Import required packages

24-nov

```
- add dbscan to basic clustering ...
```

In [ ]:

TOC

In [63]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#%matplotlib inline
import warnings
warnings.simplefilter('ignore',DeprecationWarning)
import seaborn as sns
import time
import copy

from pylab import rcParams
#import hdbscan

from sklearn.model_selection import ShuffleSplit
from sklearn.preprocessing import StandardScaler

#from sklearn.datasets import make_blobs

from sklearn.ensemble import RandomForestClassifier
from sklearn.calibration import CalibratedClassifierCV
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold, cross_val_score

from sklearn import metrics
from sklearn import metrics as mt
from sklearn.metrics import log_loss
from sklearn.metrics import accuracy_score as acc
from sklearn.metrics import confusion_matrix as conf
from sklearn.metrics import f1_score, precision_score, recall_score, classi
fication_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_recall_fscore_support as score

from sklearn.cluster import KMeans

from tabulate import tabulate

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

from __future__ import print_function
```

Read in cleaned dataset from .csv file

In [64]:
```
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... read in cleaned data set
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

data_dir = '../data/'
data_file = 'mashable_clean_dataset_for_lab_03.csv'

file_2_read = data_dir + data_file
df = pd.read_csv(file_2_read)

df_cluster = copy.deepcopy(df)

df_cluster.head()

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... add 'popular' binary based 'shares' values
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

df_cluster['shares'] = np.exp(df_cluster['ln_shares'])
df_cluster['popular'] = np.where(df_cluster['shares'] > 1400, 1, 0)

df_cluster.head()

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... read in t-SNE vectors
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

data_dir = '../data/'
data_file = 't_sne_mapping_perplex_0100.csv'

file_2_read = data_dir + data_file
df_tsne = pd.read_csv(file_2_read)

df_tsne.head()
```

Out[64]:

	n_tokens_title	num_keywords	data_channel_is_lifestyle	data_channel_is_entertainment	data_c
0	0.757447	-1.164821	0	1	0
1	-0.661657	-1.688626	0	0	0
2	-0.661657	-0.641015	0	0	0
3	-0.661657	-0.117210	0	1	0
4	1.230482	-0.117210	0	0	0

5 rows × 33 columns

Out[64]:

	n_tokens_title	num_keywords	data_channel_is_lifestyle	data_channel_is_entertainment	data_c
0	0.757447	-1.164821	0	1	0
1	-0.661657	-1.688626	0	0	0
2	-0.661657	-0.641015	0	0	0
3	-0.661657	-0.117210	0	1	0
4	1.230482	-0.117210	0	0	0

5 rows × 35 columns

Out[64]:

	x-tsne	y-tsne	sample_index
0	13.380795	10.659998	6972
1	-9.200704	3.382056	11092
2	-29.567352	-1.899089	10229
3	-3.023949	21.598028	1319
4	-14.161682	-6.246559	29199

In [65]:
```
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... join t-sne vectors with base data, since we sampled to create
# ...     the t-sne mapping
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

df_join = df_tsne.join(df_cluster, on = 'sample_index')

df_join.head()
```

Out[65]:

	x-tsne	y-tsne	sample_index	n_tokens_title	num_keywords	data_channel_is_lifestyle	
0	13.380795	10.659998	6972	-0.661657	-1.688626	0	
1	-9.200704	3.382056	11092	-2.080761	-0.641015	0	
2	-29.567352	-1.899089	10229	-0.188622	-0.641015	0	
3	-3.023949	21.598028	1319	0.757447	-1.688626	0	
4	-14.161682	-6.246559	29199	0.757447	0.406595	0	

5 rows × 38 columns

In [66]:
```
col_names = df_join.columns.values.tolist()
col_names
df_join.describe().T
```

```
Out[66]: ['x-tsne',
          'y-tsne',
          'sample_index',
          'n_tokens_title',
          'num_keywords',
          'data_channel_is_lifestyle',
          'data_channel_is_entertainment',
          'data_channel_is_socmed',
          'kw_avg_max',
          'is_weekend',
          'global_subjectivity',
          'global_rate_positive_words',
          'rate_positive_words',
          'max_positive_polarity',
          'min_negative_polarity',
          'max_negative_polarity',
          'title_sentiment_polarity',
          'abs_title_subjectivity',
          'ln_n_tokens_content',
          'ln_num_hrefs',
          'ln_num_imgs',
          'ln_num_videos',
          'ln_kw_min_min',
          'ln_kw_avg_min',
          'ln_kw_min_max',
          'ln_kw_avg_avg',
          'ln_self_reference_avg_sharess',
          'ln_LDA_00',
          'ln_LDA_01',
          'ln_LDA_02',
          'ln_LDA_03',
          'ln_LDA_04',
          'ln_global_rate_negative_words',
          'ln_min_positive_polarity',
          'ln_abs_title_sentiment_polarity',
          'ln_shares',
          'shares',
          'popular']
```

Out[66]:

	count	mean	std	min	25%	
x-tsne	13875.0	0.410117	14.157297	-32.599712	-10.773357	-0.
y-tsne	13875.0	0.040092	9.436279	-22.147314	-6.963444	-0.
sample_index	13875.0	20058.394739	11400.211047	0.000000	10224.000000	20
n_tokens_title	13875.0	-0.002783	0.997158	-3.972899	-0.661657	-0.
num_keywords	13875.0	-0.011052	0.997482	-3.260042	-0.641015	-0.
data_channel_is_lifestyle	13875.0	0.050739	0.219472	0.000000	0.000000	0.0
data_channel_is_entertainment	13875.0	0.178739	0.383147	0.000000	0.000000	0.0
data_channel_is_socmed	13875.0	0.057658	0.233103	0.000000	0.000000	0.0
kw_avg_max	13875.0	0.021806	0.998054	-1.911764	-0.621457	-0.
is_weekend	13875.0	0.130450	0.336810	0.000000	0.000000	0.0
global_subjectivity	13875.0	0.002958	0.988828	-3.799778	-0.402604	0.0
global_rate_positive_words	13875.0	0.011095	0.999152	-2.273573	-0.640051	-0.
rate_positive_words	13875.0	0.013677	0.989198	-3.586415	-0.431906	0.1
max_positive_polarity	13875.0	0.011449	0.992793	-3.053998	-0.632520	0.1
min_negative_polarity	13875.0	0.003018	0.997388	-1.646847	-0.613383	0.0
max_negative_polarity	13875.0	-0.000837	1.009672	-9.358111	-0.183490	0.0
title_sentiment_polarity	13875.0	0.004485	1.003665	-4.036308	-0.269076	-0.
abs_title_subjectivity	13875.0	0.003747	0.998231	-1.810719	-0.927896	0.8
ln_n_tokens_content	13875.0	0.009432	0.985424	-4.691612	-0.299932	0.1
ln_num_hrefs	13875.0	0.012760	0.999294	-2.664283	-0.675935	0.0
ln_num_imgs	13875.0	0.007444	1.006700	-1.146531	-0.434693	-0.
ln_num_videos	13875.0	0.012156	1.013900	-0.588439	-0.588439	-0.
ln_kw_min_min	13875.0	-0.020741	0.984775	-0.677671	-0.677671	-0.
ln_kw_avg_min	13875.0	-0.001939	1.003748	-4.682076	-0.293441	0.1
ln_kw_min_max	13875.0	0.007746	0.998410	-1.115960	-1.115960	0.5
ln_kw_avg_avg	13875.0	0.006153	1.001031	-16.296139	-0.403607	-0.
ln_self_reference_avg_sharess	13875.0	0.016240	0.992200	-2.032745	0.073490	0.3
ln_LDA_00	13875.0	0.002041	1.002881	-0.671547	-0.636885	-0.
ln_LDA_01	13875.0	0.000617	0.999165	-0.600268	-0.559735	-0.
ln_LDA_02	13875.0	-0.004631	0.995019	-0.745912	-0.696942	-0.
ln_LDA_03	13875.0	0.003400	0.999146	-0.734874	-0.687886	-0.
ln_LDA_04	13875.0	0.000682	0.996444	-0.792825	-0.744964	-0.
ln_global_rate_negative_words	13875.0	-0.004984	0.998803	-1.553284	-0.648014	-0.
ln_min_positive_polarity	13875.0	-0.010174	0.982560	-1.481184	-0.671513	0.1
ln_abs_title_sentiment_polarity	13875.0	0.000694	1.004252	-0.740381	-0.740381	-0.

In [67]:
```
# set required variables for model comparison

kmeans_tbl = pd.DataFrame(columns = [
    'model_name',
    'n_clusters',
    'inertia',
    'silhouette',
    'process_time'])

i_index = []
i_index = 0

# preparation for cross validation and model comparison, each classifier is
appended once model is fit

models = []
```

Table of Contents

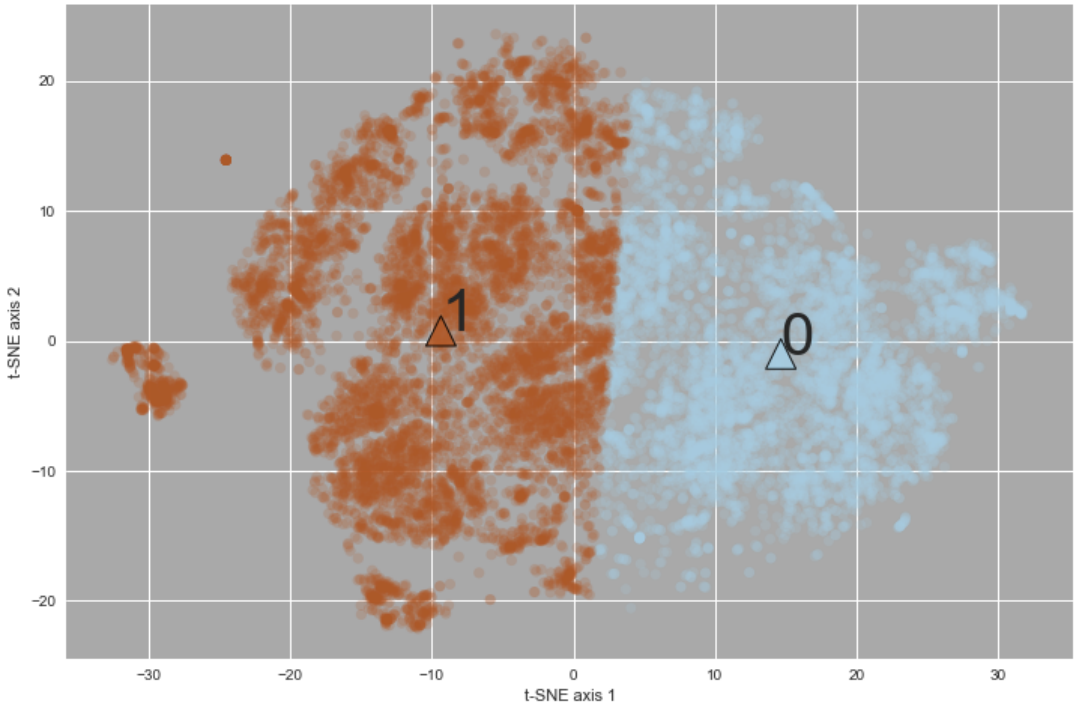
K-Means

K-Means

In [68]:
```
# ... k-means on the t-sne vectors

X_tsne = pd.DataFrame(columns=['t1', 't2'])
X_tsne['t1'] = df_join['x-tsne']
X_tsne['t2'] = df_join['y-tsne']

for n_lda in range(2, 21):

    tic = time.clock()

    print ("n_lda = ", n_lda)

    cls_lda = KMeans(n_clusters = n_lda,
                     init = 'k-means++',
                     random_state = 1);

    cls_lda.fit(X_tsne)

    kmeans_labels = cls_lda.labels_ # the labels from kmeans clustering
    kmeans_centers = cls_lda.cluster_centers_

    kmeans_inertia = cls_lda.inertia_
    print ("inertia = ", kmeans_inertia)

    kmeans_silhouette = metrics.silhouette_score(X_tsne,
                                                 kmeans_labels,
                                                 metric = 'euclidean',
                                                 sample_size = 10000)
    print ("silhouette = ", kmeans_silhouette)

    toc =  time.clock()
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... - save statistics for model comparison
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

    exe_time = '{0:.4f}'.format(toc-tic)

    raw_data = {
    'model_name' : 'KMeans - LDA features',
    'n_clusters' : n_lda,
    'inertia': kmeans_inertia,
    'silhouette': kmeans_silhouette,
    'process_time' : exe_time
    }

    df_tbl = pd.DataFrame(raw_data,
    columns = ['model_name', 'n_clusters', 'inertia', 'silhouette', 'process_time'],
    index = [i_index + 1])

    kmeans_tbl = kmeans_tbl.append(df_tbl)

