P2_M3_hist

June 15, 2021

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1 Wczytanie Danych

[2]:
```
df = pd.read_csv('data.csv')

data_lab = pd.read_csv('AllBooks_baseline_DTM_Labelled.csv')

cols = df.columns
texts = [''] * len(df)
for i in range(len(df)):
    t = texts[i]
    tmp_num = np.array(df.iloc[i])
    for j in range(len(tmp_num)):
        w = int(tmp_num[j])
        for k in range(w): t = t + ' ' + cols[j]
    texts[i] = str(t)
#    print(texts[i])

from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer

tfidf_vectorizer = TfidfVectorizer(max_df=0.9, min_df=2, use_idf=True,␣
↪stop_words='english', token_pattern=r"\b[^\d\W]+\b")

tfidf = tfidf_vectorizer.fit_transform(texts)
tfidf_feature_names = tfidf_vectorizer.get_feature_names()

df_tfidf = pd.DataFrame(tfidf.toarray(), columns=list(tfidf_feature_names))

#ramka danych ze statystykami tesktów
stats = pd.read_csv('stats_df.csv')
stats = stats.drop(['Unnamed: 0', 'index', 'text'], axis = 1)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(stats)
stat_scale = scaler.transform(stats)

stats_scale = pd.DataFrame(stat_scale, columns = stats.columns)

X = pd.merge(stats_scale.reset_index(), df_tfidf.reset_index(), on = 'index').
 ↪drop('index', axis = 1)
X.head()
```

[2]:

	len	words	avg_sen	reading_ease	grade	sentences	aaron	\
0	1.832013	1.549162	0.749075	0.162432	-0.298802	0.775681	0.0	
1	0.208099	0.189544	0.040772	0.928372	-0.808777	0.609403	0.0	
2	0.738420	0.632898	0.413880	0.768816	-0.741128	1.108236	0.0	
3	0.263277	0.197989	0.296945	0.614966	-0.683885	0.609403	0.0	
4	-0.785101	-0.806946	3.828118	0.500498	-0.668274	-0.554540	0.0	

	abandon	abasement	abate	…	yellow	yes	yesterday	yield	yieldeth	\
0	0.000000	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	
1	0.000000	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	
2	0.000000	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	
3	0.085756	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	
4	0.000000	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	

	yoga	yoke	young	youth	zeal
0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0

[5 rows x 3372 columns]

[3]:
```
Y = pd.read_csv('AllBooks_baseline_DTM_Labelled.csv')[['Unnamed: 0']]
Y['label'] = Y['Unnamed: 0'].apply(lambda x: x.split('_')[0])

def add_religion(label):
  if label == "Buddhism": return "Buddhism"
  elif label == "TaoTeChing": return "Taoism"
  elif (label == "Upanishad") | (label =="YogaSutra"): return "Hindusim"
  else: return "Old testament"


Y['rel'] = Y['label'].apply(lambda x : add_religion(x))
Y = Y.drop('Unnamed: 0', axis = 1)
Y.head()
```

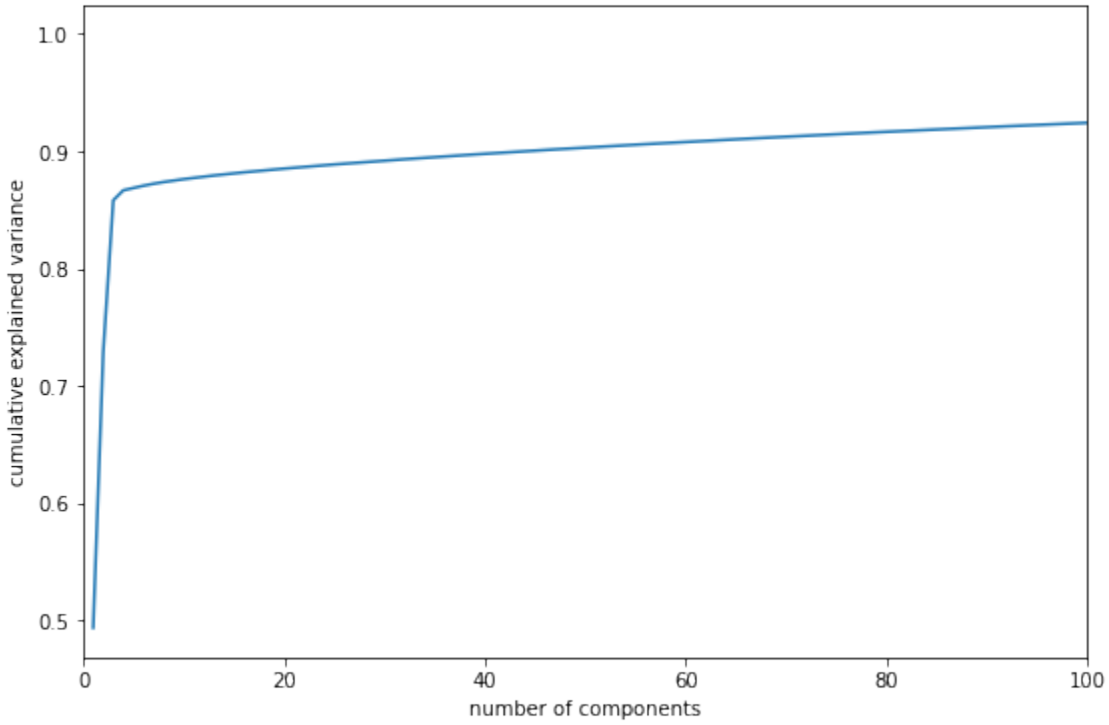
[3]:

	label	rel
0	Buddhism	Buddhism
1	Buddhism	Buddhism
2	Buddhism	Buddhism
3	Buddhism	Buddhism
4	Buddhism	Buddhism

[4]:
```
from sklearn.decomposition import PCA
```

2 PCA

[5]:
```
pca = PCA().fit(X)

plt.figure(figsize=(9,6))
plt.plot(range(1, len(pca.explained_variance_ratio_)+1), np.cumsum(pca.
 ↪explained_variance_ratio_))
plt.xlabel('number of components')
plt.xlim(0, 100)
plt.ylabel('cumulative explained variance');
```



Dla 3 komponentów mamy wyjaśnione 85% wariancji, 90% jest wyjaśnione przez 45 komponentów.

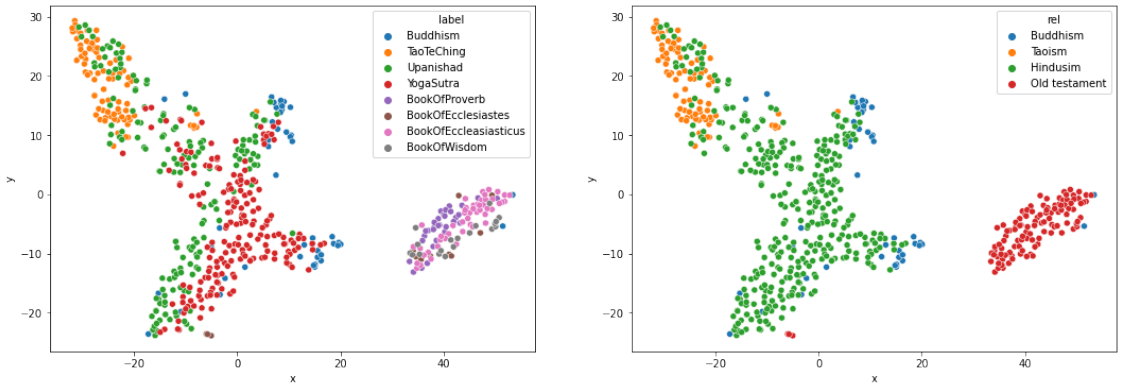
[6]:
```
X_pca45 = PCA(n_components=45).fit_transform(X)
```

[7]:
```
from sklearn.manifold import TSNE
```

[8]:
```
plt.figure(figsize=[10, 8])
tSNE = TSNE(random_state=0, verbose=1)
X_tsne = tSNE.fit_transform(X_pca45)
X_tsne = pd.DataFrame({'x': X_tsne[:, 0], 'y': X_tsne[:, 1], 'label':␣
↪Y['label'], 'rel' : Y['rel']})

f, (ax1, ax2) = plt.subplots(1, 2, figsize=[18, 6])
sns.scatterplot(data=X_tsne, x='x', y='y', hue='label', ax = ax1)
sns.scatterplot(data=X_tsne, x='x', y='y', hue='rel', ax = ax2)
plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors…
[t-SNE] Indexed 590 samples in 0.001s…
[t-SNE] Computed neighbors for 590 samples in 0.034s…
[t-SNE] Computed conditional probabilities for sample 590 / 590
[t-SNE] Mean sigma: 0.437386
[t-SNE] KL divergence after 250 iterations with early exaggeration: 61.640938
[t-SNE] KL divergence after 1000 iterations: 0.532836
```

```
<Figure size 720x576 with 0 Axes>
```



[9]:
```
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score, davies_bouldin_score, rand_score,␣
↪adjusted_mutual_info_score, mutual_info_score
```

