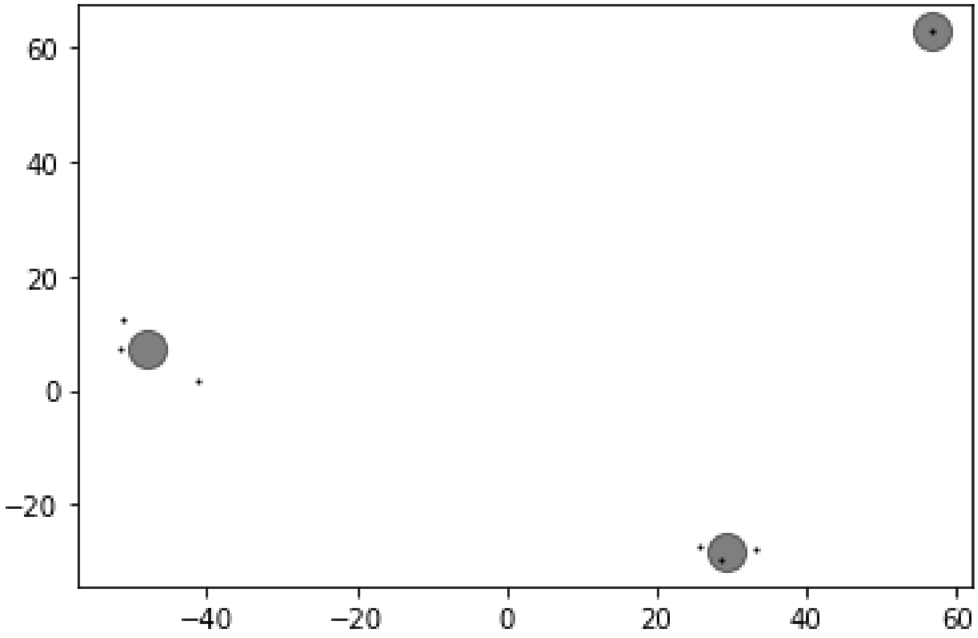


[307]:

```
#run pca on data for dimensionality reduction
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(X)

#run kmeans clustering algorithm on data
kmeans = KMeans(n_clusters=3, random_state=42).fit(reduced_data)
clusters = pd.DataFrame(index=df_norm.index, data=kmeans.labels_,␣
 ↪columns=['Category'])
```

[308]:

```
#plot books and clusters
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
plt.show()
```



[416]:

```
#find top 20 most common words in each book
top20 = []
for i in range(7):
    top20.append(df_norm.iloc[i].sort_values(ascending=False)[:20])
```

[312]:

```
top20[0]
```

[312]:

```
right            30.5825
feeling          20.3087
one              17.9195
stress           17.6805
body             17.4416
monk             17.2027
mind             16.9638
remains          15.0523
cessation        14.8134
called           14.8134
mental           13.8577
discerns         13.8577
focused          13.3799
way              13.1409
consciousness    11.2295
noble            10.9906
property         10.2738
qualities        10.0349
concentration    9.55705
form             9.31812
Name: Buddhism, dtype: object
```

[417]:

```
#create dataframe of top 20 words for each book
top20_df = pd.DataFrame()
for book in top20:
    top20_df = pd.concat([top20_df, pd.DataFrame(book.index, columns=[book.
 ↪name])], axis=1)
```

[418]:

```
top20_df
```

[418]:

```
         Buddhism TaoTeChing   Upanishad      YogaSutra Proverb  \
0           right        tao         one      spiritual   shall
1         feeling     things        self            man     man
2             one        one        mind           life     thy
3          stress        men     brahman  consciousness    thou
4            body      great         man          power  wicked
5            monk  therefore       death            one    lord
6            mind     heaven   knowledge           mind    wise
7         remains      would        know           soul    hath
8          called       thus        must         things   heart
9       cessation    without  nachiketas           self    thee
10         mental     people        said         powers     way
11       discerns       sage      senses        psychic    evil
12        focused       know      beyond            may  wisdom
13            way        yet       atman          first   mouth
14  consciousness      state      nature           must    soul
15          noble        way       knows          comes     son
```

```
16       property       like   therefore      psychical   words
17      qualities        may       heart         divine    good
18  concentration      place         god           body    fool
19   fabrications       name        body        eternal  things

   Eccleasiasticus     Wisdom
0            shall      shall
1              thy     things
2              man        thy
3             thou        god
4              god       thou
5             hath     wisdom
6             thee        man
7             lord       upon
8           things       made
9             upon       hath
10          wisdom       thee
11           heart        men
12            good       lord
13             men         us
14            fear       life
15            soul  therefore
16             one       good
17           shalt     wicked
18           glory      might
19            give   children
```

[419]:
```
#find top 20 words in all books combined
temp = (words_combined.values - np.mean(words_combined.values)) / np.
 ↪std(words_combined.values)
words_norm = pd.DataFrame(temp, index=words_combined.index, columns=['z'])
pd.DataFrame(words_norm.sort_values(by=['z'], ascending=False)[:20].index,␣
 ↪columns=['Words'])
```

[419]:
```
        Words
0       shall
1         man
2         thy
3         one
4      things
5        thou
6         god
7        life
8        hath
9   spiritual
10       lord
11       mind
```

```
12       thee
13      heart
14       soul
15     wisdom
16        men
17       upon
18       good
19        way
```

```
[323]: #look at the top 20 words for only the old testament books
       oldtest_top20 = top20_df[['Proverb', 'Eccleasiasticus', 'Wisdom']]
       oldtest_top20.head()
```

```
[323]:   Proverb Eccleasiasticus   Wisdom
       0   shall           shall    shall
       1     man             thy   things
       2     thy             man      thy
       3    thou            thou      god
       4  wicked             god     thou
```

```
[320]: #create dataframe of all old testmanet books
       #drop words that are not in any of the old testament books
       oldtest = pd.concat([proverb, eccleasiasticus, wisdom])
       oldtest = oldtest.set_index(oldtest['Books'])
       oldtest.drop(columns=['Books'], inplace=True)
       oldtest_clean = oldtest.drop(columns=words[np.where(oldtest.sum() == 0)])
```

```
[322]: oldtest_clean.head()
```

```
[322]:                    foolishness  hath  wholesome  takest  anger  kindled  \
       Books
       BookOfProverb_Ch1            0     0          0       0      0        0
       BookOfProverb_Ch2            0     1          0       0      0        0
       BookOfProverb_Ch3            0     4          0       0      0        0
       BookOfProverb_Ch4            0     0          0       0      0        0
       BookOfProverb_Ch5            0     1          0       0      0        0

                          convict  diadem  open  expecteth  …  admireth  lifeless  \
       Books                                                …
       BookOfProverb_Ch1        0       0     0          0  …         0         0
       BookOfProverb_Ch2        0       0     0          0  …         0         0
       BookOfProverb_Ch3        0       0     0          0  …         0         0
       BookOfProverb_Ch4        0       0     0          0  …         0         0
       BookOfProverb_Ch5        0       0     0          0  …         0         0

                          stout  taketh  kettle  erred  reigned  sparingly  thoughts  \
       Books
```

```
BookOfProverb_Ch1      0       0       0      0        0          0         0
BookOfProverb_Ch2      0       0       0      0        0          0         0
BookOfProverb_Ch3      0       0       0      0        0          0         0
BookOfProverb_Ch4      0       0       0      0        0          0         0
BookOfProverb_Ch5      0       0       0      0        0          0         1

                   attire
Books
BookOfProverb_Ch1       0
BookOfProverb_Ch2       0
BookOfProverb_Ch3       0
BookOfProverb_Ch4       0
BookOfProverb_Ch5       0

[5 rows x 4343 columns]
```