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... - make some plots of clusters
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

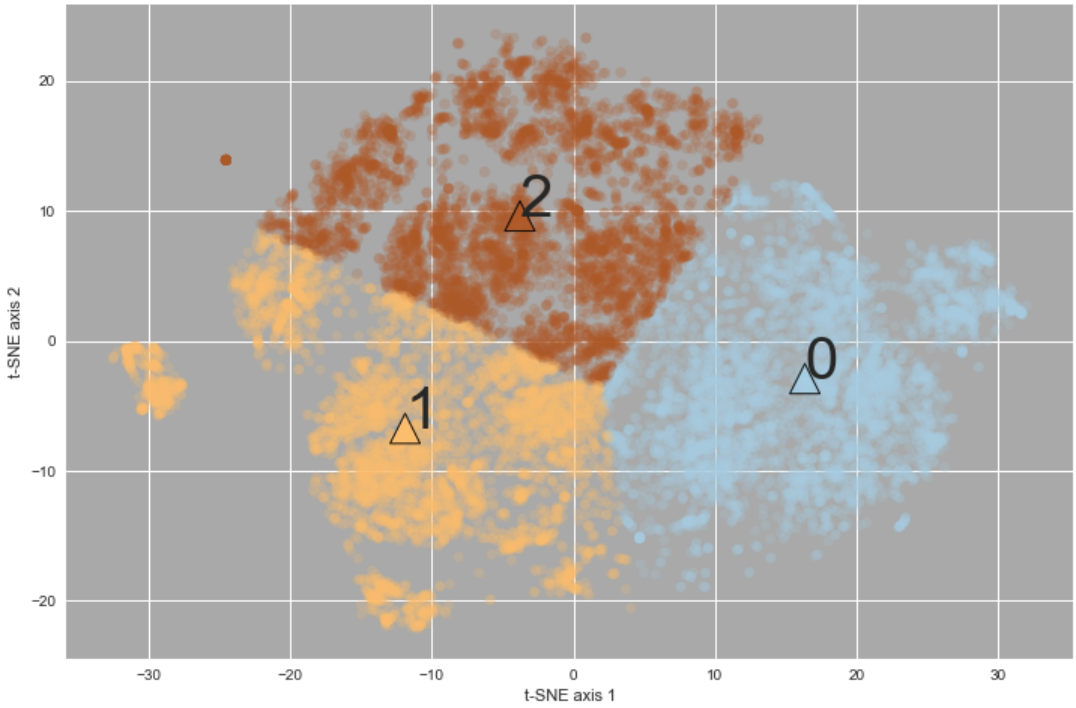
```
n_lda =  2
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)

inertia =  2070712.02857
silhouette =  0.41842916379
0 14.6668996548 -1.00877648916 0
1 -9.37160510965 0.759730709041 1
```




```
n_lda =  3
```

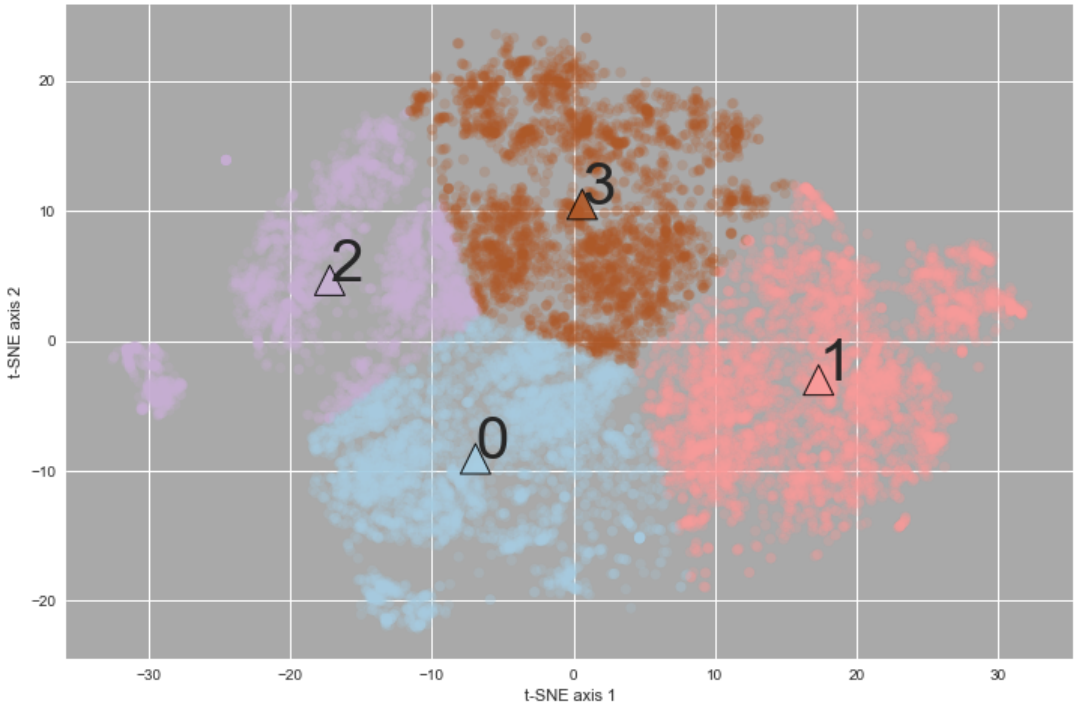
Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)

inertia =  1377777.29348
silhouette =  0.405665381546
0 16.3612249955 -2.91257777138 0
1 -11.9053093097 -6.74309905478 1
2 -3.78550261429 9.60119248758 2
```




```
n_lda =  4
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

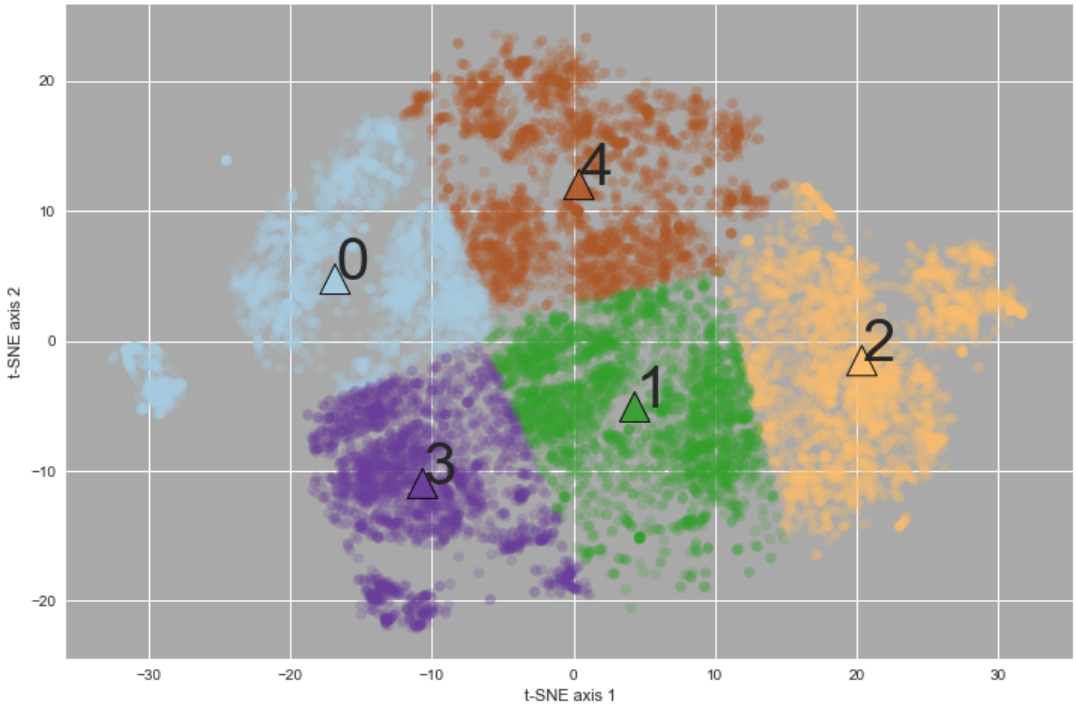
```
inertia =  1012812.1291
silhouette =  0.409536768316
0 -6.92731471166 -9.0766692136 0
1 17.3294754146 -3.02146770398 1
2 -17.2339618956 4.64870876718 2
3 0.617149042961 10.4946541961 3
```




```
n_lda =  5
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=5, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

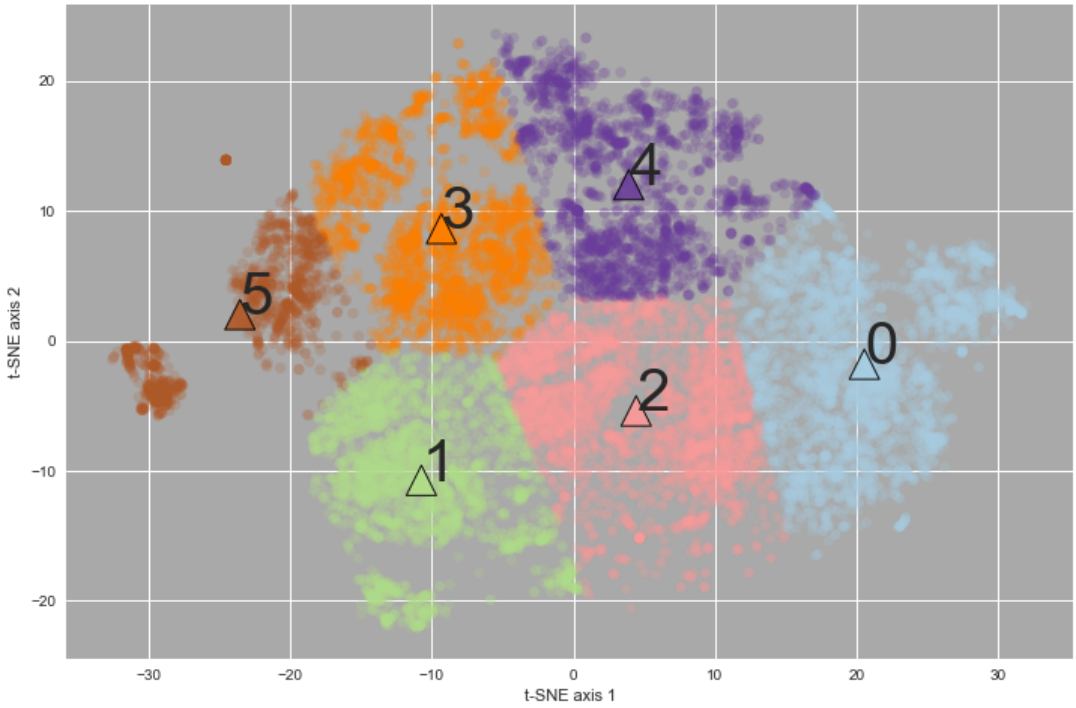
```
inertia =  815255.608738
silhouette =  0.381802080795
0 -16.8563549859 4.74538831632 0
1 4.32933844922 -5.08798799357 1
2 20.392274692 -1.53088401667 2
3 -10.6695593011 -10.9715199592 3
4 0.386072065819 12.0179964869 4
```




```
n_lda =  6
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=6, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