[10]:
```
def KMeansClustering(data, reduction, actual_labels):
    results = pd.DataFrame(columns = ['clusters', 'silhouette_score',␣
↪'davies_bouldin_score',
                                      'rand_score',␣
↪'adjusted_mutual_info_score', 'mutual_info_score'])
```

```
    fig, axs = plt.subplots(1, 4, figsize = (18, 5))

    for i in range(2, 6):
        kmeans = KMeans(n_clusters=i, random_state=0)
        kmeans.fit(data)
        y_kmeans = kmeans.predict(data)

        i_results = pd.DataFrame({'clusters':[i],
                                  'silhouette_score':[silhouette_score(data,␣
↪y_kmeans)],
                                  'davies_bouldin_score':
↪[davies_bouldin_score(data, y_kmeans)],
                                  'rand_score':[rand_score(actual_labels,␣
↪y_kmeans)],
                                  'adjusted_mutual_info_score':
↪[adjusted_mutual_info_score(actual_labels, y_kmeans)],
                                  'mutual_info_score':
↪[mutual_info_score(actual_labels, y_kmeans)]})
        results = pd.concat([results, i_results])

        sns.scatterplot(data = reduction, x = 'x', y = 'y',
                        hue = y_kmeans, legend = False,
                        ax = axs[i-2], palette='viridis')
        ax1.set_title(f'{i} clusters')


    plt.show()
    return results
```

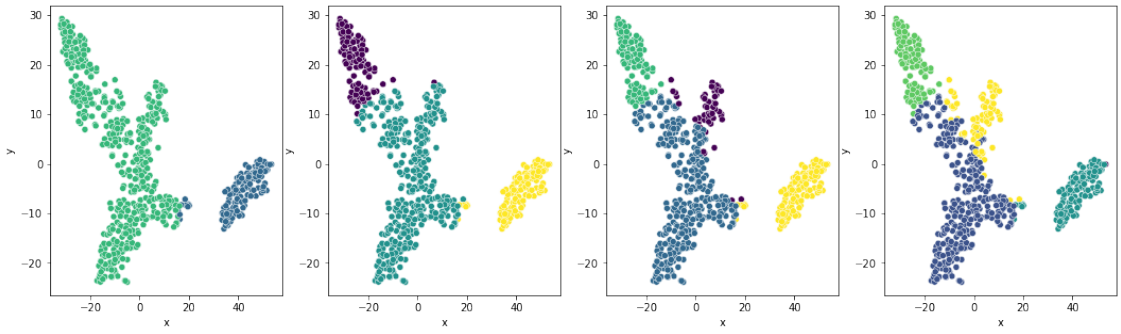
[11]:
```
def AggClustering(data, reduction, actual_labels):
    results = pd.DataFrame(columns = ['clusters', 'linkage',␣
↪'silhouette_score', 'davies_bouldin_score',
                                      'rand_score',␣
↪'adjusted_mutual_info_score', 'mutual_info_score'])

    fig, axs = plt.subplots(3, 4, figsize = (18, 15))
    linkage = ['ward', 'complete', 'single']

    for j in range(3):
        for i in range(2, 6):
            aggClus = AgglomerativeClustering(n_clusters = i, linkage =␣
↪linkage[j])
            y_aggClus = aggClus.fit_predict(data)

            i_results = pd.DataFrame({'clusters':[i],
```

```
                                    'linkage':[linkage[j]],
                                    'silhouette_score':[silhouette_score(data,␣
↪y_aggClus)],
                                    'davies_bouldin_score':
↪[davies_bouldin_score(data, y_aggClus)],
                                    'rand_score':[rand_score(actual_labels,␣
↪y_aggClus)],
                                    'adjusted_mutual_info_score':
↪[adjusted_mutual_info_score(actual_labels, y_aggClus)],
                                    'mutual_info_score':
↪[mutual_info_score(actual_labels, y_aggClus)]})
            results = pd.concat([results, i_results])

            sns.scatterplot(data = reduction, x = 'x', y = 'y',
                            hue = y_aggClus, legend = False,
                            ax = axs[j, i-2], palette='viridis')
            axs[j, i-2].set_title(f'{i} clusters, {linkage[j]} linkage')

    plt.show()
    return results
```

[12]:
```
def GMMClustering(data, reduction, actual_labels):
    results = pd.DataFrame(columns = ['clusters', 'covariance',␣
↪'silhouette_score', 'davies_bouldin_score',
                                      'rand_score',␣
↪'adjusted_mutual_info_score', 'mutual_info_score'])

    fig, axs = plt.subplots(3, 4, figsize = (18, 15))
    cov = ['full', 'tied', 'diag']

    for j in range(3):
        for i in range(2, 6):
            gmm = GaussianMixture(n_components=i, covariance_type=cov[j])
            y_gmm = gmm.fit_predict(data)

            i_results = pd.DataFrame({'clusters':[i],
                                    'covariance':[cov[j]],
                                    'silhouette_score':[silhouette_score(data,␣
↪y_gmm)],
                                    'davies_bouldin_score':
↪[davies_bouldin_score(data, y_gmm)],
                                    'rand_score':[rand_score(actual_labels,␣
↪y_gmm)],
                                    'adjusted_mutual_info_score':
↪[adjusted_mutual_info_score(actual_labels, y_gmm)],
```

```
                                    'mutual_info_score':
↪[mutual_info_score(actual_labels, y_gmm)]})
            results = pd.concat([results, i_results])

            sns.scatterplot(data = reduction, x = 'x', y = 'y',
                            hue = y_gmm, legend = False,
                            ax = axs[j, i-2], palette='viridis')
            axs[j, i-2].set_title(f'{i} clusters, {cov[j]} covarince')

    plt.show()
    return results
```

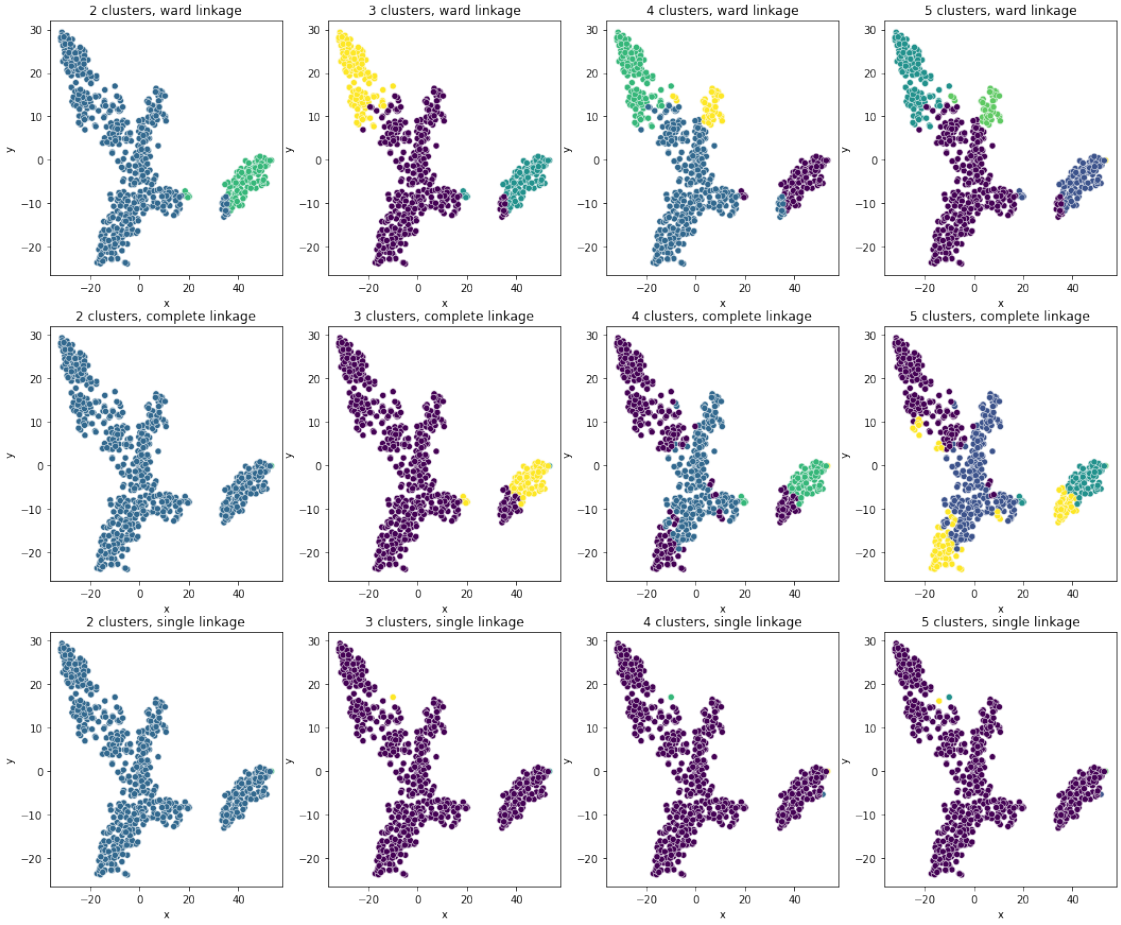
[13]:
```
a = KMeansClustering(X_pca45, X_tsne, Y['rel'])
```