[327]:

```
#standardize the old testament data
oldtest_norm = pd.DataFrame()
for i in range(len(oldtest)):
    temp = (oldtest_clean.iloc[i].values - np.mean(oldtest_clean.iloc[i].
 ↪values)) / np.std(oldtest_clean.iloc[i].values)
    oldtest_norm = pd.concat([oldtest_norm, pd.DataFrame(temp,␣
 ↪index=oldtest_clean.columns, columns=[oldtest_clean.iloc[i].name]).T])
```

[328]:

```
oldtest_norm.head()
```

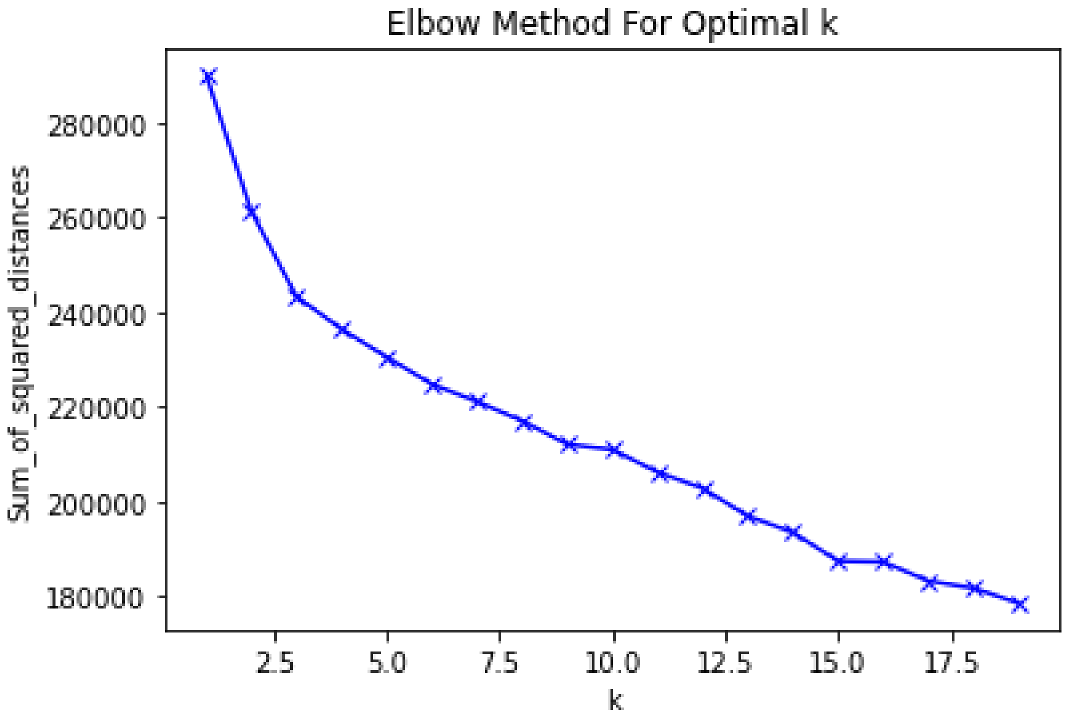
[328]:

```
                   foolishness      hath  wholesome    takest     anger  \
BookOfProverb_Ch1    -0.131811 -0.131811  -0.131811 -0.131811 -0.131811
BookOfProverb_Ch2    -0.119582  2.935389  -0.119582 -0.119582 -0.119582
BookOfProverb_Ch3    -0.117459  7.356880  -0.117459 -0.117459 -0.117459
BookOfProverb_Ch4    -0.124461 -0.124461  -0.124461 -0.124461 -0.124461
BookOfProverb_Ch5    -0.124342  2.810546  -0.124342 -0.124342 -0.124342

                    kindled   convict    diadem      open  expecteth  …  \
BookOfProverb_Ch1 -0.131811 -0.131811 -0.131811 -0.131811  -0.131811  …
BookOfProverb_Ch2 -0.119582 -0.119582 -0.119582 -0.119582  -0.119582  …
BookOfProverb_Ch3 -0.117459 -0.117459 -0.117459 -0.117459  -0.117459  …
BookOfProverb_Ch4 -0.124461 -0.124461 -0.124461 -0.124461  -0.124461  …
BookOfProverb_Ch5 -0.124342 -0.124342 -0.124342 -0.124342  -0.124342  …

                   admireth  lifeless     stout    taketh   kettle     erred  \
BookOfProverb_Ch1 -0.131811 -0.131811 -0.131811 -0.131811 -0.131811 -0.131811
BookOfProverb_Ch2 -0.119582 -0.119582 -0.119582 -0.119582 -0.119582 -0.119582
BookOfProverb_Ch3 -0.117459 -0.117459 -0.117459 -0.117459 -0.117459 -0.117459
BookOfProverb_Ch4 -0.124461 -0.124461 -0.124461 -0.124461 -0.124461 -0.124461
BookOfProverb_Ch5 -0.124342 -0.124342 -0.124342 -0.124342 -0.124342 -0.124342
```

```
                     reigned  sparingly  thoughts    attire
BookOfProverb_Ch1 -0.131811  -0.131811 -0.131811 -0.131811
BookOfProverb_Ch2 -0.119582  -0.119582 -0.119582 -0.119582
BookOfProverb_Ch3 -0.117459  -0.117459 -0.117459 -0.117459
BookOfProverb_Ch4 -0.124461  -0.124461 -0.124461 -0.124461
BookOfProverb_Ch5 -0.124342  -0.124342  2.810546 -0.124342

[5 rows x 4343 columns]
```