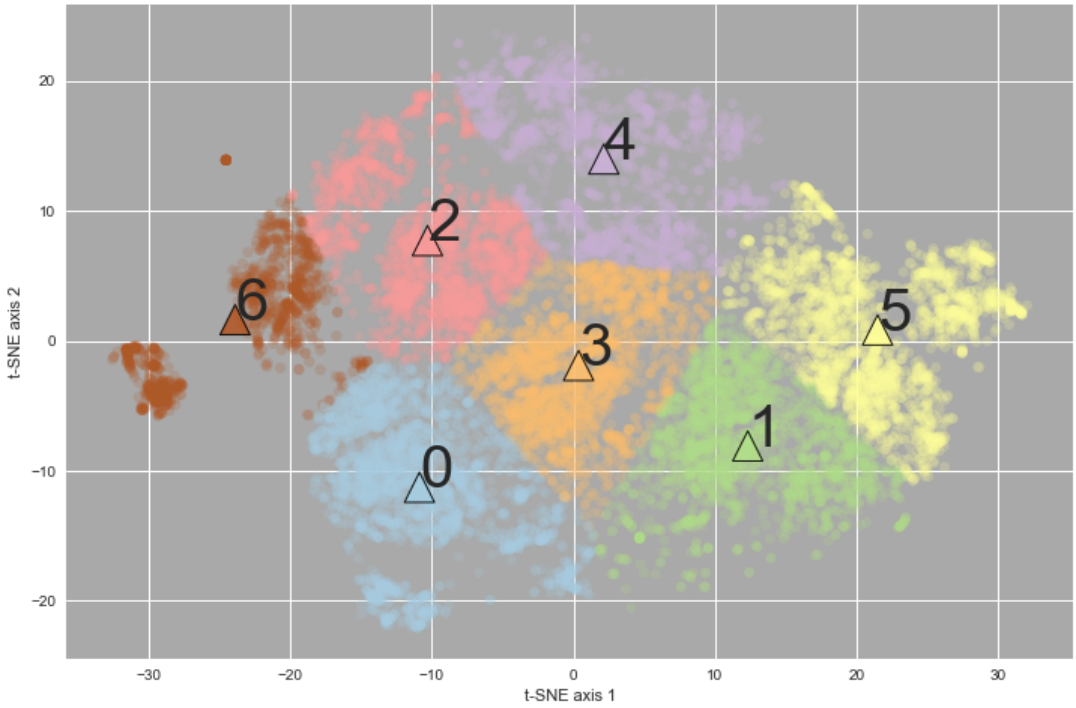
```
inertia =  681787.724205
silhouette =  0.390416438959
0 20.5665000771 -1.80188993641 0
1 -10.7531469228 -10.7389639845 1
2 4.43364639948 -5.39376153134 2
3 -9.33757256467 8.62785302864 3
4 3.90678392421 12.0034300871 4
5 -23.5816914773 2.01589642483 5
```




```
n_lda =  7
```

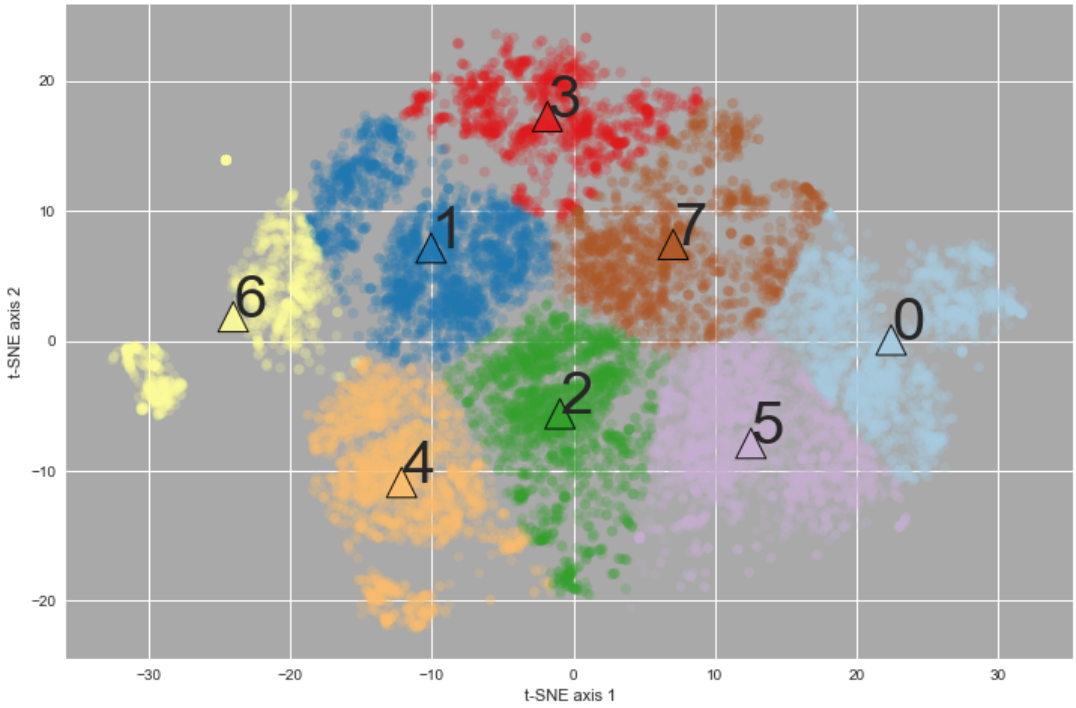
Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=7, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

```
inertia =  572099.625581
silhouette =  0.381918951414
0 -10.8883170036 -11.2712188963 0
1 12.3250678982 -8.0896783909 1
2 -10.3210413115 7.70578926628 2
3 0.363387871227 -1.90066399043 3
4 2.1445568887 13.9808181398 4
5 21.5047706021 0.832370278598 5
6 -23.9282176655 1.6196825843 6
```




```
n_lda =  8
```

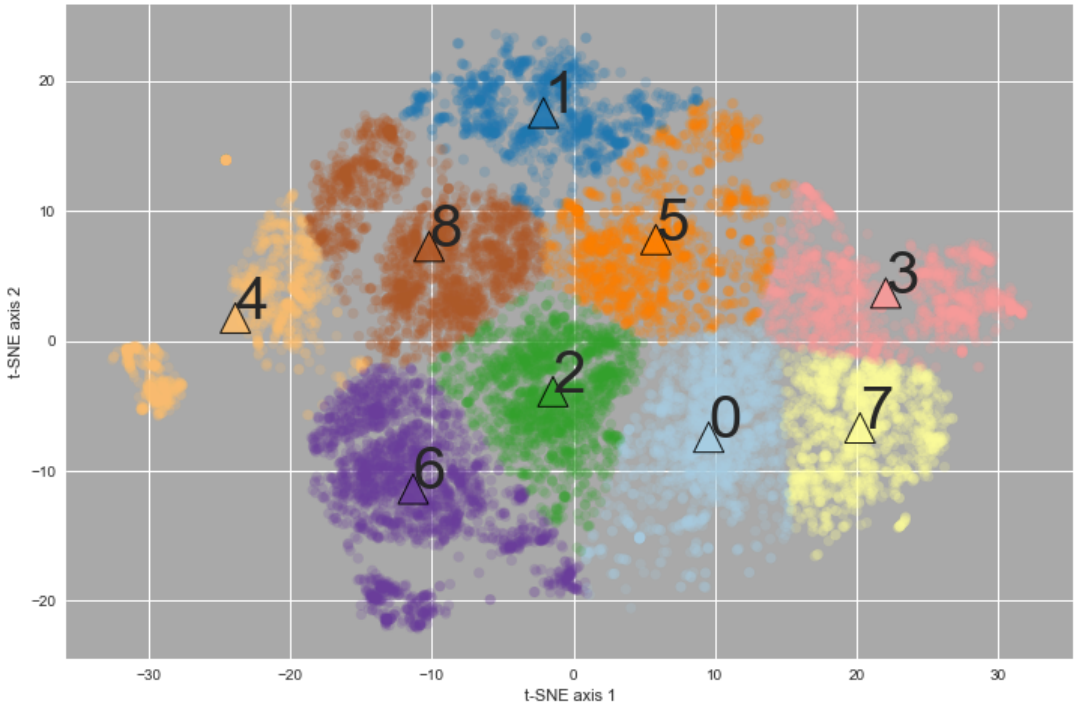
Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=8, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)

inertia =  492596.904026
silhouette =  0.39225531454
0 22.4628979922 0.0630593300668 0
1 -10.0663524143 7.13627163857 1
2 -0.952445194248 -5.61211746601 2
3 -1.84131076404 17.244783894 3
4 -12.1486791764 -10.9018930303 4
5 12.5486353202 -7.89214743778 5
6 -24.0554560018 1.84721677018 6
7 7.05307374104 7.41159329644 7
```




```
n_lda =  9
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=9, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

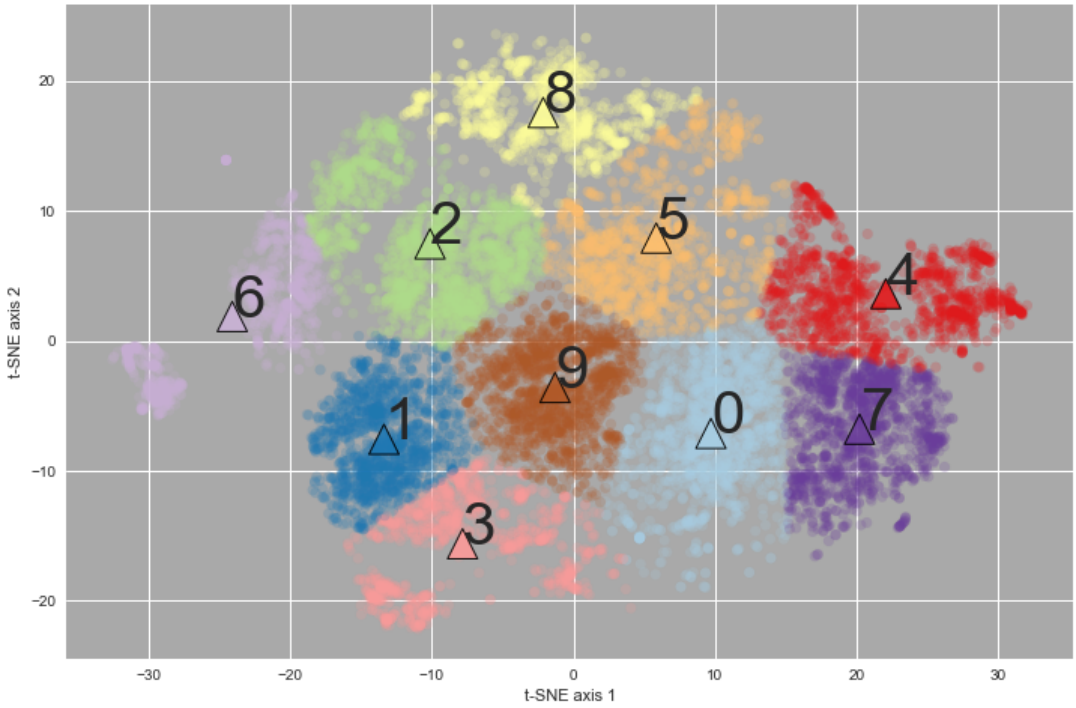
```
inertia =  440862.709477
silhouette =  0.389933901204
0 9.56432585778 -7.41961051149 0
1 -2.118048205 17.527580424 1
2 -1.44771823155 -3.90916990563 2
3 22.0868799808 3.66904377108 3
4 -23.9040905823 1.74918671819 4
5 5.8286591556 7.78496109103 5
6 -11.3324079796 -11.3946344746 6
7 20.2606597311 -6.70628429327 7
8 -10.2113007651 7.25372972869 8
```




```
n_lda =  10
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=10, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

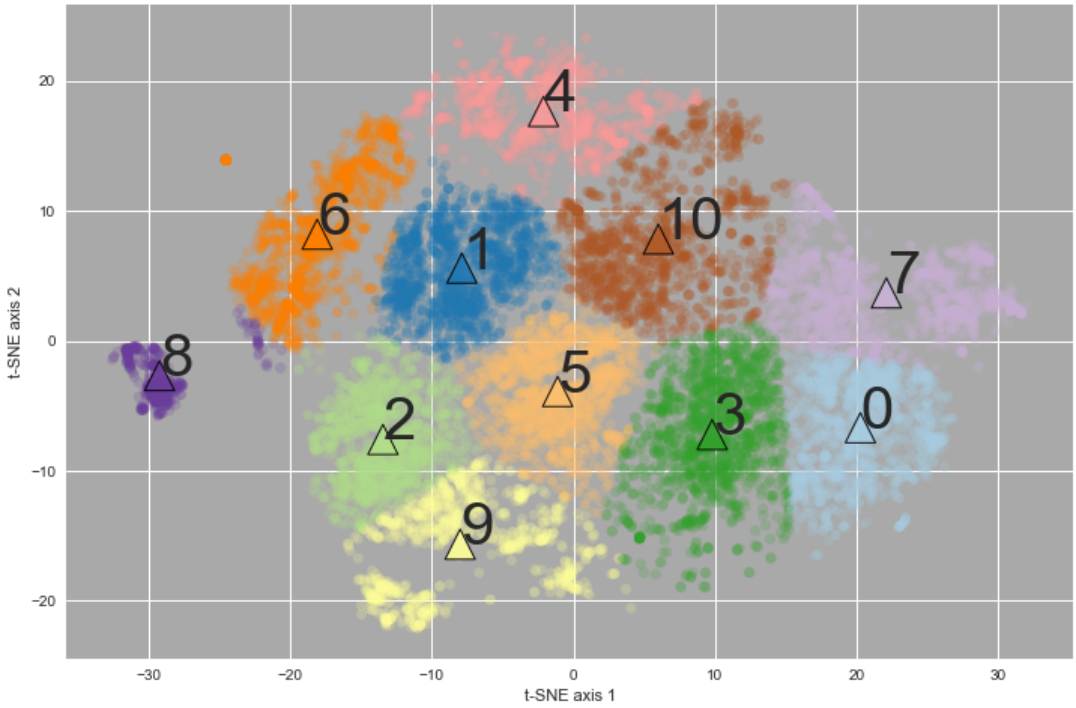
```
inertia =  390726.744467
silhouette =  0.386973771979
0 9.71820902639 -7.12458485323 0
1 -13.3992191538 -7.57617481751 1
2 -10.1429274621 7.45853125952 2
3 -7.84186061529 -15.6146571846 3
4 22.090191087 3.62035702581 4
5 5.85170505628 7.90148254424 5
6 -24.1253546952 1.86473165569 6
7 20.2290941814 -6.79772780938 7
8 -2.14269021607 17.5554106807 8
9 -1.3031917726 -3.5931532703 9
```