[14]:
```
a.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[14]: <pandas.io.formats.style.Styler at 0x1497ee50>

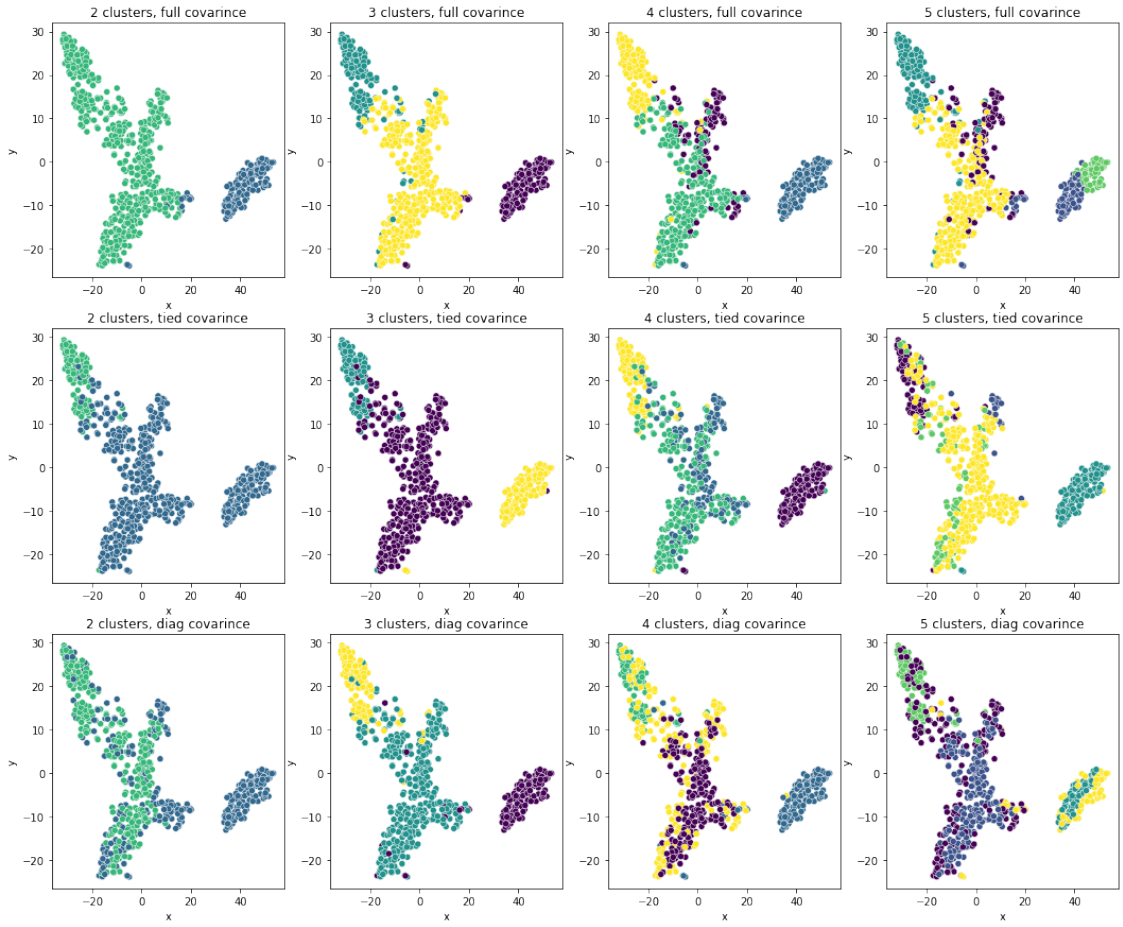
[15]:
```
b = AggClustering(X_pca45, X_tsne, Y['rel'])
```



[16]:
```
b.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[16]: <pandas.io.formats.style.Styler at 0x149847d8>

[17]:
```
c = GMMClustering(X_pca45, X_tsne, Y['rel'])
```



[18]:
```
c.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[18]:
```
<pandas.io.formats.style.Styler at 0x1498e6d0>
```

3 TruncatedSVD

[19]:
```
from sklearn.decomposition import TruncatedSVD
from scipy.sparse import csr_matrix
```

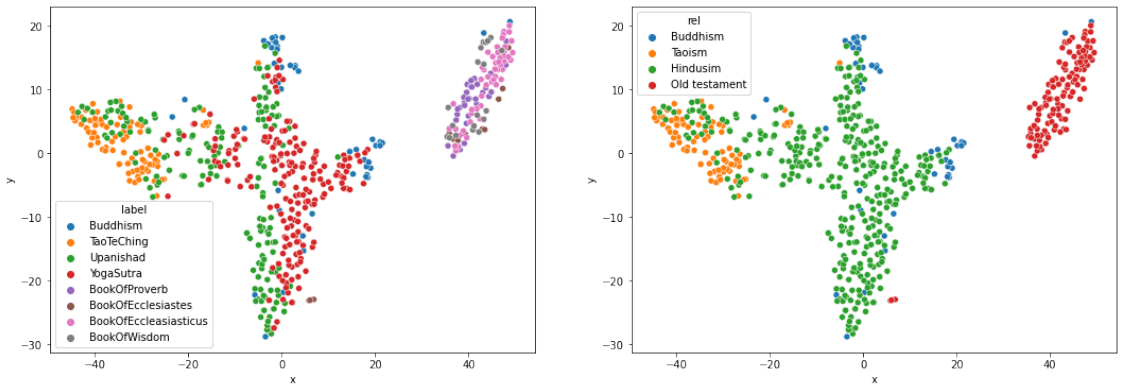
[20]:
```
X_csr = csr_matrix(X)

tsvd = TruncatedSVD(n_components=50)
tsvd.fit(X_csr)
X_tsvd_t = tsvd.transform(X_csr)
```

[21]:
```
plt.figure(figsize=[10, 8])
tSNE = TSNE(random_state=0, verbose=1)
```

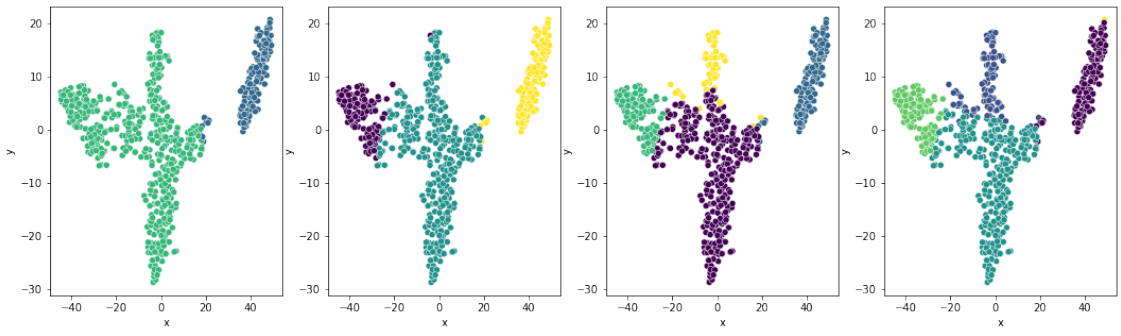
```
X_tsne = tSNE.fit_transform(X_tsvd_t)
X_tsne = pd.DataFrame({'x': X_tsne[:, 0], 'y': X_tsne[:, 1], 'label':␣
 ↪Y['label'], 'rel' : Y['rel']})

f, (ax1, ax2) = plt.subplots(1, 2, figsize=[18, 6])
sns.scatterplot(data=X_tsne, x='x', y='y', hue='label', ax = ax1)
sns.scatterplot(data=X_tsne, x='x', y='y', hue='rel', ax = ax2)
plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors…
[t-SNE] Indexed 590 samples in 0.000s…
[t-SNE] Computed neighbors for 590 samples in 0.021s…
[t-SNE] Computed conditional probabilities for sample 590 / 590
[t-SNE] Mean sigma: 0.440224
[t-SNE] KL divergence after 250 iterations with early exaggeration: 61.060123
[t-SNE] KL divergence after 1000 iterations: 0.535423
```