[335]:

```python
#prepare old testament data for clustering
X2 = oldtest_norm

#use elbow method to optimize number of clusters
Sum_of_squared_distances2 = []
K2 = range(1,20)
for k in K2:
    km = KMeans(n_clusters=k)
    km = km.fit(X2)
    Sum_of_squared_distances2.append(km.inertia_)
plt.plot(K2, Sum_of_squared_distances2, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



[336]:

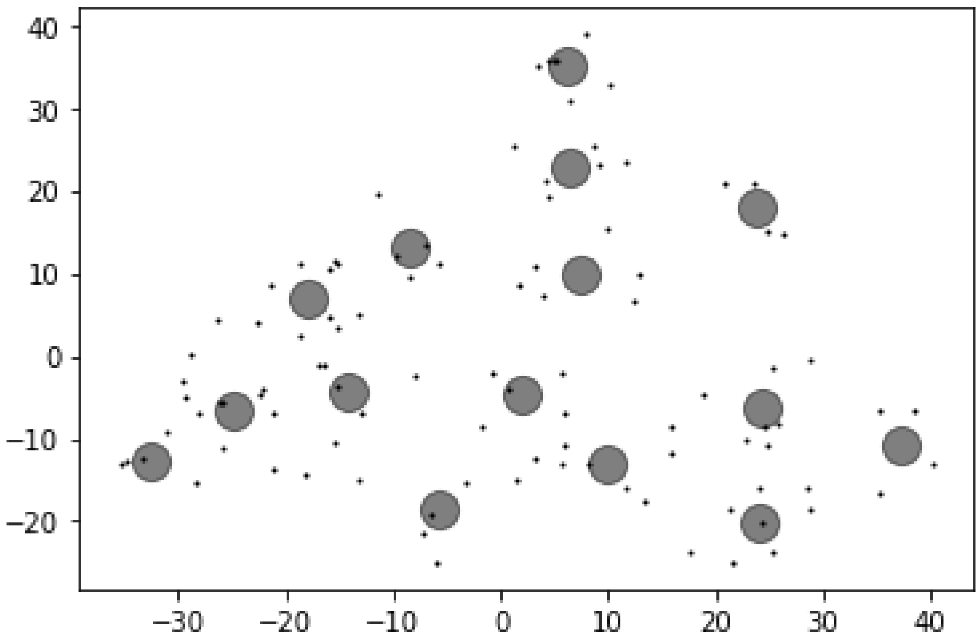
```python
#run pca on old testament data for dimensionality reduction
pca2 = PCA(n_components=2)
reduced_data2 = pca.fit_transform(X2)

#run kmeans clustering algorithm on old testament data
kmeans2 = KMeans(n_clusters=15, random_state=42).fit(reduced_data2)
clusters2 = pd.DataFrame(index=oldtest_norm.index, data=kmeans2.labels_,␣
 ↪columns=['Category'])
clusters2
```

[336]:

```
                   Category
BookOfProverb_Ch1         3
BookOfProverb_Ch2         1
BookOfProverb_Ch3        10
BookOfProverb_Ch4        10
BookOfProverb_Ch5        13
…                      …
BookOfWisdom_Ch15         4
BookOfWisdom_Ch16         5
BookOfWisdom_Ch17         9
BookOfWisdom_Ch18        11
BookOfWisdom_Ch19        11

[100 rows x 1 columns]
```

[337]:

```python
#plot old testament chapters and clusters
centers2 = kmeans2.cluster_centers_
plt.scatter(centers2[:, 0], centers2[:, 1], c='black', s=200, alpha=0.5)
plt.plot(reduced_data2[:, 0], reduced_data2[:, 1], 'k.', markersize=2)
plt.show()
```



[388]:

```
#clean data set by dropping words of length 1 and the word nt
#this is because the data was likely split on all punctuation, including␣
↪apostrophes
#therefore the letter s, which is in the data set, and the letters nt are not␣
↪actual words and can be removed
words_clean = []
for w in words:
    if len(w) != 1 and w != 'nt':
        words_clean.append(w)
words_clean = np.array(words_clean)
df_clean = df[words_clean]
#remove words that have fewer than 5 instances
#this is because it is unlikely for such rare words to be important in our␣
↪analysis
df_clean.drop(columns=words_clean[np.where(df_clean.sum().values < 5)],␣
↪inplace=True)
df_clean['Books'] = df['Books'].values
```

```
c:\users\l\appdata\local\programs\python\python37\lib\site-
packages\pandas\core\frame.py:4170: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
c:\users\l\appdata\local\programs\python\python37\lib\site-
```

```
packages\ipykernel_launcher.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  del sys.path[0]
```

[389]:

```
df_clean.head()
```

[389]:

```
   hath  feelings  anger  open  rage  looketh  illumination  tell  build  \
0     0         0      0     1     0        0             0     0      0
1     0         0      0     0     0        0             0     0      0
2     0         0      0     0     0        0             0     1      0
3     0         0      0     0     0        0             1     0      0
4     0         0      0     0     0        0             0     0      0

   neither  …  needs  well  state  production  developed  regarded  taketh  \
0        0  …      0     0      0           0          0         0       0
1        0  …      0     0      0           0          0         0       0
2        0  …      0     0      0           0          0         0       0
3        0  …      0     0      0           0          2         0       0
4        0  …      0     0      0           0          0         0       0

   thoughts  illumines         Books
0         0          0  Buddhism_Ch1
1         0          0  Buddhism_Ch2
2         0          0  Buddhism_Ch3
3         0          0  Buddhism_Ch4
4         0          0  Buddhism_Ch5

[5 rows x 2199 columns]
```

[390]:

```
#fix the order of the columns
cols = df_clean.columns.tolist()
cols = cols[-1:] + cols[:-1]
df_clean = df_clean[cols]

#get new list of words
words_clean = df_clean.columns[1:]
```

[391]:

```
#create new old testament data based on the cleaned data frame
oldtest_temp = df_clean[(df_clean['Books'].str.
 ↪contains('Proverb|Eccleasiasticus|Wisdom'))]
oldtest_final = oldtest_temp.drop(columns=['Books'])

#remove words that occur zero times in the old testament
```

```
oldtest_final = oldtest_final.drop(columns=words_clean[np.where(oldtest_final.
 ↪sum() == 0)])
oldtest_final.reset_index(drop=True, inplace=True)
```

[392]:

```
oldtest_final.head()
```

[392]:

```
   hath  anger  open  rage  looketh  tell  build  neither  soft  land  …  \
0     0      0     0     0        0     0      0        0     0     0  …
1     1      0     0     0        0     0      0        1     0     0  …
2     4      0     0     0        0     0      0        0     0     0  …
3     0      0     0     0        0     0      0        2     0     0  …
4     1      0     0     0        0     0      0        1     0     0  …

   walketh  business  red  grows  glad  well  state  regarded  taketh  \
0        0         0    0      0     0     0      0         1       0
1        0         0    0      0     1     0      0         0       0
2        0         0    0      0     0     0      0         0       0
3        0         0    0      0     0     0      0         0       0
4        0         0    0      0     0     1      0         0       0

   thoughts
0         0
1         0
2         0
3         0
4         1

[5 rows x 1604 columns]
```

[393]:

```
#standardize the old testmanet data
oldtest_final_norm = pd.DataFrame()
oldtest_final_words = oldtest_final
for i in range(len(oldtest_final)):
    temp = (oldtest_final_words.iloc[i].values - np.mean(oldtest_final_words.
 ↪iloc[i].values)) / np.std(oldtest_final_words.iloc[i].values)
    oldtest_final_norm = pd.concat([oldtest_final_norm, pd.DataFrame(temp,␣
 ↪index=oldtest_final_words.columns).T])
```

[394]:

```
#add books back to dataframe
oldtest_final_norm['Books'] = oldtest_clean.index
```

[395]:

```
#fix the order of columns
cols = oldtest_final_norm.columns.tolist()
cols = cols[-1:] + cols[:-1]
oldtest_final_norm = oldtest_final_norm[cols]
```

[396]:

```
oldtest_final_norm.head()
```

[396]:

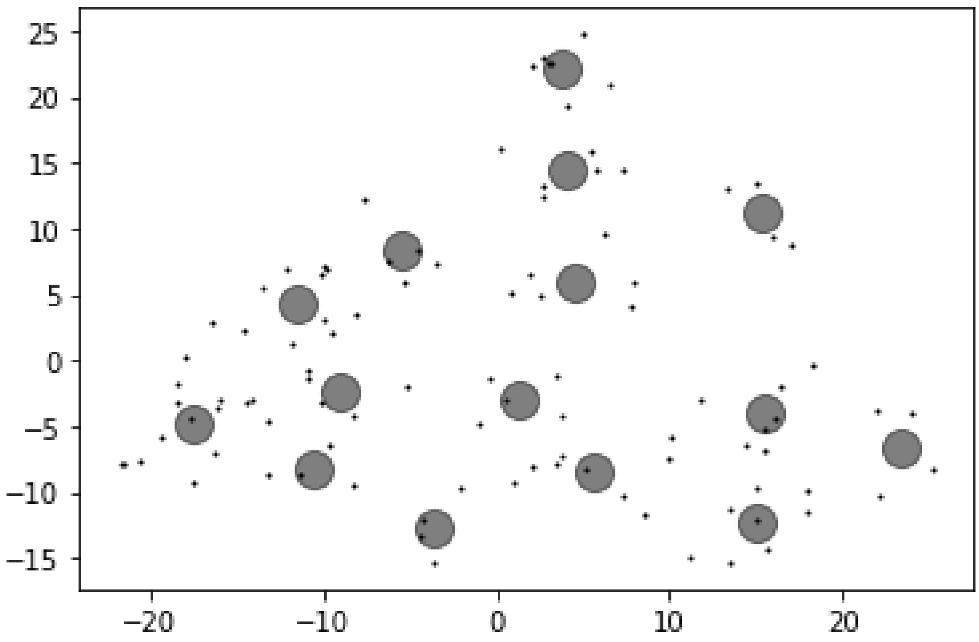
```
                Books      hath     anger      open      rage   looketh  \
0  BookOfProverb_Ch1 -0.199877 -0.199877 -0.199877 -0.199877 -0.199877
0  BookOfProverb_Ch2  1.696387 -0.193200 -0.193200 -0.193200 -0.193200
0  BookOfProverb_Ch3  4.456197 -0.176919 -0.176919 -0.176919 -0.176919
0  BookOfProverb_Ch4 -0.195927 -0.195927 -0.195927 -0.195927 -0.195927
0  BookOfProverb_Ch5  1.664622 -0.181888 -0.181888 -0.181888 -0.181888

       tell     build   neither      soft  …   walketh  business       red  \
0 -0.199877 -0.199877 -0.199877 -0.199877  … -0.199877 -0.199877 -0.199877
0 -0.193200 -0.193200  1.696387 -0.193200  … -0.193200 -0.193200 -0.193200
0 -0.176919 -0.176919 -0.176919 -0.176919  … -0.176919 -0.176919 -0.176919
0 -0.195927 -0.195927  2.572940 -0.195927  … -0.195927 -0.195927 -0.195927
0 -0.181888 -0.181888  1.664622 -0.181888  … -0.181888 -0.181888 -0.181888

      grows      glad      well     state  regarded    taketh  thoughts
0 -0.199877 -0.199877 -0.199877 -0.199877  1.141555 -0.199877 -0.199877
0 -0.193200  1.696387 -0.193200 -0.193200 -0.193200 -0.193200 -0.193200
0 -0.176919 -0.176919 -0.176919 -0.176919 -0.176919 -0.176919 -0.176919
0 -0.195927 -0.195927 -0.195927 -0.195927 -0.195927 -0.195927 -0.195927
0 -0.181888 -0.181888  1.664622 -0.181888 -0.181888 -0.181888  1.664622

[5 rows x 1605 columns]
```

[398]:

```
#run clustering algorithm on data frame
pca_oldtest = PCA(n_components=2)
reduced_data_oldtest = pca.fit_transform(oldtest_final_norm.
 ↪drop(columns=['Books']))
kmeans_oldtest = KMeans(n_clusters=15, random_state=42).
 ↪fit(reduced_data_oldtest)
clusters_oldtest = pd.DataFrame(index=oldtest_final_norm.index,␣
 ↪data=kmeans_oldtest.labels_, columns=['Category'])
```

[399]:

```
#plot clusters and chapters in the old testament
centers_oldtest = kmeans_oldtest.cluster_centers_
plt.scatter(centers_oldtest[:, 0], centers_oldtest[:, 1], c='black', s=200,␣
 ↪alpha=0.5)
plt.plot(reduced_data_oldtest[:, 0], reduced_data_oldtest[:, 1], 'k.',␣
 ↪markersize=2)
plt.show()
```



```
[400]: #create dataframe of coordinates of each book, and its category, for easier␣
        ↪visualization
       oldtest_coords_clusters = pd.DataFrame([clusters_oldtest.index, [item[0] for␣
        ↪item in reduced_data_oldtest], [item[1] for item in reduced_data_oldtest],␣
        ↪clusters_oldtest['Category'].values]).T
       oldtest_coords_clusters.columns = [['Book', 'x', 'y', 'Category']]
```

```
[401]: #create new cluster names
       cluster_names = []
       for i in range(15):
           cluster_names.append('Center of Cluster ' + str(i))
```

```
[402]: #create dataframe of cluster coordinates for easier visualization
       cluster_coords = pd.DataFrame([cluster_names, [item[0] for item in␣
        ↪centers_oldtest], [item[1] for item in centers_oldtest]]).T
       cluster_coords.columns = [['Category', 'x', 'y']]
```

```
  [4]: #load in old testament dataframe for tone analysis
       df = pd.read_csv('oldtest.csv')
       df.rename(columns={'Unnamed: 0': 'Book'}, inplace=True)
```

```
  [5]: df.head()
```

```
  [5]:                 Book      hath     anger      open      rage   looketh  \
       0  BookOfProverb_Ch1 -0.199877 -0.199877 -0.199877 -0.199877 -0.199877
```