```
n_lda =  11
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=11, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

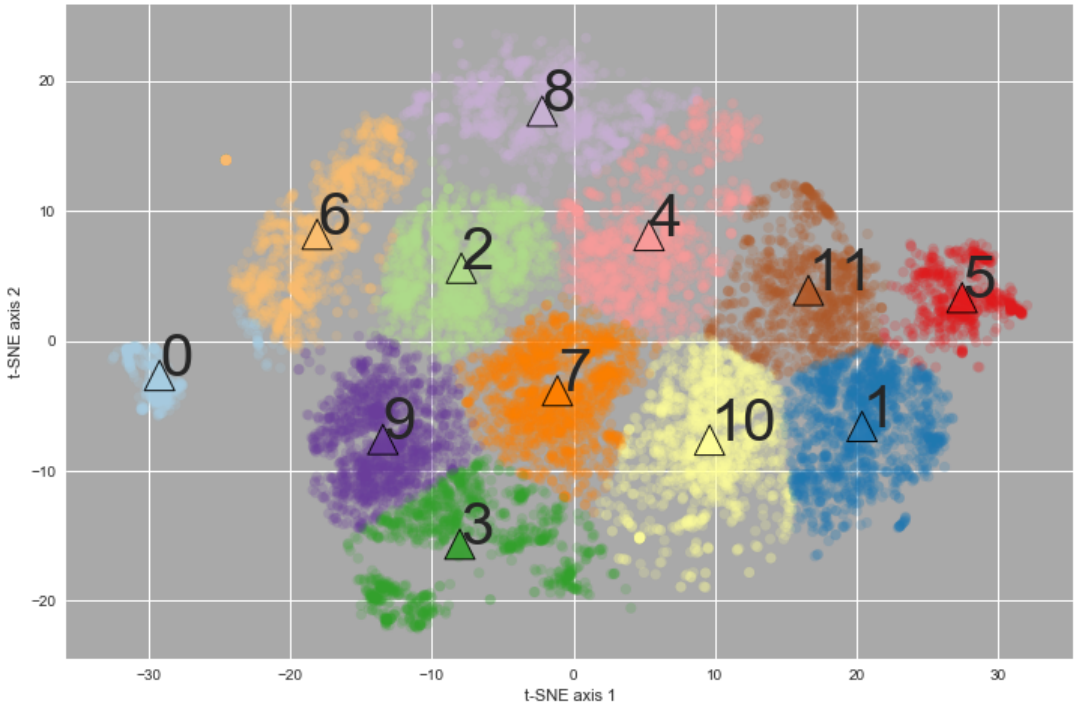
```
inertia =  341697.605188
silhouette =  0.405464167084
0 20.2875483005 -6.68771237431 0
1 -7.890108043 5.59020476815 1
2 -13.4719936162 -7.58872462517 2
3 9.8086070276 -7.21709625234 3
4 -2.12210091313 17.6413424727 4
5 -1.11717675585 -3.89953893611 5
6 -18.1177319714 8.19414182337 6
7 22.1269271469 3.68083122395 7
8 -29.2660928005 -2.67297381175 8
9 -8.01258277539 -15.6512734216 9
10 6.0018050164 7.82340136649 10
```




```
n_lda =  12
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=12, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

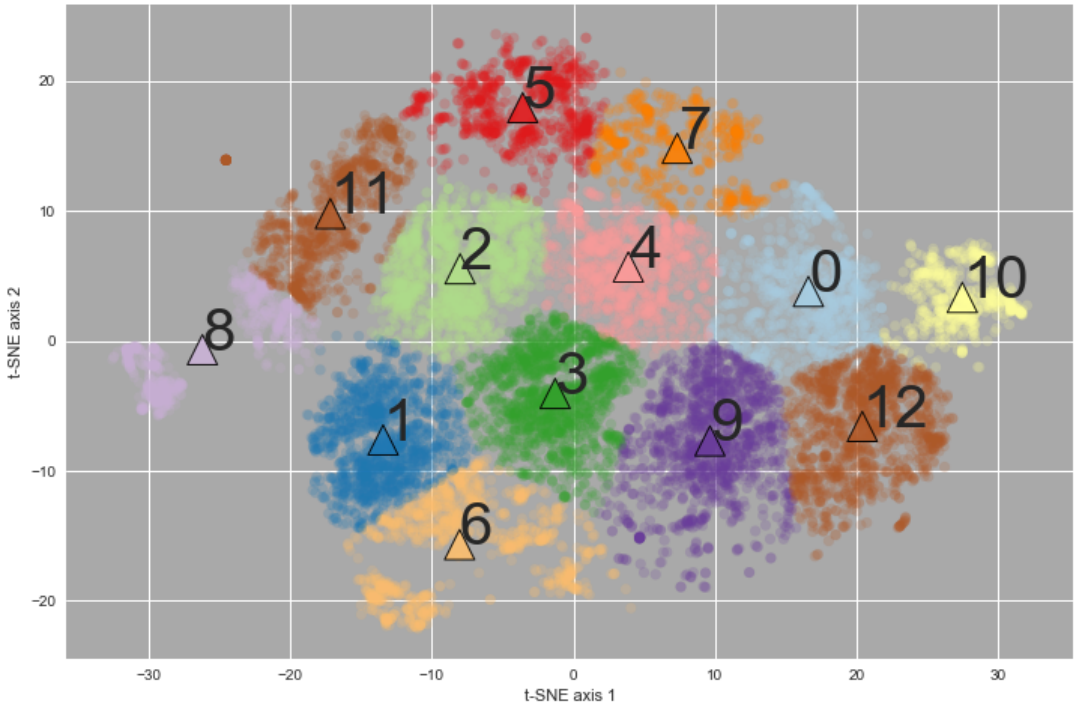
```
inertia =  305172.622725
silhouette =  0.419961335307
0 -29.2660928005 -2.67297381175 0
1 20.4039269975 -6.59370594566 1
2 -7.92596942326 5.5853949982 2
3 -8.03451378266 -15.6491353931 3
4 5.34451226642 8.09291576855 4
5 27.4788849834 3.31719812754 5
6 -18.1229500543 8.18766181619 6
7 -1.13185339259 -3.84154088284 7
8 -2.2124962332 17.6606515308 8
9 -13.4719936162 -7.58872462517 9
10 9.6256711466 -7.64328615439 10
11 16.6130916098 3.87446438603 11
```




```
n_lda =  13
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=13, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

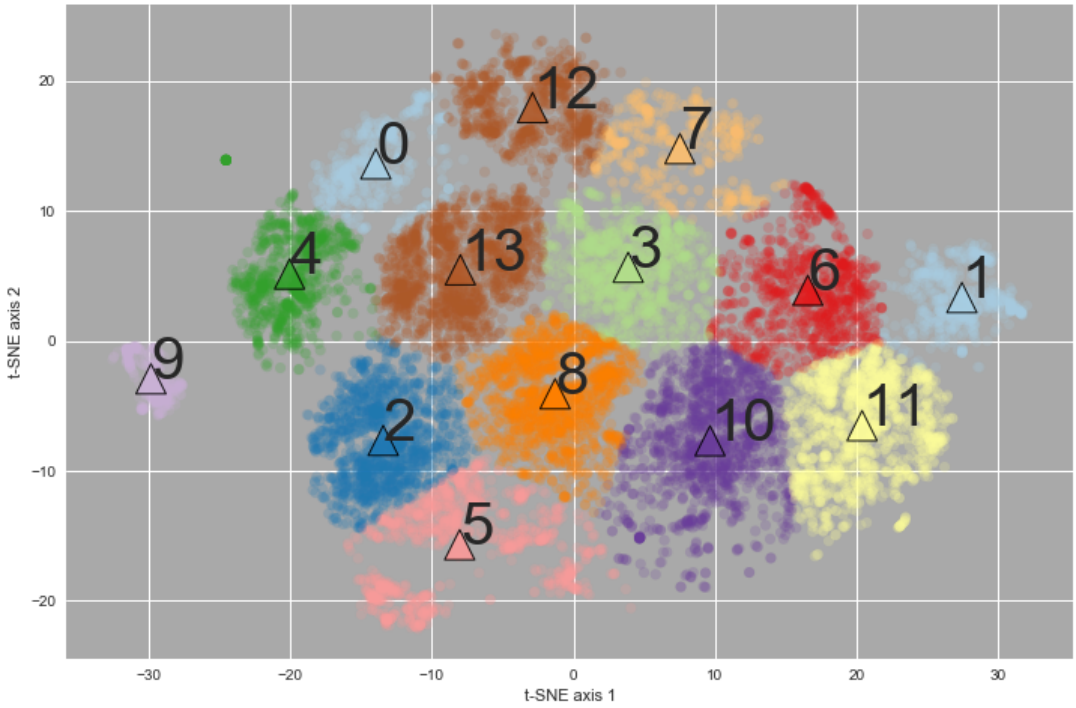
```
inertia =  276464.63328
silhouette =  0.418638699702
0 16.6128323313 3.79229018044 0
1 -13.460106102 -7.62534711326 1
2 -8.01620264311 5.53670872351 2
3 -1.29509748313 -4.05384898365 3
4 3.88030912329 5.67625975235 4
5 -3.58874704555 17.9327505289 5
6 -8.05043273105 -15.6794728258 6
7 7.33842850009 14.8021557967 7
8 -26.2359335505 -0.686514503472 8
9 9.6537357145 -7.70211055569 9
10 27.492974196 3.33053740171 10
11 -17.1893659641 9.78104160868 11
12 20.4225956939 -6.5886027333 12
```




```
n_lda =  14
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=14, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

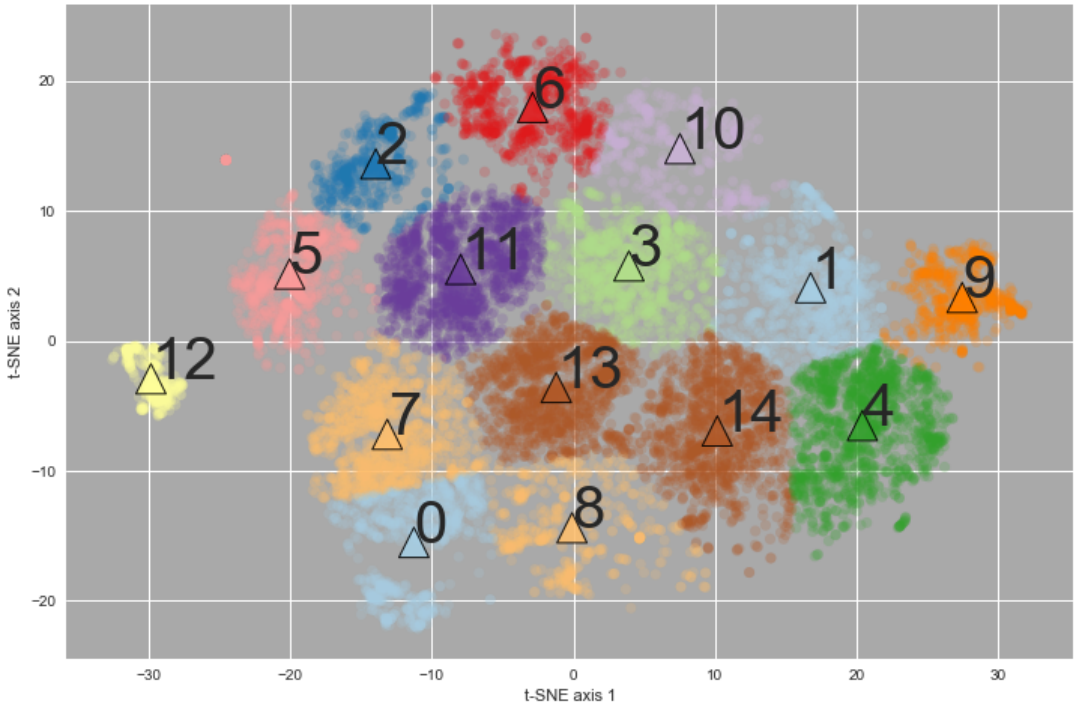
```
inertia =  249174.814922
silhouette =  0.429639773771
0 -13.9768522069 13.6074522702 0
1 27.4657450932 3.32493312053 1
2 -13.4516132894 -7.66427373006 2
3 3.87161812961 5.65046761259 3
4 -20.0731576473 5.13535725769 4
5 -8.04642100533 -15.6882462764 5
6 16.5718555107 3.88331037698 6
7 7.53015651328 14.7737772465 7
8 -1.29889326462 -4.08028898328 8
9 -29.8751777639 -2.92615552292 9
10 9.65878250032 -7.69448649046 10
11 20.4076072557 -6.53004385277 11
12 -2.89547500882 17.9312641711 12
13 -7.99609805535 5.45789708793 13
```