```
<Figure size 720x576 with 0 Axes>
```



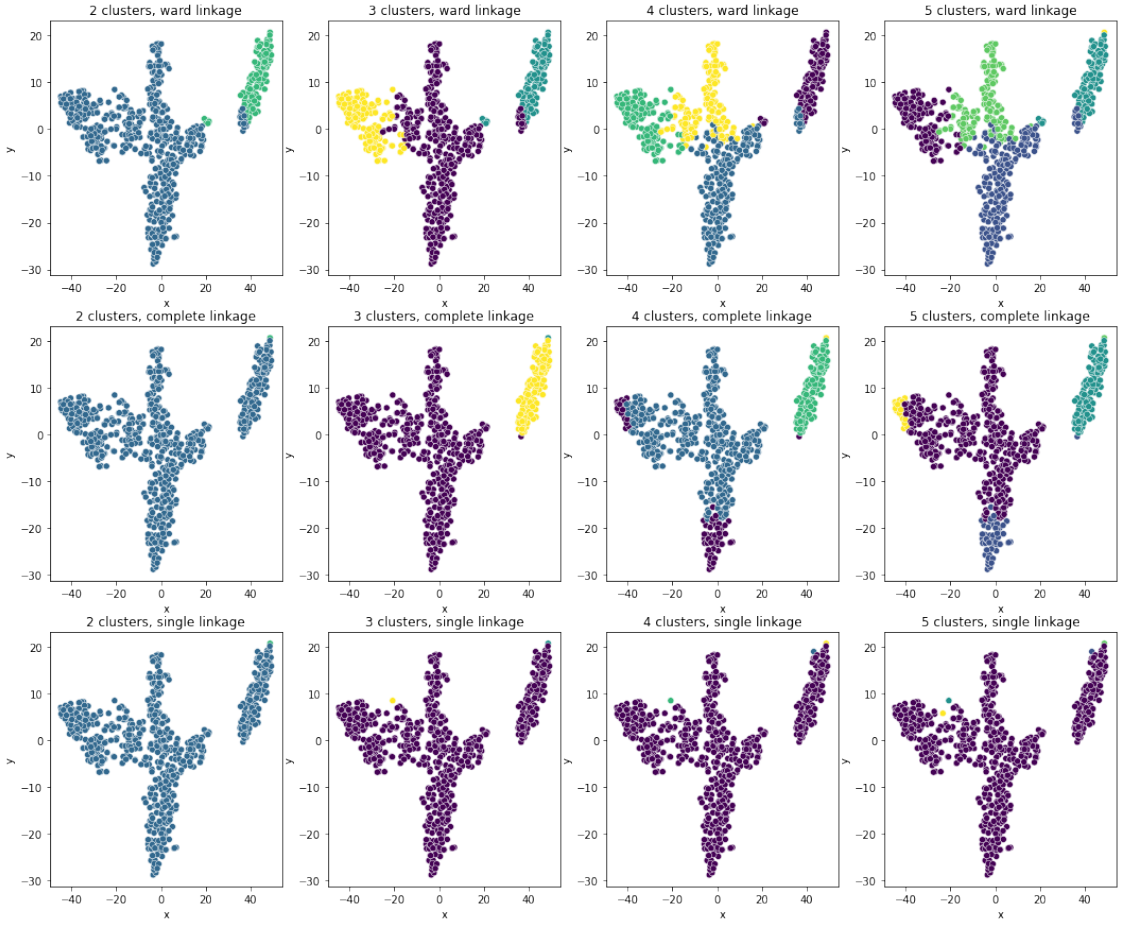
[22]:
```
a = KMeansClustering(X_tsvd_t, X_tsne, Y['rel'])
```



[23]:
```
a.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[23]: <pandas.io.formats.style.Styler at 0x20094550>

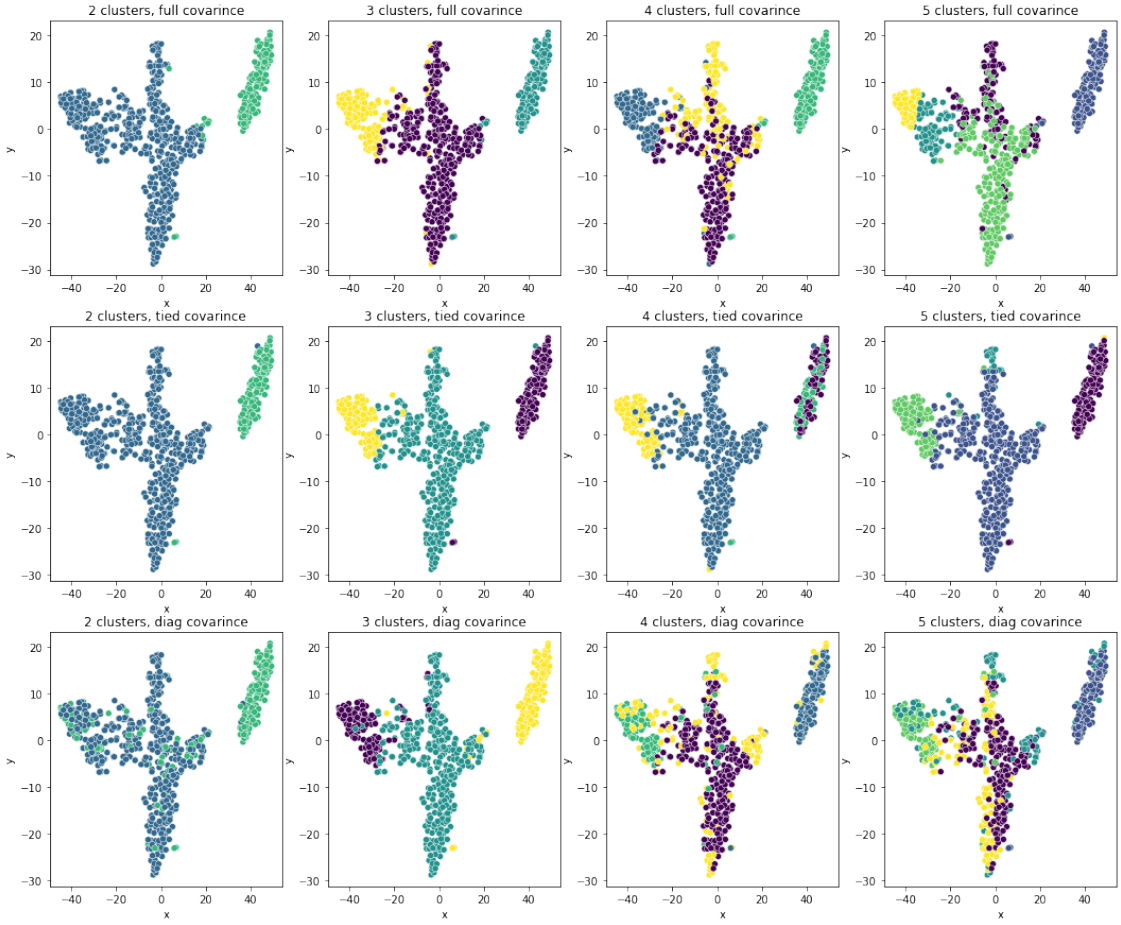
[24]:
```
b = AggClustering(X_tsvd_t, X_tsne, Y['rel'])
```



[25]:
```
b.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[25]: <pandas.io.formats.style.Styler at 0x1e1ff8e0>

[26]:
```
c = GMMClustering(X_tsvd_t, X_tsne, Y['rel'])
```



[27]:
```
c.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[27]: <pandas.io.formats.style.Styler at 0x1de146b8>

4 NMF

[28]:
```
from sklearn.decomposition import NMF
```

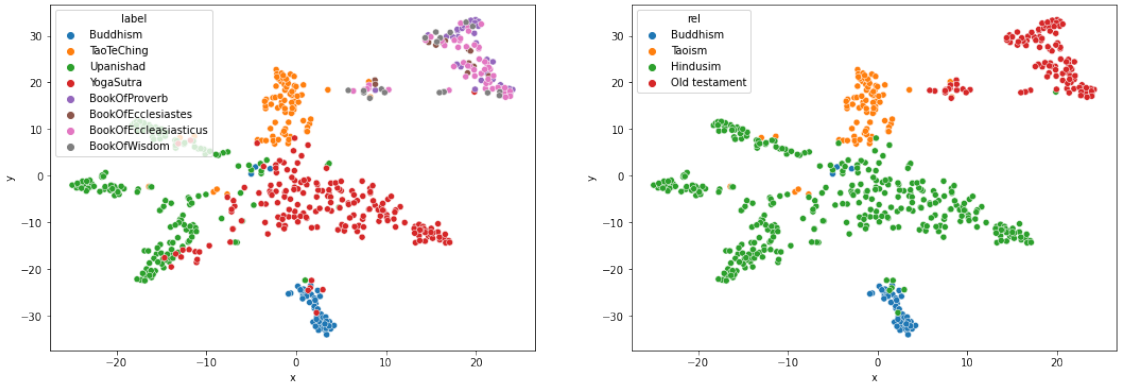
[29]:
```
X_tfidf_csr = csr_matrix(df_tfidf)
nmf = NMF(n_components=8)
X_nmf_t = nmf.fit_transform(X_tfidf_csr)
```

```
c:\users\user\appdata\local\programs\python\python38-32\lib\site-
packages\sklearn\decomposition\_nmf.py:312: FutureWarning: The 'init' value,
when 'init=None' and n_components is less than n_samples and n_features, will be
changed from 'nndsvd' to 'nndsvda' in 1.1 (renaming of 0.26).
  warnings.warn(("The 'init' value, when 'init=None' and "
```