```
1  BookOfProverb_Ch2  1.696387 -0.193200 -0.193200 -0.193200 -0.193200
2  BookOfProverb_Ch3  4.456197 -0.176919 -0.176919 -0.176919 -0.176919
3  BookOfProverb_Ch4 -0.195927 -0.195927 -0.195927 -0.195927 -0.195927
4  BookOfProverb_Ch5  1.664622 -0.181888 -0.181888 -0.181888 -0.181888

       tell     build   neither      soft  …   walketh  business       red  \
0 -0.199877 -0.199877 -0.199877 -0.199877  … -0.199877 -0.199877 -0.199877
1 -0.193200 -0.193200  1.696387 -0.193200  … -0.193200 -0.193200 -0.193200
2 -0.176919 -0.176919 -0.176919 -0.176919  … -0.176919 -0.176919 -0.176919
3 -0.195927 -0.195927  2.572940 -0.195927  … -0.195927 -0.195927 -0.195927
4 -0.181888 -0.181888  1.664622 -0.181888  … -0.181888 -0.181888 -0.181888

      grows      glad      well     state  regarded    taketh  thoughts
0 -0.199877 -0.199877 -0.199877 -0.199877  1.141555 -0.199877 -0.199877
1 -0.193200  1.696387 -0.193200 -0.193200 -0.193200 -0.193200 -0.193200
2 -0.176919 -0.176919 -0.176919 -0.176919 -0.176919 -0.176919 -0.176919
3 -0.195927 -0.195927 -0.195927 -0.195927 -0.195927 -0.195927 -0.195927
4 -0.181888 -0.181888  1.664622 -0.181888 -0.181888 -0.181888  1.664622

[5 rows x 1605 columns]
```

[21]:

```
#initialize ibm watson
authenticator = IAMAuthenticator('5G41D_Li4QuGa-DkQwJD5Dl3CBHzBcV6BTSLwlyHufcQ')
tone_analyzer = ToneAnalyzerV3(
    version='2017-09-21',
    authenticator=authenticator
)

tone_analyzer.set_service_url('https://api.us-south.tone-analyzer.watson.cloud.
 ↪ibm.com/instances/bf993a7b-7133-4d4e-9977-f6467fc4cc52')
```

[12]:

```
#get list of all words in the books
words = np.array(df.columns[1:])
```

[40]:

```
#run tone analysis on words
tone_lst = []
for i in words:
    tone_analysis = tone_analyzer.tone(
        {'text': i},
        content_type='text/plain'
    ).get_result()
    tone_lst.append((i, tone_analysis['document_tone']['tones']))
```

[219]:

```
#drop words that have no associated tone
#create list of tuples containing word and its associated tone
real_tone_lst = []
for i in tone_lst:
    if len(i[1]) > 0:
            temp_score = 0
            temp_tone = ''
            for j in i[1]:
                if j['score'] > temp_score:
                    temp_score = j['score']
                    temp_tone = j['tone_name']
            real_tone_lst.append((i[0], temp_tone))
```

[93]:

```
#get list of all words that have associated tone
tone_words = []
for i in real_tone_lst:
    tone_words.append(i[0])
tone_words.insert(0, 'Book') #keep book in the words for easier column indexing
tone_words = np.array(tone_words)
```

[100]:

```
#only keep words with an associated tone
df = df[tone_words]
```

[101]:

```
df.head()
```

[101]:

```
                Book     anger      rage     build   neither       let  \
0  BookOfProverb_Ch1 -0.199877 -0.199877 -0.199877 -0.199877  5.165852
1  BookOfProverb_Ch2 -0.193200 -0.193200 -0.193200  1.696387 -0.193200
2  BookOfProverb_Ch3 -0.176919 -0.176919 -0.176919 -0.176919  3.297918
3  BookOfProverb_Ch4 -0.195927 -0.195927 -0.195927  2.572940  8.110672
4  BookOfProverb_Ch5 -0.181888 -0.181888 -0.181888  1.664622  9.050663

       felt     great   embrace   violent  …    secret   incline    spread  \
0 -0.199877 -0.199877 -0.199877 -0.199877  … -0.199877 -0.199877  1.141555
1 -0.193200 -0.193200 -0.193200 -0.193200  … -0.193200  3.585974 -0.193200
2 -0.176919 -0.176919 -0.176919 -0.176919  … -0.176919 -0.176919 -0.176919
3 -0.195927 -0.195927  1.188506 -0.195927  … -0.195927  1.188506 -0.195927
4 -0.181888 -0.181888 -0.181888 -0.181888  … -0.181888  1.664622 -0.181888

        cry  necessity       set   fearful      glad     state  thoughts
0 -0.199877  -0.199877 -0.199877 -0.199877 -0.199877 -0.199877 -0.199877
1 -0.193200  -0.193200 -0.193200 -0.193200  1.696387 -0.193200 -0.193200
2 -0.176919  -0.176919 -0.176919 -0.176919 -0.176919 -0.176919 -0.176919
3 -0.195927  -0.195927 -0.195927 -0.195927 -0.195927 -0.195927 -0.195927
4 -0.181888  -0.181888 -0.181888 -0.181888 -0.181888 -0.181888  1.664622

[5 rows x 486 columns]
```

[102]:

```
#get unique tones
tones = []
for i in real_tone_lst:
    tones.append(i[1])
tones = np.array((set(tones)))
```

[112]:

```
#create dictionary containing tones and its values for each book
tone_dct = {'Sadness': np.zeros(len(df)), 'Joy': np.zeros(len(df)), 'Anger': np.
 ↪zeros(len(df)), 'Analytical': np.zeros(len(df)), 'Tentative': np.
 ↪zeros(len(df)), 'Confident': np.zeros(len(df)), 'Fear': np.zeros(len(df))}
for i in real_tone_lst:
    word = i[0]
    tone = i[1]
    tone_dct[tone] += df[word].values
```

[114]:

```
#add columns of tone values
df['SADNESS'] = tone_dct['Sadness']
df['JOY'] = tone_dct['Joy']
df['ANGER'] = tone_dct['Anger']
df['ANALYTICAL'] = tone_dct['Analytical']
df['TENTATIVE'] = tone_dct['Tentative']
df['CONFIDENT'] = tone_dct['Confident']
df['FEAR'] = tone_dct['Fear']
```

```
c:\users\l\appdata\local\programs\python\python37\lib\site-
packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
c:\users\l\appdata\local\programs\python\python37\lib\site-
packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

c:\users\l\appdata\local\programs\python\python37\lib\site-
packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
c:\users\l\appdata\local\programs\python\python37\lib\site-
```

```
packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
c:\users\l\appdata\local\programs\python\python37\lib\site-
packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
c:\users\l\appdata\local\programs\python\python37\lib\site-
packages\ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

c:\users\l\appdata\local\programs\python\python37\lib\site-
packages\ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
```

[256]:

```
#create pandas series for each book and its total tone
prov_tone = df[df['Book'].str.contains('Proverb')][['SADNESS', 'ANGER', 'JOY',␣
 ↪'ANALYTICAL', 'TENTATIVE', 'CONFIDENT', 'FEAR']].sum()
eccl_tone = df[df['Book'].str.contains('Eccleasiasticus')][['SADNESS', 'ANGER',␣
 ↪'JOY', 'ANALYTICAL', 'TENTATIVE', 'CONFIDENT', 'FEAR']].sum()
wis_tone = df[df['Book'].str.contains('Wisdom')][['SADNESS', 'ANGER', 'JOY',␣
 ↪'ANALYTICAL', 'TENTATIVE', 'CONFIDENT', 'FEAR']].sum()
```

[262]:

```
#function to normalize data on scale from 0 to 1
def NormalizeData(data):
    return (data - np.min(data)) / (np.max(data) - np.min(data))
```

[267]:

```
#normalize data
prov_tone = NormalizeData(prov_tone.values)
eccl_tone = NormalizeData(eccl_tone.values)
```

```
wis_tone = NormalizeData(wis_tone.values)
```

[280]:
```
#turn series into dataframes
prov_df = pd.DataFrame(prov_tone).T
prov_df.columns = ['Sadness', 'Anger', 'Joy', 'Analytical', 'Tentative',␣
 ↪'Confident', 'Fear']

eccl_df = pd.DataFrame(eccl_tone).T
eccl_df.columns = ['Sadness', 'Anger', 'Joy', 'Analytical', 'Tentative',␣
 ↪'Confident', 'Fear']

wis_df = pd.DataFrame(wis_tone).T
wis_df.columns = ['Sadness', 'Anger', 'Joy', 'Analytical', 'Tentative',␣
 ↪'Confident', 'Fear']
```

[404]:
```
prov_df
```

[404]:
```
    Sadness     Anger  Joy  Analytical  Tentative  Confident     Fear
0  0.599071  0.690424  1.0         0.0   0.521823   0.488139  0.51443
```

[405]:
```
eccl_df
```

[405]:
```
    Sadness     Anger  Joy  Analytical  Tentative  Confident      Fear
0  0.516486  0.550526  1.0         0.0   0.540251   0.552315  0.548456
```

[406]:
```
wis_df
```

[406]:
```
    Sadness     Anger  Joy  Analytical  Tentative  Confident      Fear
0  0.266508  0.342749  1.0         0.0   0.557436   0.307545  0.307791
```

[281]:
```
#combines all dataframes into one
all_df = pd.concat([prov_df, eccl_df, wis_df])
```

[283]:
```
#keep track of which book is which
all_df['Books'] = ['Proverbs', 'Eccleasiasticus', 'Wisdom']
```

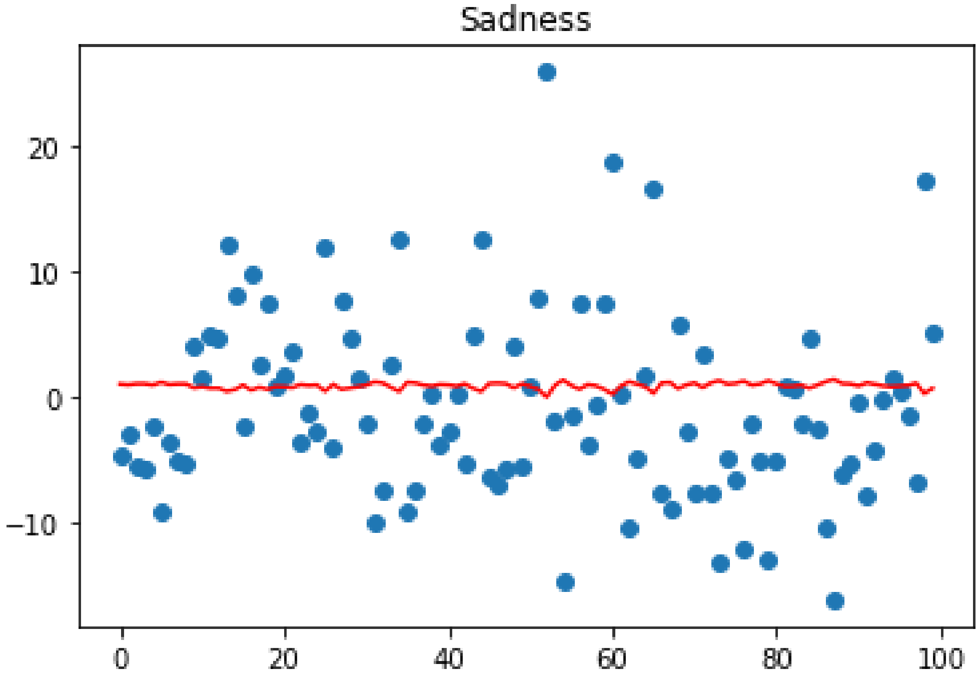
[407]:
```
all_df
```

[407]:
```
    Sadness     Anger  Joy  Analytical  Tentative  Confident      Fear  \
0  0.599071  0.690424  1.0         0.0   0.521823   0.488139  0.514430
0  0.516486  0.550526  1.0         0.0   0.540251   0.552315  0.548456
0  0.266508  0.342749  1.0         0.0   0.557436   0.307545  0.307791

             Books
0         Proverbs
0  Eccleasiasticus
0           Wisdom
```

This section is an attempt at linear regression on tone values. The tone values are too scattered for a linear regression model to accurately make predictions, as shown by the roughly horizontal regression lines in the graphs

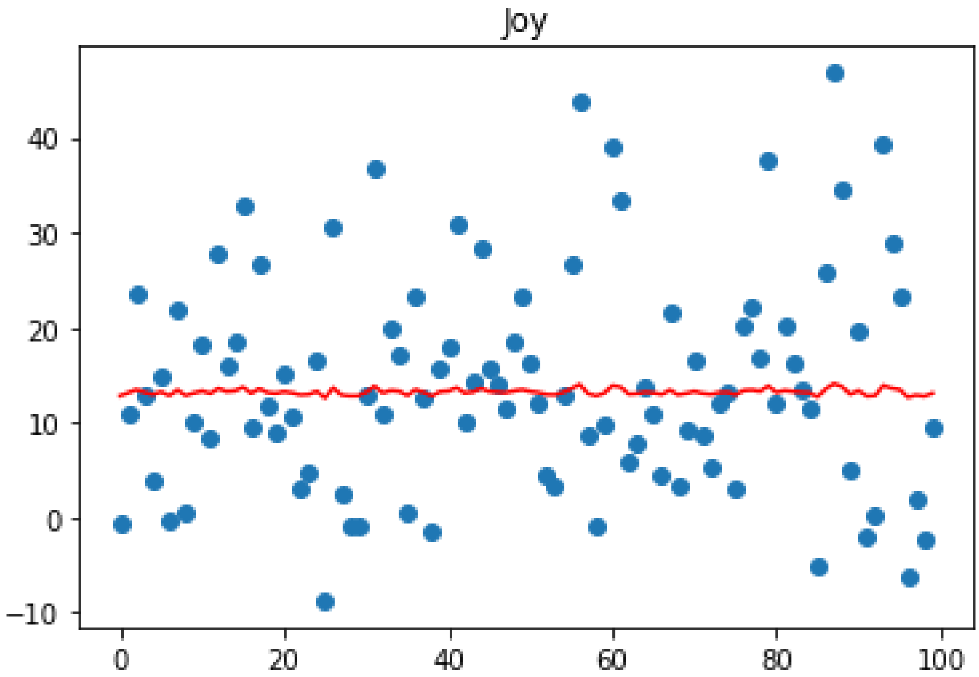
[197]:

```
X = np.array(df.index).reshape(-1, 1)
y = df['SADNESS'].values
reg = LinearRegression().fit(X, y)
reg_pred = reg.predict(y.reshape(-1, 1))

plt.scatter(np.array(df.index), df['SADNESS'])
plt.plot(X, reg_pred, color='red')
plt.title('Sadness')
plt.show()
```

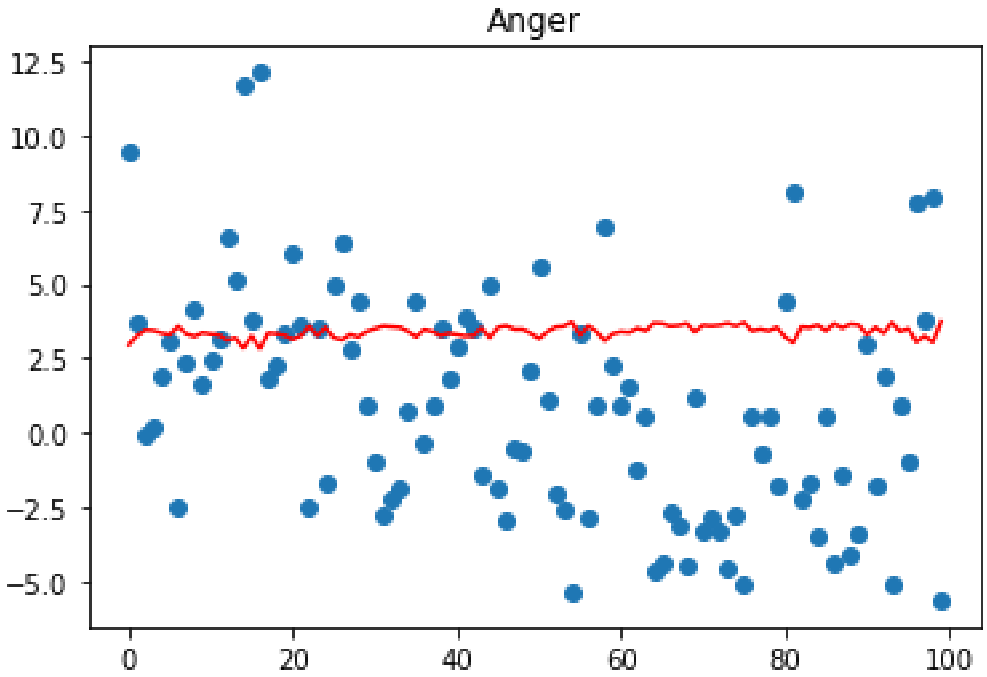


[198]:

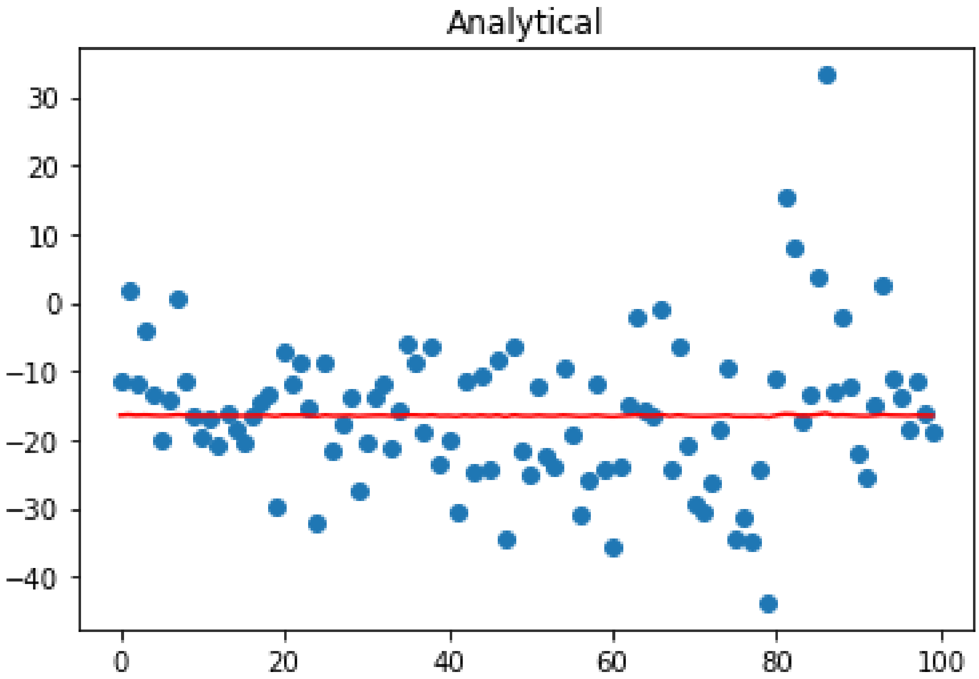
```
X = np.array(df.index).reshape(-1, 1)
y = df['JOY'].values
reg = LinearRegression().fit(X, y)
reg_pred = reg.predict(y.reshape(-1, 1))

plt.scatter(np.array(df.index), df['JOY'])
plt.plot(X, reg_pred, color='red')
plt.title('Joy')
plt.show()
```



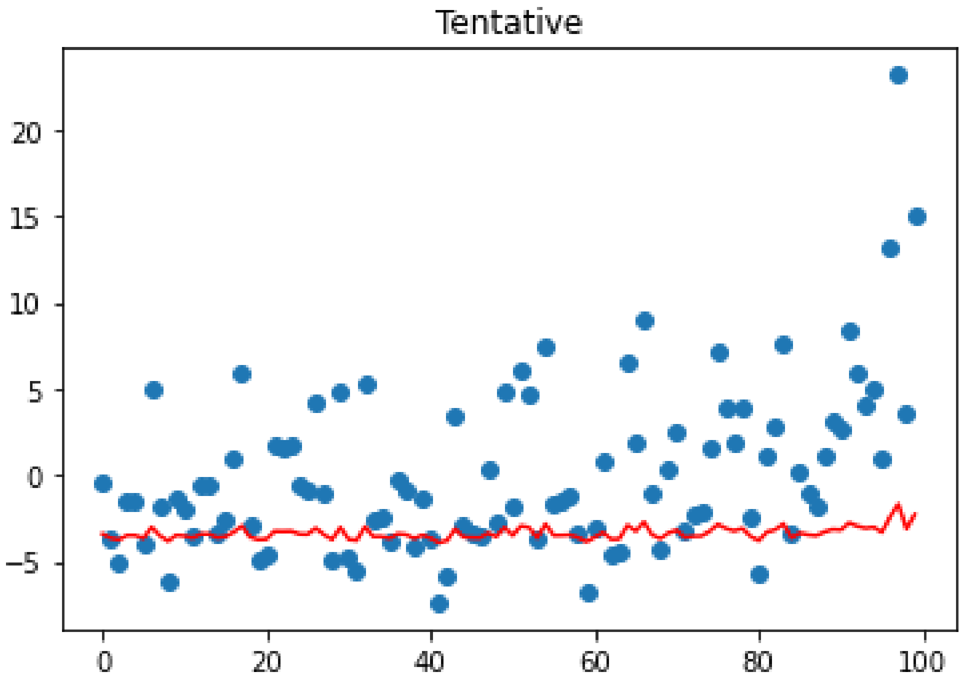
[199]:
```
X = np.array(df.index).reshape(-1, 1)
y = df['ANGER'].values
reg = LinearRegression().fit(X, y)
reg_pred = reg.predict(y.reshape(-1, 1))