```
n_lda =  15
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=15, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

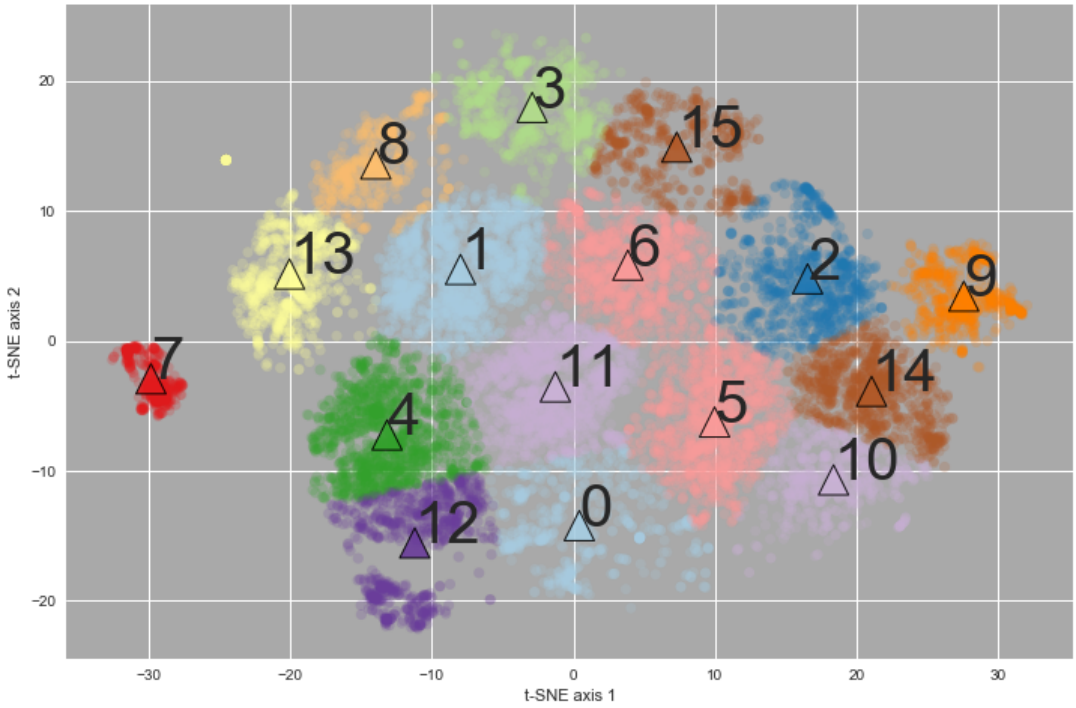
```
inertia =  225681.949289
silhouette =  0.432241898745
0 -11.2813892334 -15.5124128142 0
1 16.7700063956 4.07676749269 1
2 -13.9768522069 13.6074522702 2
3 3.92241174798 5.75168608383 3
4 20.4301202058 -6.52272738093 4
5 -20.0731395594 5.1387558646 5
6 -2.89367093672 17.9482343101 6
7 -13.1662533543 -7.20162914696 7
8 -0.104159746954 -14.3877740498 8
9 27.483784838 3.32607771437 9
10 7.53015651328 14.7737772465 10
11 -7.97463985207 5.49317035791 11
12 -29.8751777639 -2.92615552292 12
13 -1.21511042505 -3.59362407486 13
14 10.1692148185 -6.93527606437 14
```




```
n_lda =  16
```

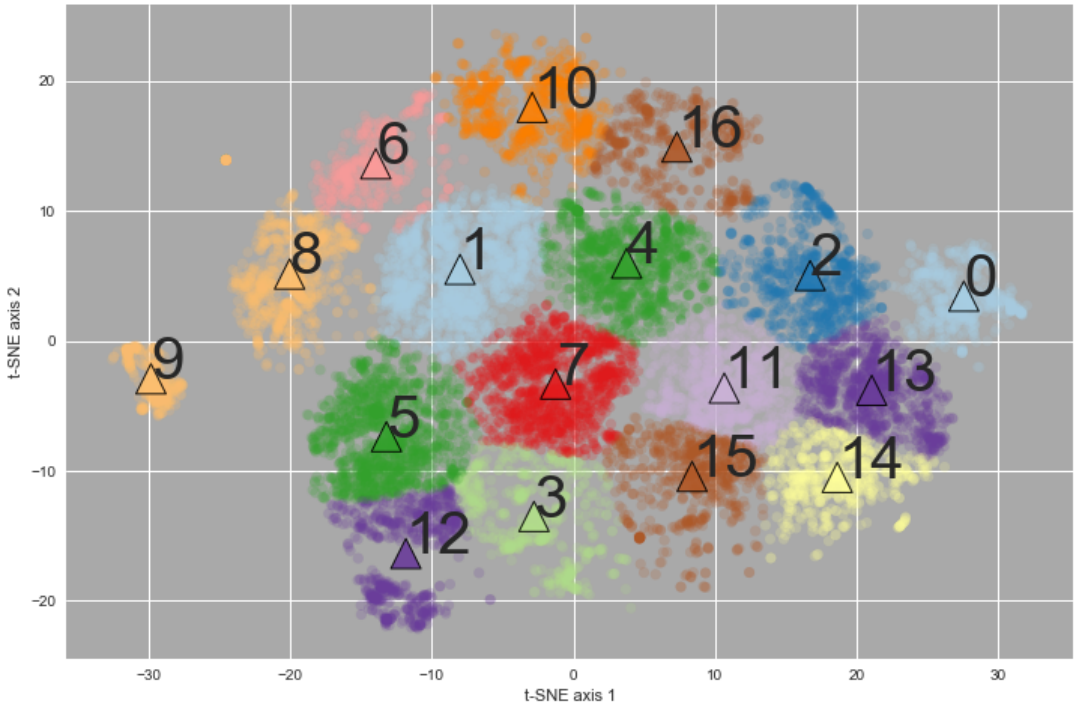
Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=16, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

```
inertia =  209032.662416
silhouette =  0.430820201045
0 0.413560853459 -14.2202666767 0
1 -7.98600213542 5.48780621647 1
2 16.5567111934 4.76678786827 2
3 -2.91742131505 17.9499191999 3
4 -13.1794344998 -7.2402074136 4
5 9.9783960599 -6.24981541699 5
6 3.84654793831 5.81948408412 6
7 -29.8751777639 -2.92615552292 7
8 -13.9768522069 13.6074522702 8
9 27.6013175822 3.42938696465 9
10 18.3900263877 -10.699234561 10
11 -1.28173072679 -3.54389509377 11
12 -11.223389814 -15.5494176157 12
13 -20.0807202093 5.14126772688 13
14 21.0795495439 -3.88765145316 14
15 7.30497668183 14.8956916382 15
```



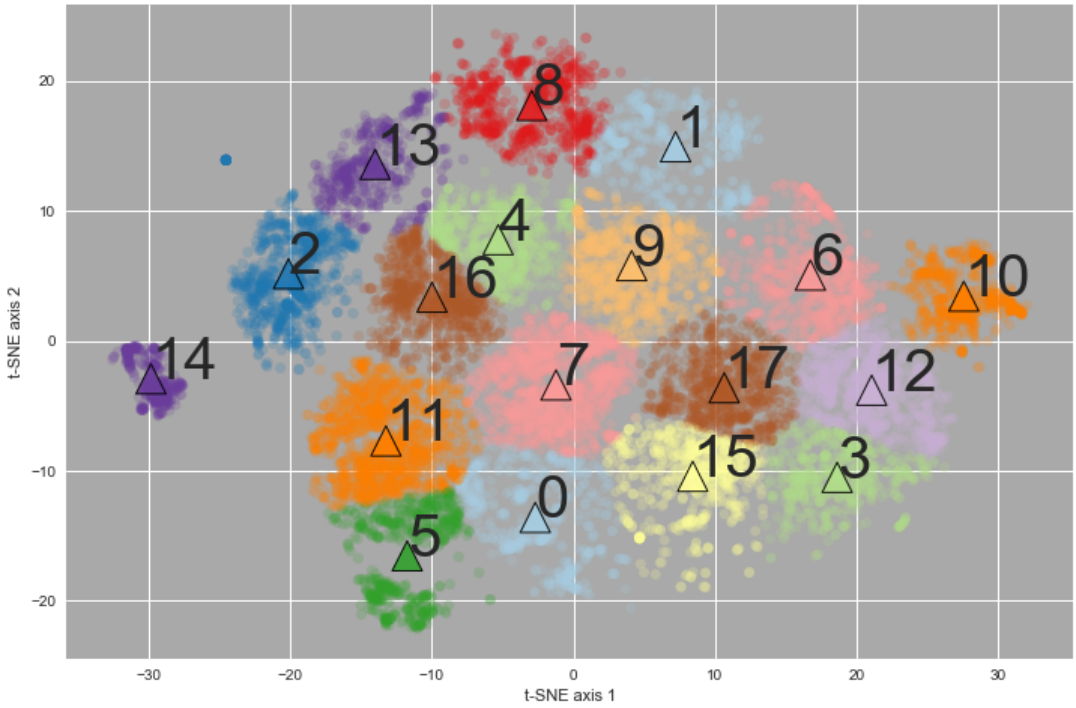

```
n_lda =  17
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=17, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)

inertia =  194316.264578
silhouette =  0.429383928018
0 27.6013175822 3.42938696465 0
1 -8.02188806787 5.4933876381 1
2 16.7382527662 5.02235589077 2
3 -2.79179951931 -13.5161484295 3
4 3.75928639008 5.96814789496 4
5 -13.2222072202 -7.37690723827 5
6 -13.9793591142 13.6175906352 6
7 -1.26182461654 -3.32942917796 7
8 -20.0731576473 5.13535725769 8
9 -29.8751777639 -2.92615552292 9
10 -2.91742131505 17.9499191999 10
11 10.6646430883 -3.63896601622 11
12 -11.8491506919 -16.368137221 12
13 21.0911062757 -3.81048123798 13
14 18.6532927311 -10.5130680163 14
15 8.38772497117 -10.4368855305 15
16 7.31317453511 14.9024113557 16
```