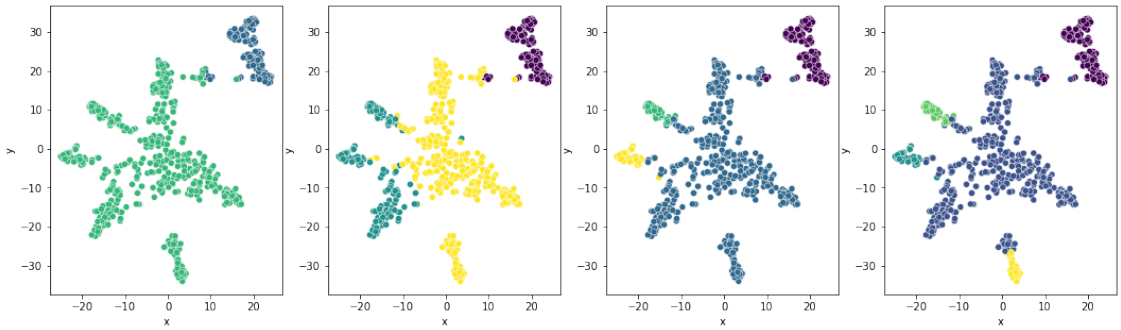
[30]:
```
plt.figure(figsize=[10, 8])
tSNE = TSNE(random_state=0, verbose=1)
X_tsne = tSNE.fit_transform(X_nmf_t)
X_tsne = pd.DataFrame({'x': X_tsne[:, 0], 'y': X_tsne[:, 1], 'label':␣
↪Y['label'], 'rel' : Y['rel']})

f, (ax1, ax2) = plt.subplots(1, 2, figsize=[18, 6])
sns.scatterplot(data=X_tsne, x='x', y='y', hue='label', ax = ax1)
sns.scatterplot(data=X_tsne, x='x', y='y', hue='rel', ax = ax2)
plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors…
[t-SNE] Indexed 590 samples in 0.003s…
[t-SNE] Computed neighbors for 590 samples in 0.026s…
[t-SNE] Computed conditional probabilities for sample 590 / 590
[t-SNE] Mean sigma: 0.038622
[t-SNE] KL divergence after 250 iterations with early exaggeration: 58.535061
[t-SNE] KL divergence after 1000 iterations: 0.425435
```

```
<Figure size 720x576 with 0 Axes>
```



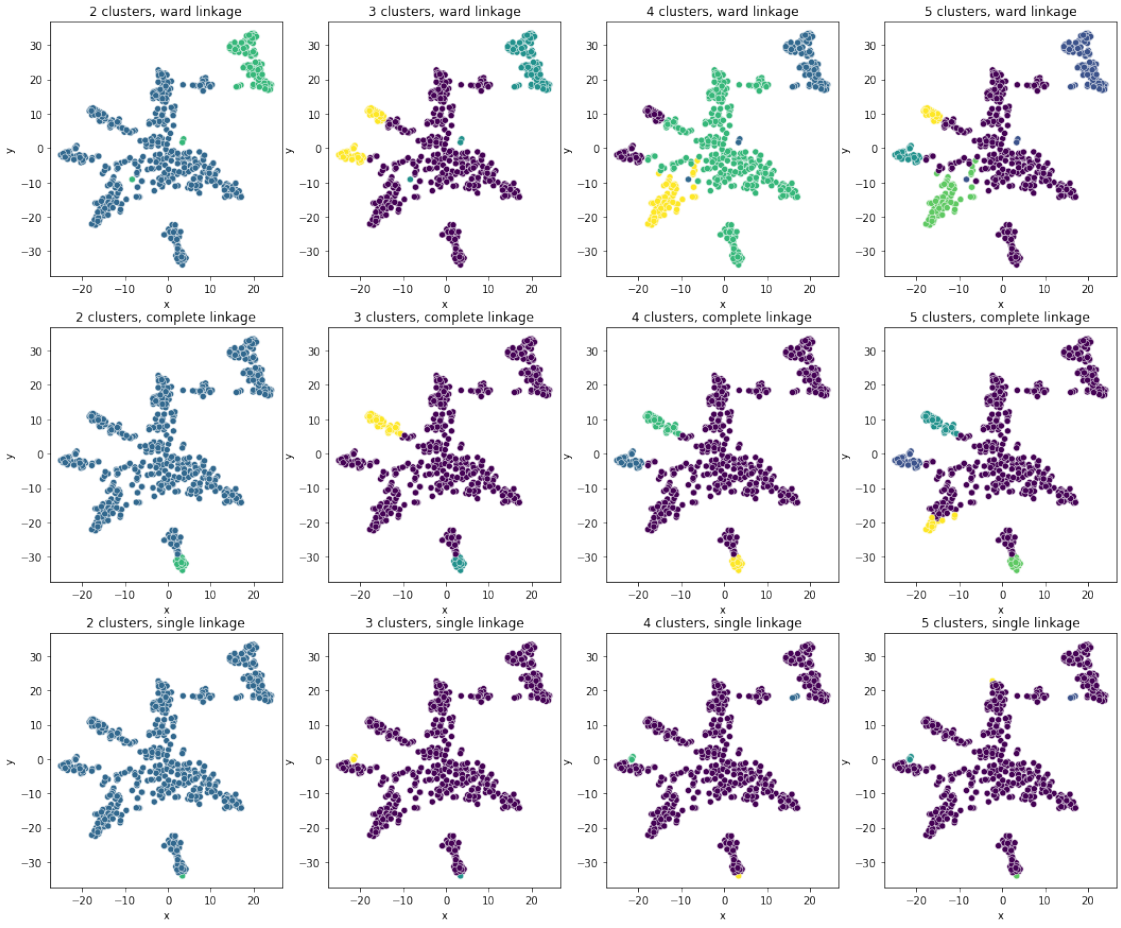
[31]:
```
a = KMeansClustering(X_nmf_t, X_tsne, Y['rel'])
```



[32]:
```
a.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[32]: <pandas.io.formats.style.Styler at 0x1e081760>

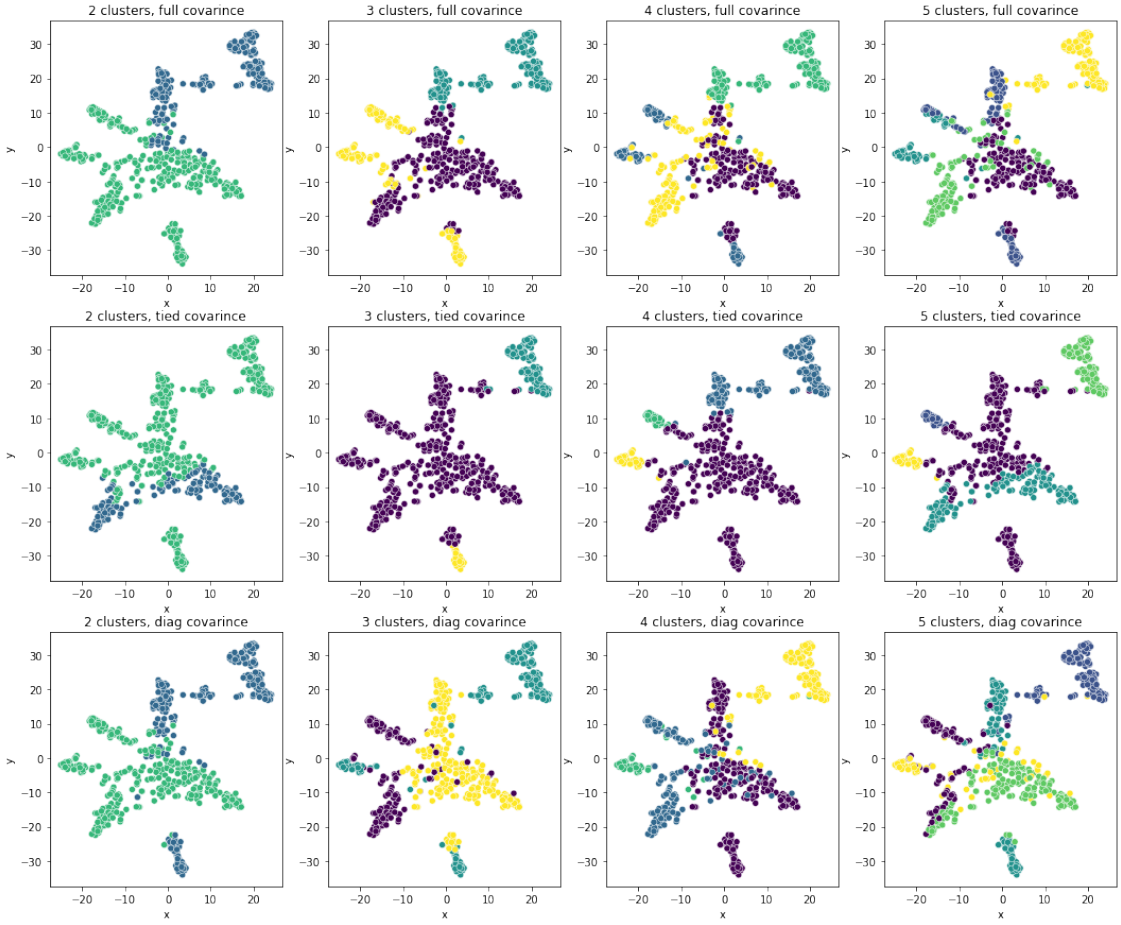
[33]:
```
b = AggClustering(X_nmf_t, X_tsne, Y['rel'])
```



[34]:
```
b.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[34]: <pandas.io.formats.style.Styler at 0x1df03520>