plt.scatter(np.array(df.index), df['ANGER'])
plt.plot(X, reg_pred, color='red')
plt.title('Anger')
plt.show()
```



[200]:
```
X = np.array(df.index).reshape(-1, 1)
y = df['ANALYTICAL'].values
reg = LinearRegression().fit(X, y)
reg_pred = reg.predict(y.reshape(-1, 1))

plt.scatter(np.array(df.index), df['ANALYTICAL'])
plt.plot(X, reg_pred, color='red')
plt.title('Analytical')
plt.show()
```

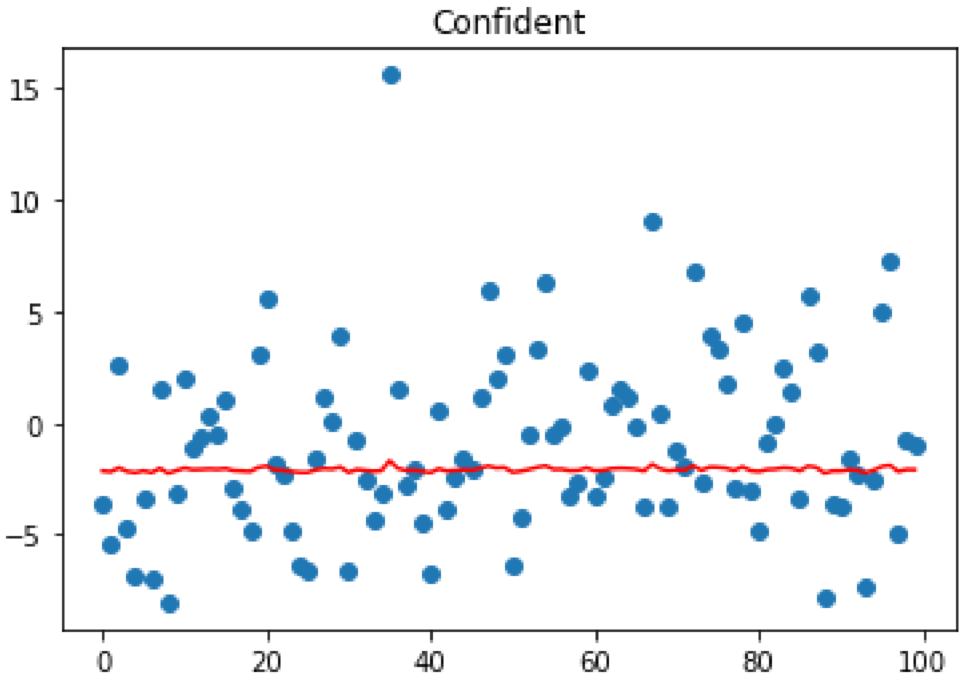


[202]:
```
X = np.array(df.index).reshape(-1, 1)
y = df['TENTATIVE'].values
reg = LinearRegression().fit(X, y)
reg_pred = reg.predict(y.reshape(-1, 1))

plt.scatter(np.array(df.index), df['TENTATIVE'])
plt.plot(X, reg_pred, color='red')
plt.title('Tentative')
plt.show()
```



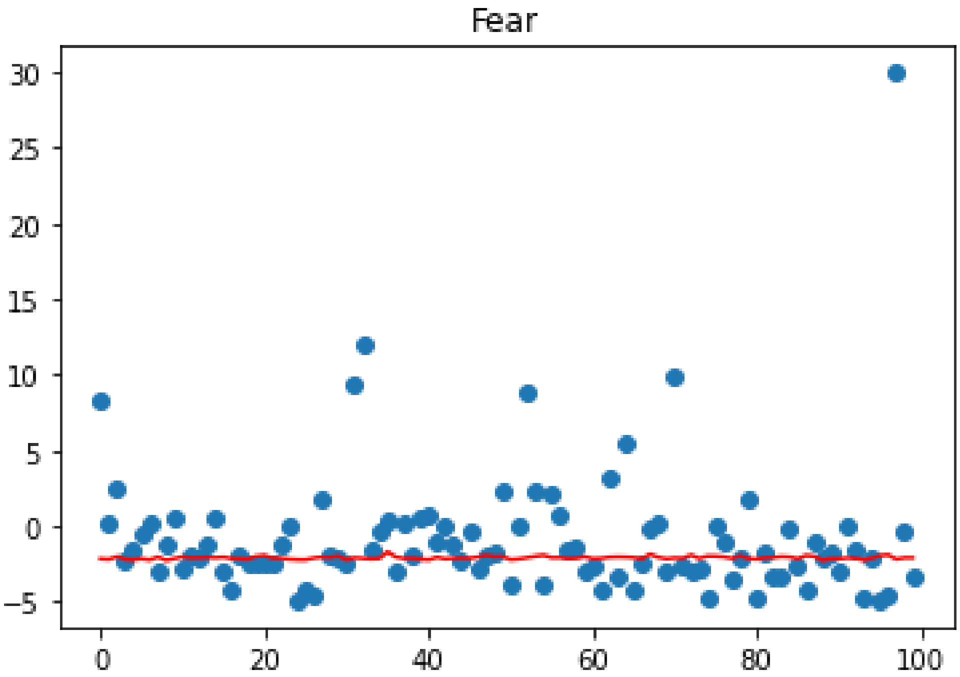
[205]:

```
X = np.array(df.index).reshape(-1, 1)
y = df['CONFIDENT'].values
reg = LinearRegression().fit(X, y)
reg_pred = reg.predict(y.reshape(-1, 1))

plt.scatter(np.array(df.index), df['CONFIDENT'])
plt.plot(X, reg_pred, color='red')
plt.title('Confident')
plt.show()
```



[206]:

```
X = np.array(df.index).reshape(-1, 1)
y = df['CONFIDENT'].values
reg = LinearRegression().fit(X, y)
reg_pred = reg.predict(y.reshape(-1, 1))

plt.scatter(np.array(df.index), df['FEAR'])
plt.plot(X, reg_pred, color='red')
plt.title('Fear')
plt.show()
```


Fear
30
25
20
15
10
5
0
−5
0
20
40
60
80
100