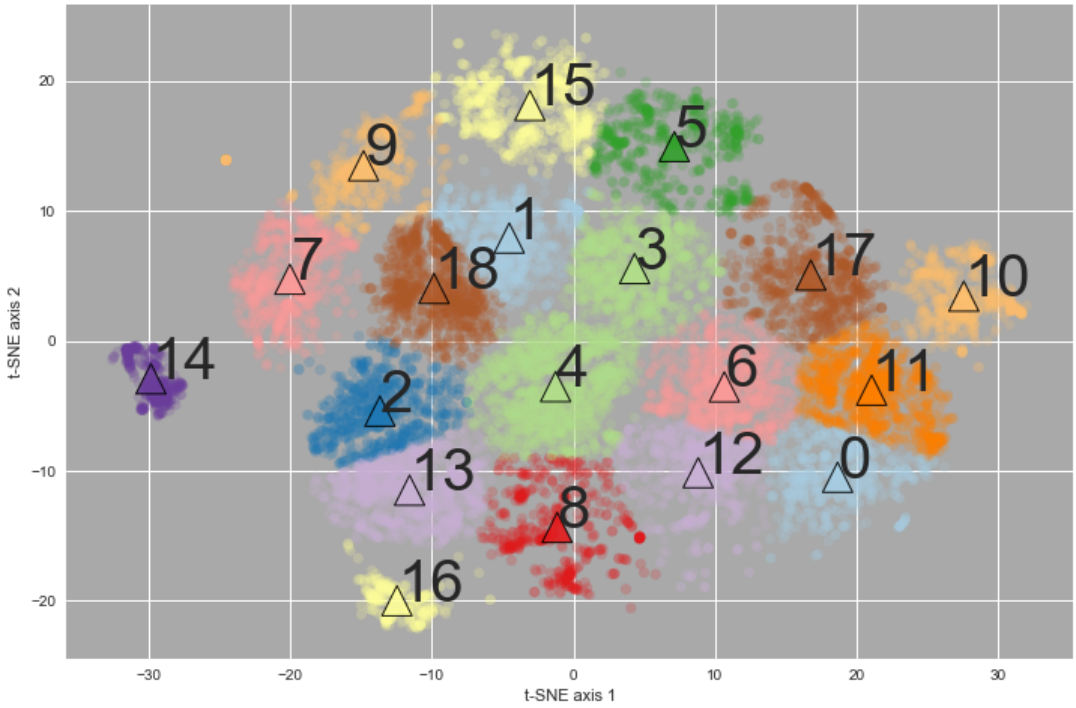

```
n_lda =  18
```

Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=18, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)

inertia =  180360.344098
silhouette =  0.426188266338
0 -2.70107952043 -13.5870602839 0
1 7.22128555022 14.9229802476 1
2 -20.1490377601 5.16093915361 2
3 18.6458299083 -10.5078912694 3
4 -5.32420851717 7.7761353801 4
5 -11.7562640898 -16.5411553894 5
6 16.7563000034 5.046733018 6
7 -1.23409034822 -3.39871103839 7
8 -2.95231563694 18.0988248109 8
9 4.11240169648 5.77841505911 9
10 27.6013175822 3.42938696465 10
11 -13.2571445537 -7.70397330589 11
12 21.0858195715 -3.80167846544 12
13 -14.0245507175 13.5678087121 13
14 -29.8751777639 -2.92615552292 14
15 8.43378877938 -10.4292003701 15
16 -10.0035278516 3.35588575095 16
17 10.6599728484 -3.61122134096 17
```




```
n_lda =  19
```

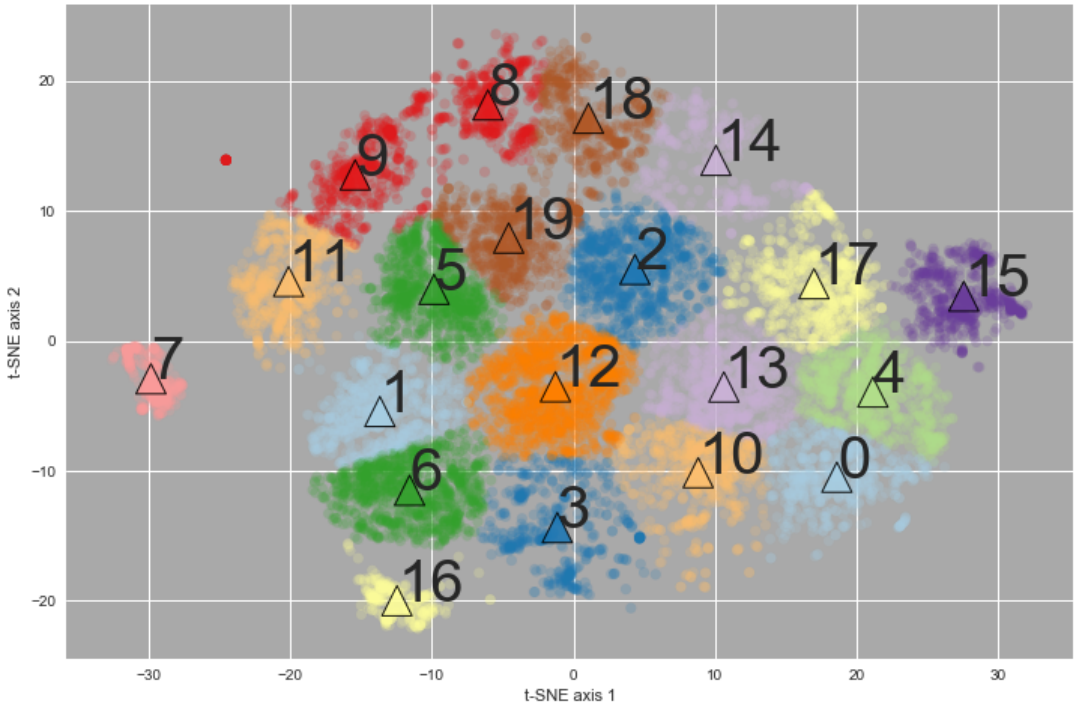
Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=19, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)

inertia =  168733.96683
silhouette =  0.42780557717
0 18.6776913501 -10.4975620707 0
1 -4.53827750324 7.90395518427 1
2 -13.6821064821 -5.41491587355 2
3 4.32323457786 5.55134598293 3
4 -1.26437491302 -3.52243031092 4
5 7.13686298421 14.9375507404 5
6 10.6680707415 -3.54454178221 6
7 -20.0529945845 4.73487777596 7
8 -1.15009598061 -14.3571533749 8
9 -14.8423685126 13.4643113743 9
10 27.6013175822 3.42938696465 10
11 21.0846440675 -3.79732300387 11
12 8.81954959259 -10.1504959343 12
13 -11.5886565916 -11.495498514 13
14 -29.8751777639 -2.92615552292 14
15 -3.08072916976 18.1027628141 15
16 -12.4787139126 -19.9983340945 16
17 16.7863301026 5.04895017278 17
18 -9.86134960799 4.0229410542 18
```




```
n_lda =  20
```

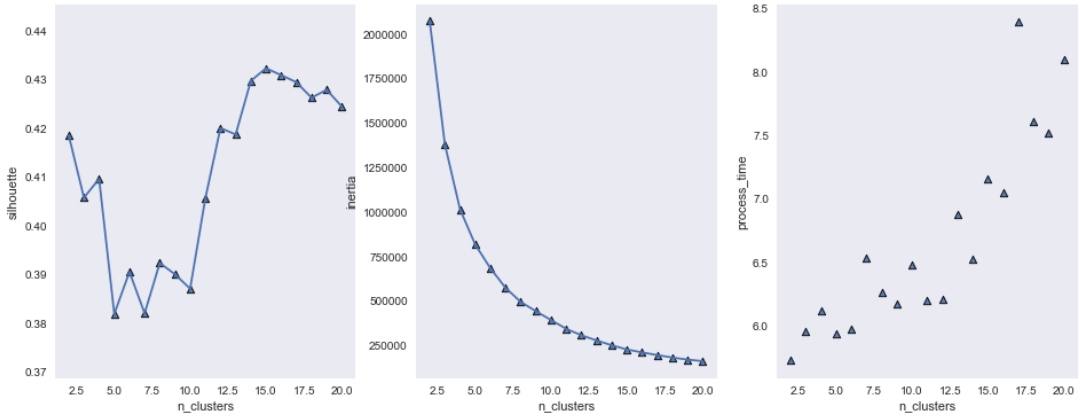
Out[68]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=20, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

```
inertia =  160244.495743
silhouette =  0.424376458384
0 18.62591689 -10.4920787108 0
1 -13.6864108917 -5.44152260136 1
2 4.34568199426 5.48664591144 2
3 -1.14212577728 -14.3686275053 3
4 21.1711699427 -3.94374090467 4
5 -9.85875953323 3.95514419619 5
6 -11.5873829513 -11.5018139615 6
7 -29.8751777639 -2.92615552292 7
8 -6.0474765727 18.1415476214 8
9 -15.4327442538 12.7589184542 9
10 8.81954959259 -10.1504959343 10
11 -20.1414252901 4.56331791433 11
12 -1.27331138556 -3.53769141216 12
13 10.6369832214 -3.55578519805 13
14 10.0771508714 13.8772617563 14
15 27.6013175822 3.42938696465 15
16 -12.4787139126 -19.9983340945 16
17 17.008922481 4.38364048605 17
18 1.05915701389 17.1155977412 18
19 -4.58166900978 7.90020623337 19
```


8
18
14
9
19
2
11
5
17
15
7
12
13
4
1
10
0
6
3
16
t-SNE axis 2
t-SNE axis 1
20
10
0
−10
−20
−30
−20
−10
0
10
20
30

In [69]:

```
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... - plot metrics across models for comparison
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

plt.figure(figsize=(16, 6));

# ... silhouette values

plt.subplot(131);
plt.scatter(kmeans_tbl['n_clusters'],
            kmeans_tbl['silhouette'],
            s = 40,
            linewidths = 1.0,
            marker = '^',
            edgecolors = 'black',
            alpha = 0.90);

plt.plot(kmeans_tbl['n_clusters'],
         kmeans_tbl['silhouette'])

plt.xlabel('n_clusters'), plt.ylabel('silhouette');
plt.grid();

# ... inertia values

plt.subplot(132);
plt.scatter(kmeans_tbl['n_clusters'],
            kmeans_tbl['inertia'],
            s = 40,
            linewidths = 1.0,
            marker = '^',
            edgecolors = 'black',
            alpha = 0.90);

plt.plot(kmeans_tbl['n_clusters'],
         kmeans_tbl['inertia'])

plt.xlabel('n_clusters'), plt.ylabel('inertia');
plt.grid();

# ... process time

plt.subplot(133);
plt.scatter(kmeans_tbl['n_clusters'],
            kmeans_tbl['process_time'],
            s = 40,
            linewidths = 1.0,
            marker = '^',
            edgecolors = 'black',
            alpha = 0.90);

#plt.plot(kmeans_tbl['n_clusters'],
#         kmeans_tbl['process_time'])

plt.xlabel('n_clusters'), plt.ylabel('process_time');
```