[35]:
```
c = GMMClustering(X_nmf_t, X_tsne, Y['rel'])
```



[36]:
```
c.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[36]: <pandas.io.formats.style.Styler at 0x15cad2e0>

5 Sparse PCA

[37]:
```
from sklearn.decomposition import SparsePCA
```

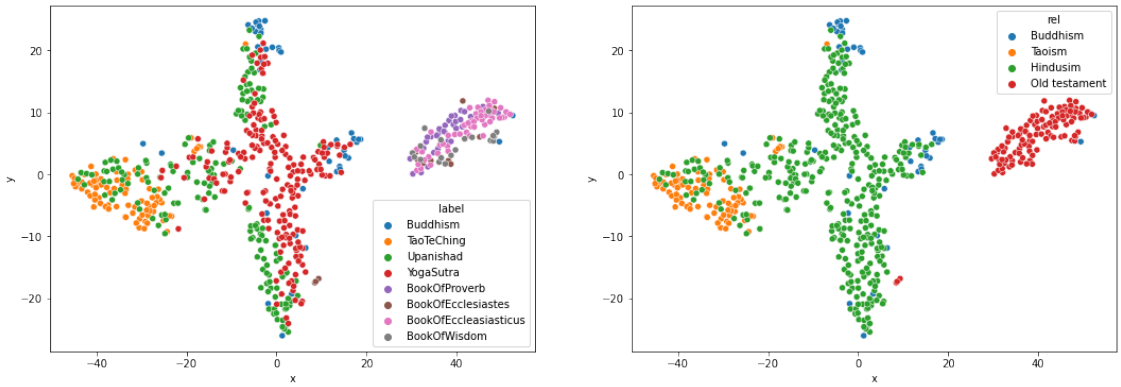
[38]:
```
spca = SparsePCA(n_components=45)
X_spca_t = spca.fit_transform(X)
```

[39]:
```
plt.figure(figsize=[10, 8])
tSNE = TSNE(random_state=0, verbose=1)
X_tsne = tSNE.fit_transform(X_spca_t)
X_tsne = pd.DataFrame({'x': X_tsne[:, 0], 'y': X_tsne[:, 1], 'label':␣
 ↪Y['label'], 'rel' : Y['rel']})
```

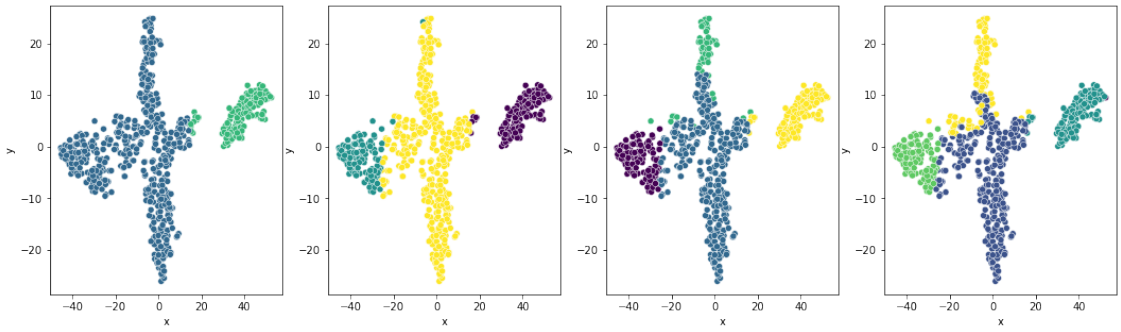
```
f, (ax1, ax2) = plt.subplots(1, 2, figsize=[18, 6])
sns.scatterplot(data=X_tsne, x='x', y='y', hue='label', ax = ax1)
sns.scatterplot(data=X_tsne, x='x', y='y', hue='rel', ax = ax2)
plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors…
[t-SNE] Indexed 590 samples in 0.003s…
[t-SNE] Computed neighbors for 590 samples in 0.038s…
[t-SNE] Computed conditional probabilities for sample 590 / 590
[t-SNE] Mean sigma: 0.382511
[t-SNE] KL divergence after 250 iterations with early exaggeration: 58.356899
[t-SNE] KL divergence after 1000 iterations: 0.468120
```

```
<Figure size 720x576 with 0 Axes>
```



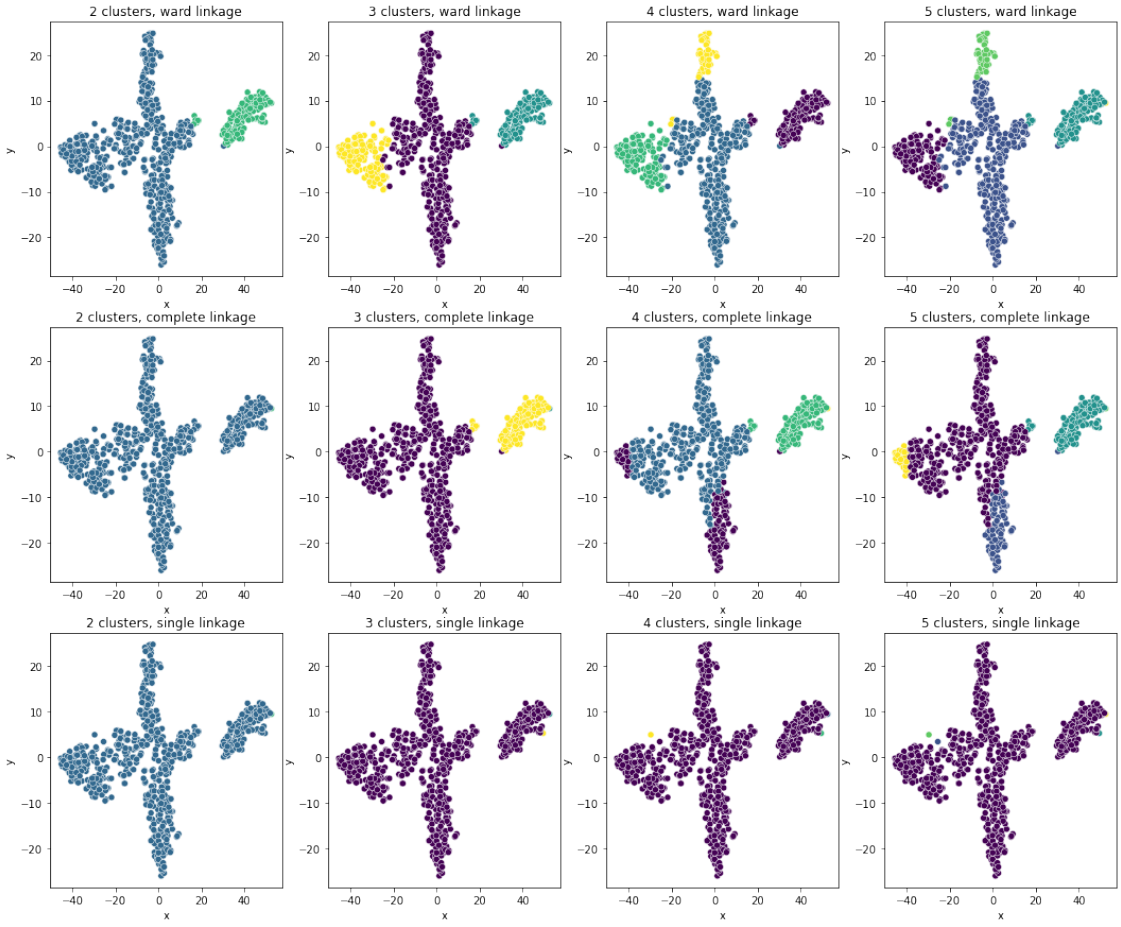
[40]:
```
a = KMeansClustering(X_spca_t, X_tsne, Y['rel'])
```



[41]:
```
a.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[41]: <pandas.io.formats.style.Styler at 0x1e139fe8>

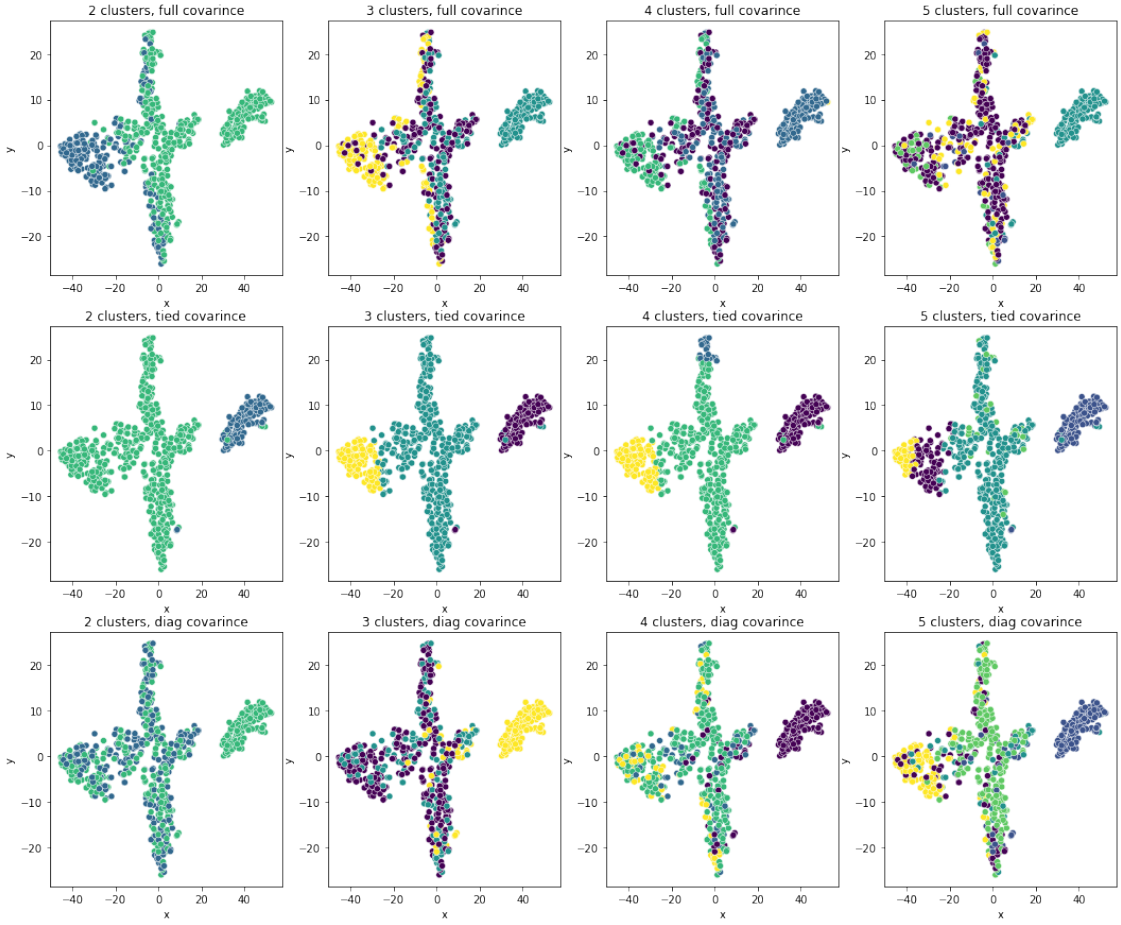
[42]:
```
b = AggClustering(X_spca_t, X_tsne, Y['rel'])
```