- choose kmeans clustering with preferred number of clusters

In [70]:
```
n_clusters_chosen = 4

for n_lda in [4, 15] :

    tic = time.clock()

    print ("n_lda = ", n_lda)

    cls_lda = KMeans(n_clusters = n_lda,
                     init = 'k-means++',
                     random_state = 1);

    cls_lda.fit(X_tsne)

    kmeans_labels = cls_lda.labels_ # the labels from kmeans clustering
    kmeans_centers = cls_lda.cluster_centers_

    kmeans_inertia = cls_lda.inertia_
    print ("inertia = ", kmeans_inertia)

    kmeans_silhouette = metrics.silhouette_score(X_tsne,
                                                 kmeans_labels,
                                                 metric = 'euclidean',
                                                 sample_size = 10000)
    print ("silhouette = ", kmeans_silhouette)

    toc =  time.clock()

# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# ... - make some plots of clusters
# ... -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

    plt.figure(figsize=(12, 12));
    ax = plt.gca();

    X_tsne_values = X_tsne.values;
    plt.scatter(X_tsne_values[:, 0], X_tsne_values[:, 1],
                c = df_join['popular'],
                cmap = plt.cm.Spectral,
                s = 50,
                linewidths = 0,
                alpha = 0.20);
    plt.scatter(kmeans_centers[:, 0], kmeans_centers[:, 1],
                c = range(n_lda),
                cmap = plt.cm.tab20,
                s = 400,
                linewidths = 1.0,
                marker = '^',
                edgecolors = 'black',
                alpha = 0.90);
    for ii in range(n_lda) :
        plt.text(kmeans_centers[ii, 0], kmeans_centers[ii, 1], ii, fontsize = 40)
        print(ii, kmeans_centers[ii, 0], kmeans_centers[ii, 1], ii)

    print(kmeans_labels)
```

```
n_lda =  4
```

Out[70]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

```
inertia =  1012812.1291
silhouette =  0.408384464173
```

Out[70]:
```
<matplotlib.figure.Figure at 0x7fdaa217b278>
```

Out[70]:
```
<matplotlib.collections.PathCollection at 0x7fdaa35d8198>
```

Out[70]:
```
<matplotlib.collections.PathCollection at 0x7fdaa35d89e8>
```

Out[70]:
```
<matplotlib.text.Text at 0x7fdaa2192e80>
```

```
0 -6.92731471166 -9.0766692136 0
```

Out[70]:
```
<matplotlib.text.Text at 0x7fdaa2192400>
```

```
1 17.3294754146 -3.02146770398 1
```

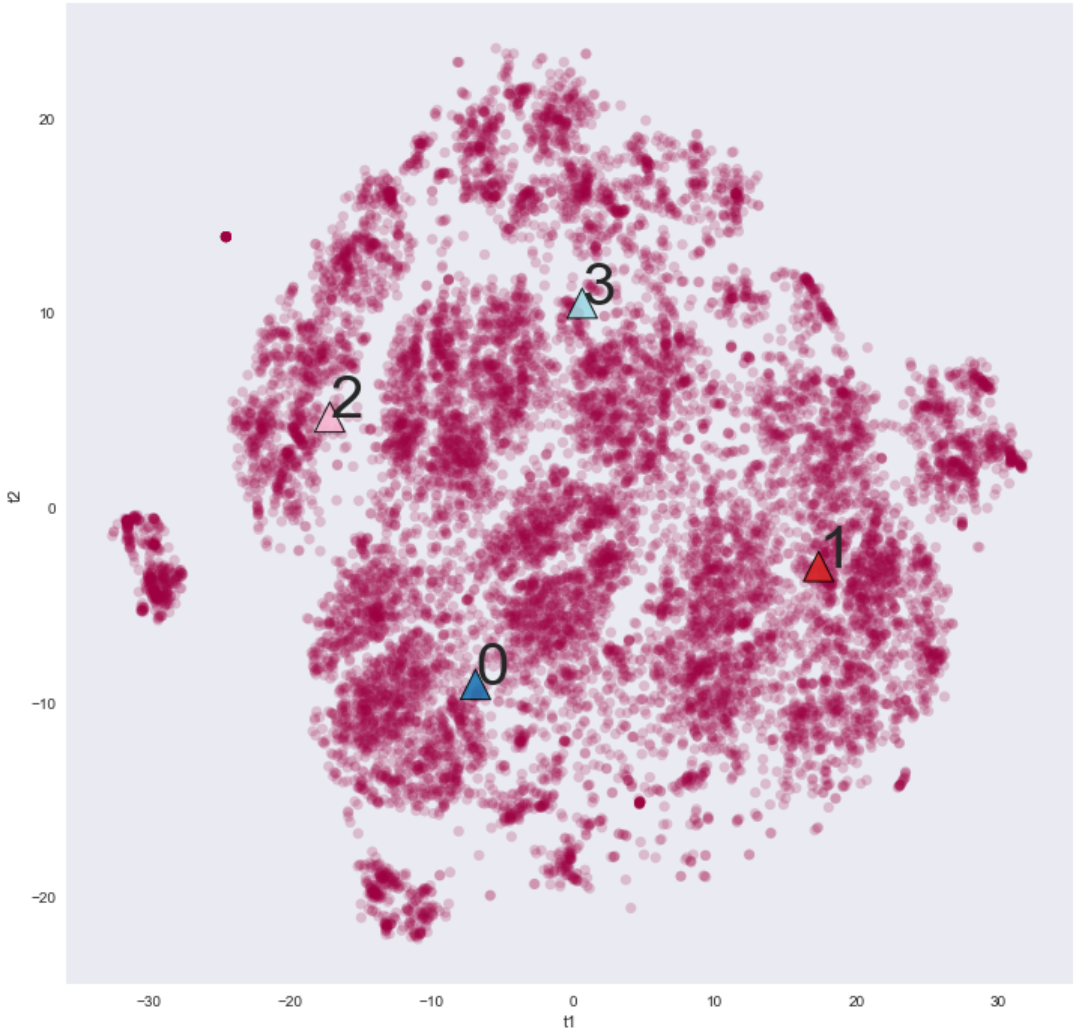
Out[70]:
```
<matplotlib.text.Text at 0x7fdaa232b668>
```

```
2 -17.2339618956 4.64870876718 2
```

Out[70]:
```
<matplotlib.text.Text at 0x7fdaa232b320>
```

```
3 0.617149042961 10.4946541961 3
[3 2 2 ..., 3 3 0]
[[ -6.92731471  -9.07666921]
 [ 17.32947541  -3.0214677 ]
 [-17.2339619    4.64870877]
 [  0.61714904  10.4946542 ]]
```

Out[70]:
```
(<matplotlib.text.Text at 0x7fdaa1e97be0>,
 <matplotlib.text.Text at 0x7fdaa222ed68>)
```




```
n_lda =  15
```

Out[70]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=15, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=1, tol=0.0001, verbose=0)
```

```
inertia =  225681.949289
silhouette =  0.433311740009
```

Out[70]:
```
<matplotlib.figure.Figure at 0x7fdaa3792198>
```

Out[70]:
```
<matplotlib.collections.PathCollection at 0x7fdaa21a0940>
```

Out[70]:
```
<matplotlib.collections.PathCollection at 0x7fdaa2442208>
```

Out[70]:
```
<matplotlib.text.Text at 0x7fdaa21a0b70>
```

```
0 -11.2813892334 -15.5124128142 0
```

Out[70]:
```
<matplotlib.text.Text at 0x7fdaa3694438>
```

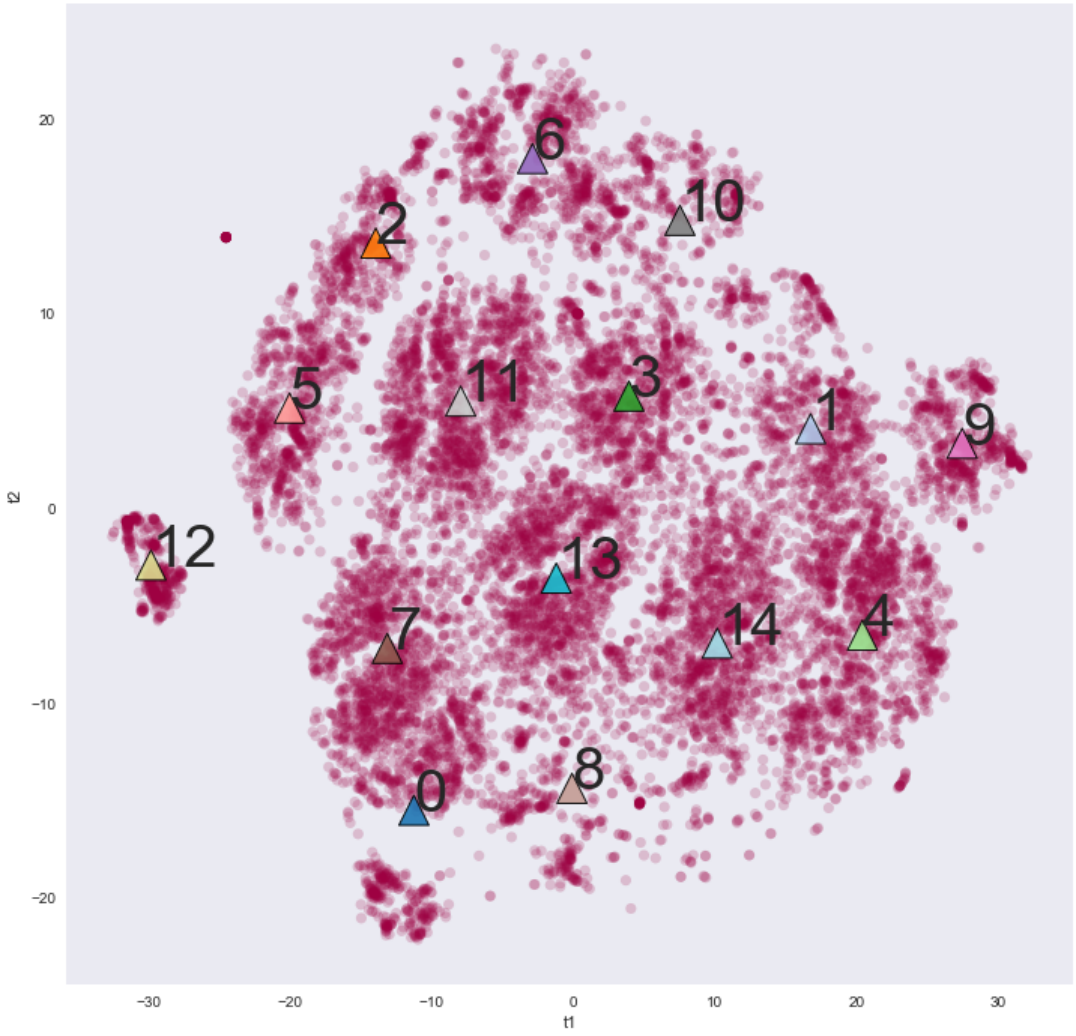
```
1 16.7700063956 4.07676749269 1
```

Out[70]:
```
<matplotlib.text.Text at 0x7fdaa3694160>
```