[43]:
```
b.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[43]:
```
<pandas.io.formats.style.Styler at 0x13819a78>
```

[44]:
```
c = GMMClustering(X_spca_t, X_tsne, Y['rel'])
```



[45]:
```
c.reset_index(drop=True).style.background_gradient(cmap='Blues')
```

[45]: <pandas.io.formats.style.Styler at 0x14ac5670>

6 Najlepsze klastrowania

[46]:
```
from sklearn.metrics import calinski_harabasz_score
```

[47]:
```
def metrics(data, actual_labels, predict):
    return pd.DataFrame({
        'silhouette_score':[silhouette_score(data, predict)],
        'davies_bouldin_score':[davies_bouldin_score(data, predict)],
        'rand_score':[rand_score(actual_labels, predict)],
        'adjusted_mutual_info_score':[adjusted_mutual_info_score(actual_labels,␣
↪predict)],
        'mutual_info_score':[mutual_info_score(actual_labels, predict)],
        'calinski_harabasz_score' :[calinski_harabasz_score(data, predict)]
```

```
    }).transpose()
```

[48]:
```
from wordcloud import WordCloud, STOPWORDS
stopwords = set(STOPWORDS)

def show_wordcloud(data):
    wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=100,
        max_font_size=30,
        scale=3,
        random_state=1)

    wordcloud=wordcloud.generate(str(data))

    fig = plt.figure(1, figsize=(12, 12))
    plt.axis('off')

    plt.imshow(wordcloud)
    plt.show()
```

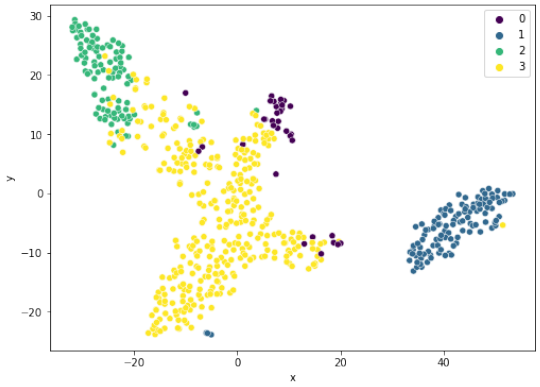
6.0.1 PCA45, 3 klastry, GMM, tied

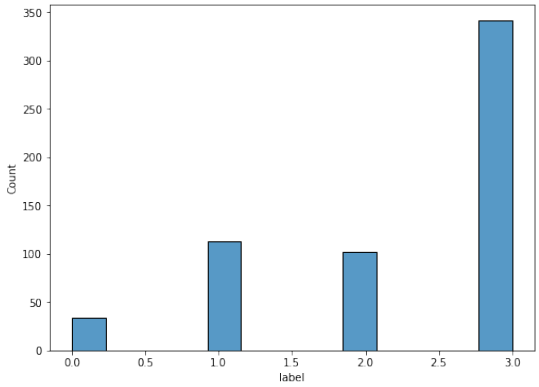
6.0.2 PCA45, 4 klastry, GMM, tied

[49]:
```
gmm = GaussianMixture(n_components = 4, covariance_type='tied')
X_pca45_gmm_4 = gmm.fit_predict(X_pca45)

tSNE = TSNE(random_state=0, verbose=1)
X_tsne = tSNE.fit_transform(X_pca45)
X_tmp = pd.DataFrame({'x': X_tsne[:, 0], 'y': X_tsne[:, 1], 'label':␣
↪X_pca45_gmm_4})
X_tsne = pd.DataFrame({'x': X_tsne[:, 0], 'y': X_tsne[:, 1], 'label':␣
↪Y['label'], 'rel' : Y['rel']})

f, (ax1, ax2) = plt.subplots(1, 2, figsize=[18, 6])
sns.scatterplot(data = X_tsne, x = 'x', y = 'y', hue = X_pca45_gmm_4, legend =␣
↪'auto', palette='viridis', ax = ax1)
sns.histplot(X_tmp['label'], ax = ax2, palette='viridis')
plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors…
[t-SNE] Indexed 590 samples in 0.001s…
[t-SNE] Computed neighbors for 590 samples in 0.026s…
[t-SNE] Computed conditional probabilities for sample 590 / 590
[t-SNE] Mean sigma: 0.437386
```

```
[t-SNE] KL divergence after 250 iterations with early exaggeration: 61.640938
[t-SNE] KL divergence after 1000 iterations: 0.532836
```





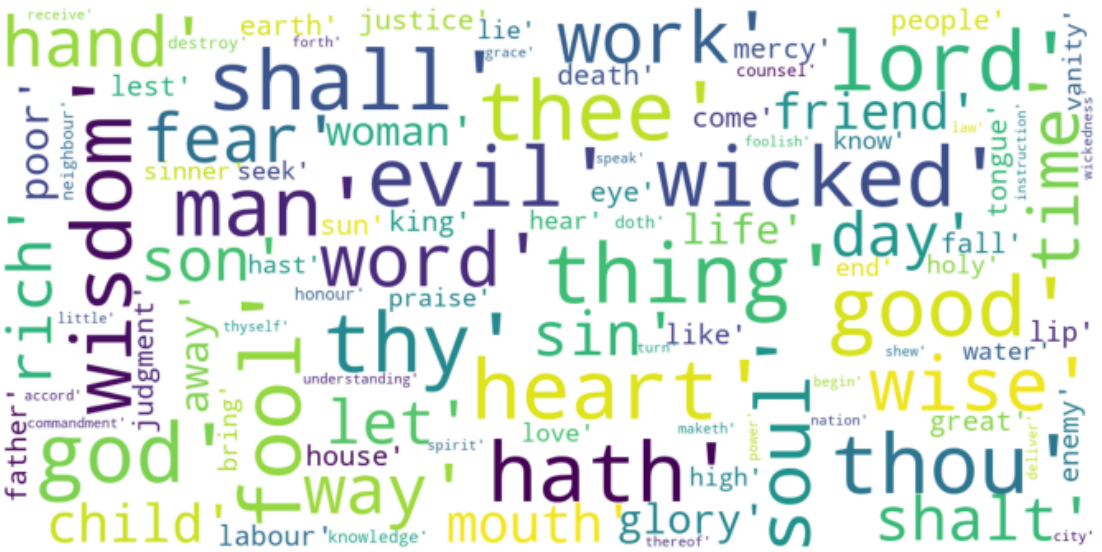
[50]:
```
metrics(X_pca45, Y['rel'], X_pca45_gmm_4)
```

[50]:

	0
silhouette_score	0.402307
davies_bouldin_score	1.021393
rand_score	0.891675
adjusted_mutual_info_score	0.759235
mutual_info_score	0.835996
calinski_harabasz_score	360.647869

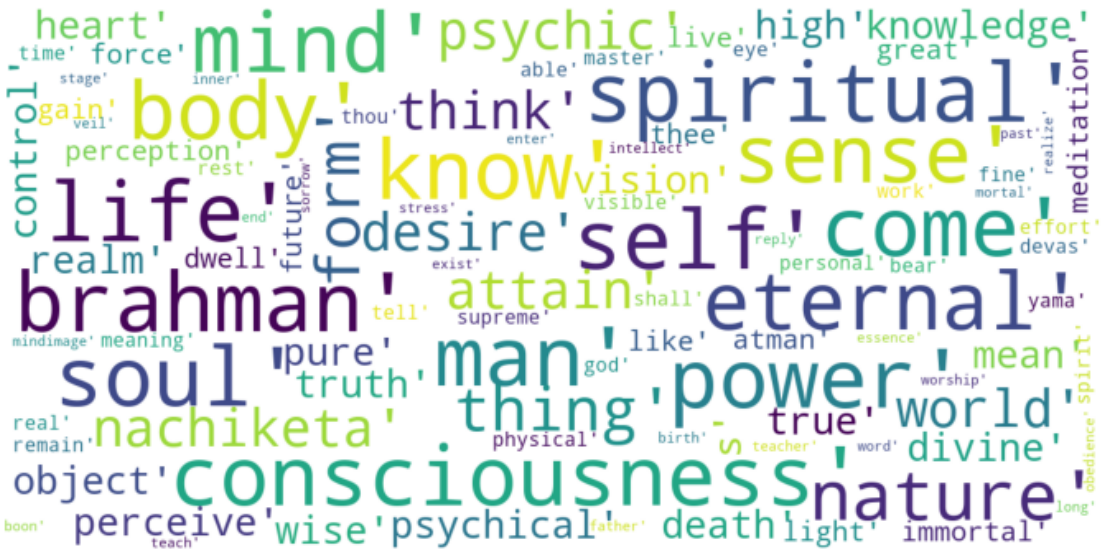
[51]:
```
show_wordcloud(df_tfidf.loc[X_pca45_gmm_4 == 0].sum().
↪sort_values(ascending=False).to_dict())
```

[52]:
```
show_wordcloud(df_tfidf.loc[X_pca45_gmm_4 == 1].sum().
↪sort_values(ascending=False).to_dict())
```



[53]:
```
show_wordcloud(df_tfidf.loc[X_pca45_gmm_4 == 2].sum().
↪sort_values(ascending=False).to_dict())
```

[54]:
```
show_wordcloud(df_tfidf.loc[X_pca45_gmm_4 == 3].sum().
↪sort_values(ascending=False).to_dict())
```



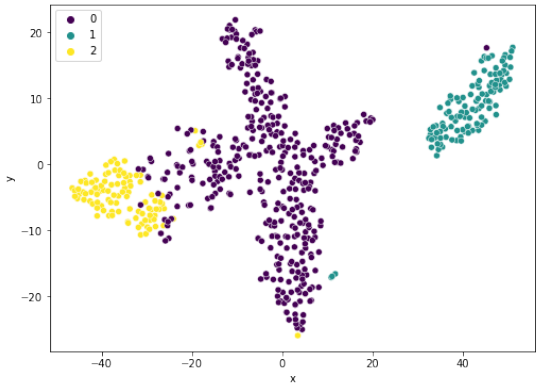
6.1 TruncatedSVD, 3 klastry, GMM, tied

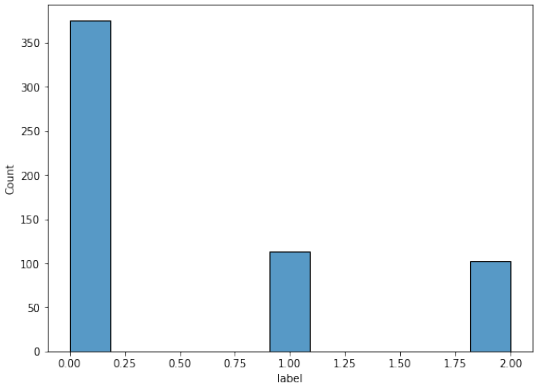
[55]:
```
X_csr = csr_matrix(X)

tsvd = TruncatedSVD(n_components=50)
tsvd.fit(X_csr)
X_tsvd_t = tsvd.transform(X_csr)
```

[56]:
```
gmm = GaussianMixture(n_components = 3, covariance_type='tied')
X_tsvd_gmm_3 = gmm.fit_predict(X_tsvd_t)

tSNE = TSNE(random_state=0, verbose=1)
X_tsne = tSNE.fit_transform(X_tsvd_t)
X_tmp = pd.DataFrame({'x': X_tsne[:, 0], 'y': X_tsne[:, 1], 'label':␣
↪X_tsvd_gmm_3})
X_tsne = pd.DataFrame({'x': X_tsne[:, 0], 'y': X_tsne[:, 1], 'label':␣
↪Y['label'], 'rel' : Y['rel']})

f, (ax1, ax2) = plt.subplots(1, 2, figsize=[18, 6])
sns.scatterplot(data = X_tsne, x = 'x', y = 'y', hue = X_tsvd_gmm_3, legend =␣
↪'auto', palette='viridis', ax = ax1)
sns.histplot(X_tmp['label'], ax = ax2, palette='viridis')
plt.show()
```

```
[t-SNE] Computing 91 nearest neighbors…
[t-SNE] Indexed 590 samples in 0.000s…
[t-SNE] Computed neighbors for 590 samples in 0.026s…
[t-SNE] Computed conditional probabilities for sample 590 / 590
```

```
[t-SNE] Mean sigma: 0.438929
[t-SNE] KL divergence after 250 iterations with early exaggeration: 62.003750
[t-SNE] KL divergence after 1000 iterations: 0.535137
```



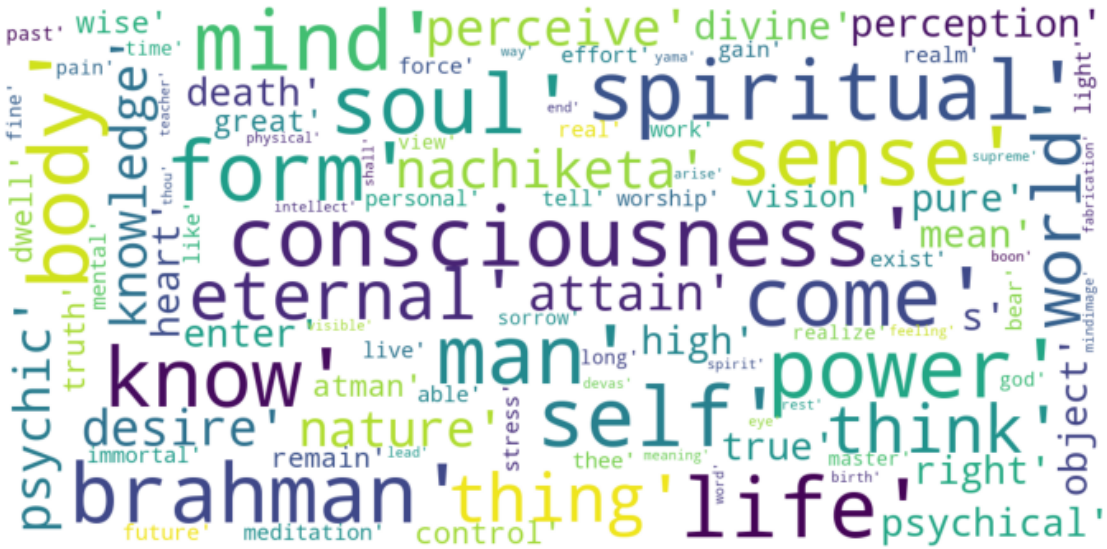


[57]:
```
metrics(X_tsvd_t, Y['rel'], X_tsvd_gmm_3)
```

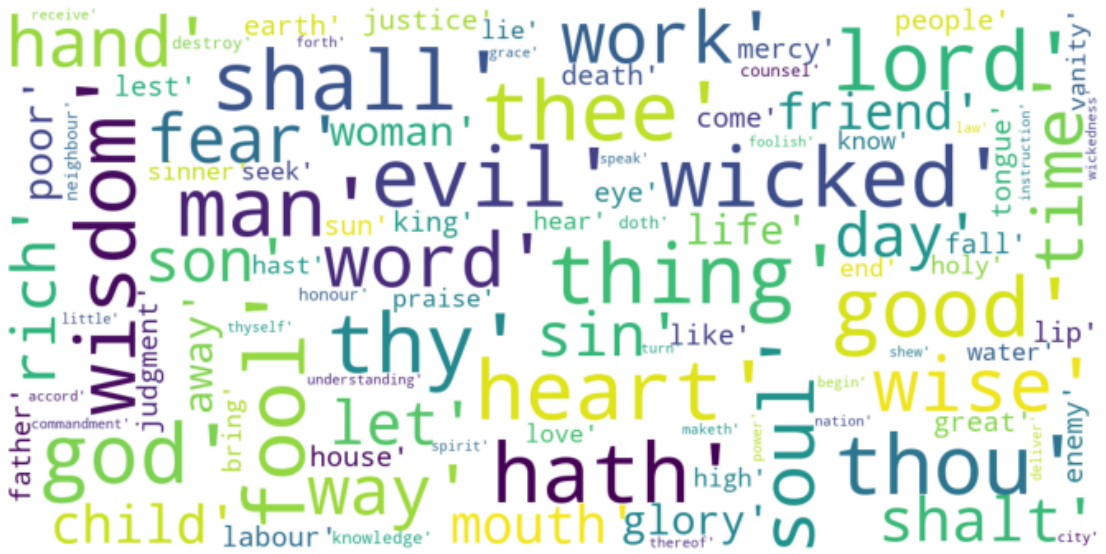
[57]:

	0
silhouette_score	0.443955
davies_bouldin_score	0.801161
rand_score	0.849340
adjusted_mutual_info_score	0.713454
mutual_info_score	0.716342
calinski_harabasz_score	455.342691

[58]:
```
show_wordcloud(df_tfidf.loc[X_tsvd_gmm_3 == 0].sum().
↪sort_values(ascending=False).to_dict())
```



[59]:
```
show_wordcloud(df_tfidf.loc[X_tsvd_gmm_3 == 1].sum().
↪sort_values(ascending=False).to_dict())
```



[60]:
```
show_wordcloud(df_tfidf.loc[X_tsvd_gmm_3 == 2].sum().
↪sort_values(ascending=False).to_dict())
```