```
          2 -13.9768522069 13.6074522702 2

Out[70]: <matplotlib.text.Text at 0x7fdaa3694898>

          3 3.92241174798 5.75168608383 3

Out[70]: <matplotlib.text.Text at 0x7fdaa20db4e0>

          4 20.4301202058 -6.52272738093 4

Out[70]: <matplotlib.text.Text at 0x7fdaa20db978>

          5 -20.0731395594 5.1387558646 5

Out[70]: <matplotlib.text.Text at 0x7fdaa20dbba8>

          6 -2.89367093672 17.9482343101 6

Out[70]: <matplotlib.text.Text at 0x7fdaa3415b00>

          7 -13.1662533543 -7.20162914696 7

Out[70]: <matplotlib.text.Text at 0x7fdaa220d588>

          8 -0.104159746954 -14.3877740498 8

Out[70]: <matplotlib.text.Text at 0x7fdaa3694828>

          9 27.483784838 3.32607771437 9

Out[70]: <matplotlib.text.Text at 0x7fdaa20dbf28>

          10 7.53015651328 14.7737772465 10

Out[70]: <matplotlib.text.Text at 0x7fdab0e559e8>

          11 -7.97463985207 5.49317035791 11

Out[70]: <matplotlib.text.Text at 0x7fdaa20dbf98>

          12 -29.8751777639 -2.92615552292 12

Out[70]: <matplotlib.text.Text at 0x7fdab0824e10>

          13 -1.21511042505 -3.59362407486 13

Out[70]: <matplotlib.text.Text at 0x7fdab0824320>
```

```
14 10.1692148185 -6.93527606437 14
[10 11 12 ...,  3 11 13]
[[-11.28138923 -15.51241281]
 [ 16.7700064    4.07676749]
 [-13.97685221  13.60745227]
 [  3.92241175   5.75168608]
 [ 20.43012021  -6.52272738]
 [-20.07313956   5.13875586]
 [ -2.89367094  17.94823431]
 [-13.16625335  -7.20162915]
 [ -0.10415975 -14.38777405]
 [ 27.48378484   3.32607771]
 [  7.53015651  14.77377725]
 [ -7.97463985   5.49317036]
 [-29.87517776  -2.92615552]
 [ -1.21511043  -3.59362407]
 [ 10.16921482  -6.93527606]]
```

Out[70]:
```
(<matplotlib.text.Text at 0x7fdaa2488cf8>,
 <matplotlib.text.Text at 0x7fdaa2487f60>)
```



In [71]:
```
len(kmeans_labels)
```

Out[71]:
```
13875
```

In [72]:
```
X_all_together = copy.deepcopy(df_join)

len(X_all_together)

X_all_together['kmeans_labels'] = kmeans_labels

X_all_together.describe().T
```

Out[72]: 13875

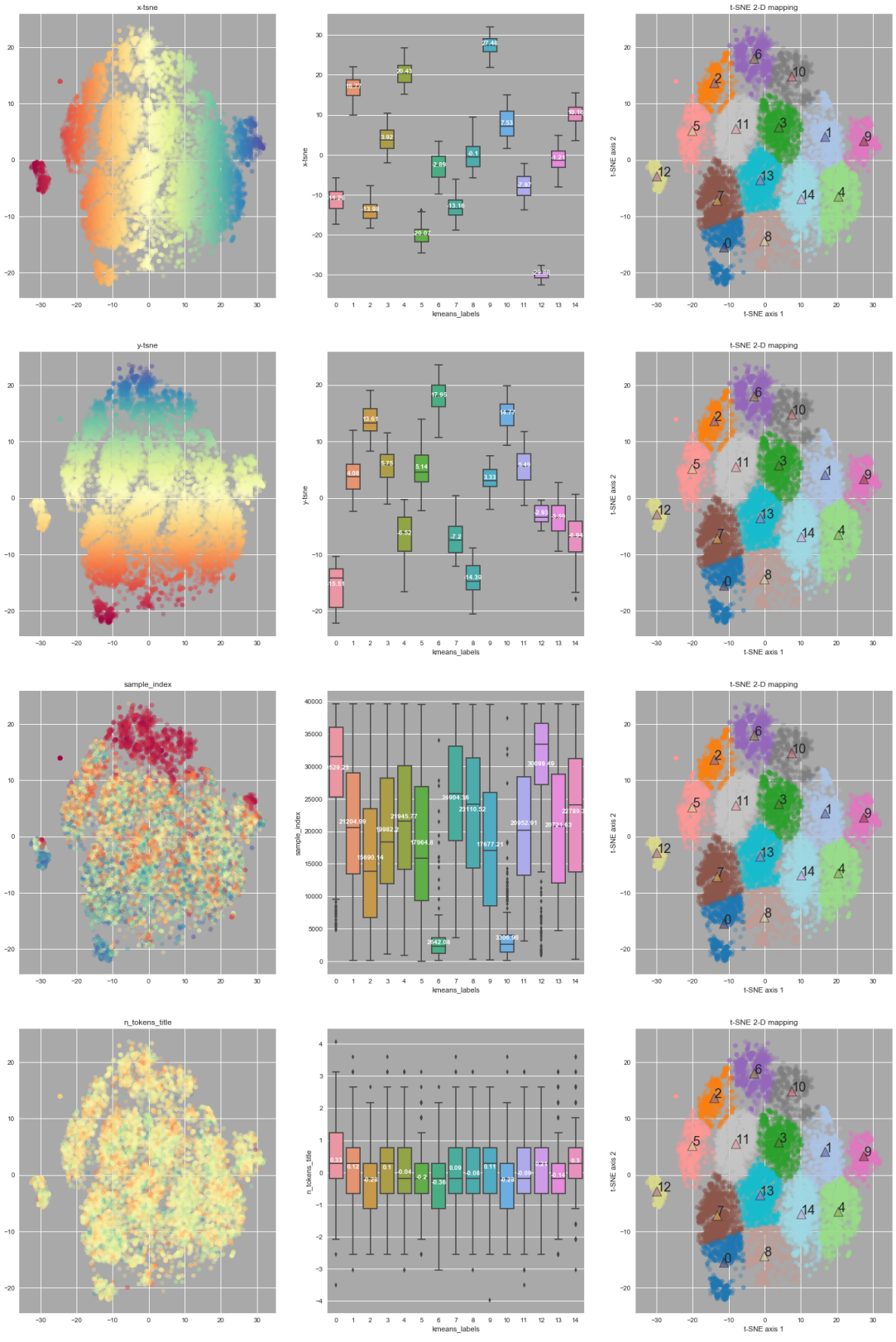
Out[72]:

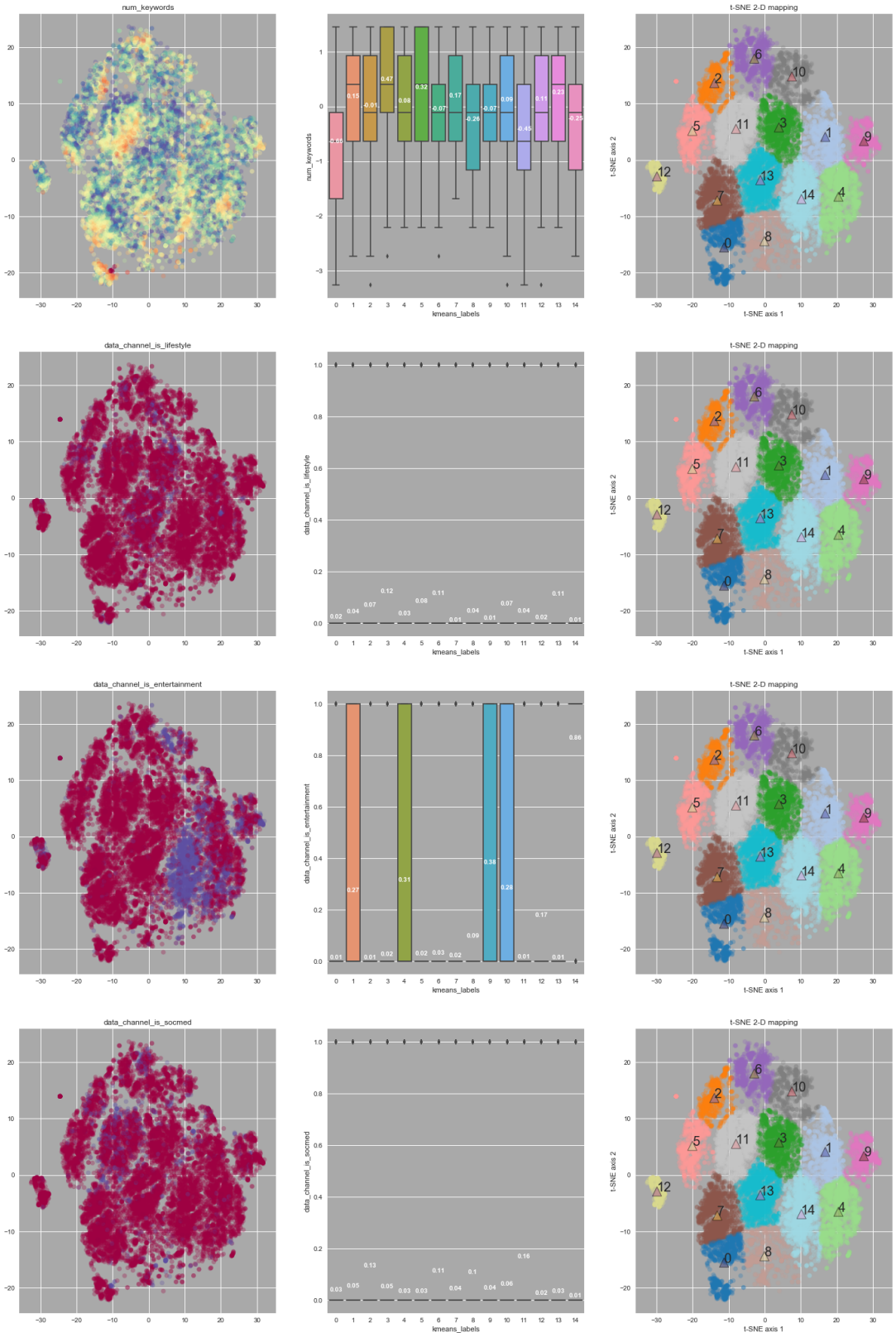
	count	mean	std	min	
x-tsne	13875.0	0.410117	14.157297	-32.599712	-10.7
y-tsne	13875.0	0.040092	9.436279	-22.147314	-6.96
sample_index	13875.0	20058.394739	11400.211047	0.000000	1022
n_tokens_title	13875.0	-0.002783	0.997158	-3.972899	-0.66
num_keywords	13875.0	-0.011052	0.997482	-3.260042	-0.64
data_channel_is_lifestyle	13875.0	0.050739	0.219472	0.000000	0.00
data_channel_is_entertainment	13875.0	0.178739	0.383147	0.000000	0.00
data_channel_is_socmed	13875.0	0.057658	0.233103	0.000000	0.00
kw_avg_max	13875.0	0.021806	0.998054	-1.911764	-0.62
is_weekend	13875.0	0.130450	0.336810	0.000000	0.00
global_subjectivity	13875.0	0.002958	0.988828	-3.799778	-0.40
global_rate_positive_words	13875.0	0.011095	0.999152	-2.273573	-0.64
rate_positive_words	13875.0	0.013677	0.989198	-3.586415	-0.43
max_positive_polarity	13875.0	0.011449	0.992793	-3.053998	-0.63
min_negative_polarity	13875.0	0.003018	0.997388	-1.646847	-0.61
max_negative_polarity	13875.0	-0.000837	1.009672	-9.358111	-0.18
title_sentiment_polarity	13875.0	0.004485	1.003665	-4.036308	-0.26
abs_title_subjectivity	13875.0	0.003747	0.998231	-1.810719	-0.92
ln_n_tokens_content	13875.0	0.009432	0.985424	-4.691612	-0.29
ln_num_hrefs	13875.0	0.012760	0.999294	-2.664283	-0.67
ln_num_imgs	13875.0	0.007444	1.006700	-1.146531	-0.43
ln_num_videos	13875.0	0.012156	1.013900	-0.588439	-0.58
ln_kw_min_min	13875.0	-0.020741	0.984775	-0.677671	-0.67
ln_kw_avg_min	13875.0	-0.001939	1.003748	-4.682076	-0.29
ln_kw_min_max	13875.0	0.007746	0.998410	-1.115960	-1.11
ln_kw_avg_avg	13875.0	0.006153	1.001031	-16.296139	-0.40
ln_self_reference_avg_sharess	13875.0	0.016240	0.992200	-2.032745	0.07
ln_LDA_00	13875.0	0.002041	1.002881	-0.671547	-0.63
ln_LDA_01	13875.0	0.000617	0.999165	-0.600268	-0.55
ln_LDA_02	13875.0	-0.004631	0.995019	-0.745912	-0.69
ln_LDA_03	13875.0	0.003400	0.999146	-0.734874	-0.68
ln_LDA_04	13875.0	0.000682	0.996444	-0.792825	-0.74
ln_global_rate_negative_words	13875.0	-0.004984	0.998803	-1.553284	-0.64
ln_min_positive_polarity	13875.0	-0.010174	0.982560	-1.481184	-0.67
ln_abs_title_sentiment_polarity	13875.0	0.000694	1.004252	-0.740381	-0.74

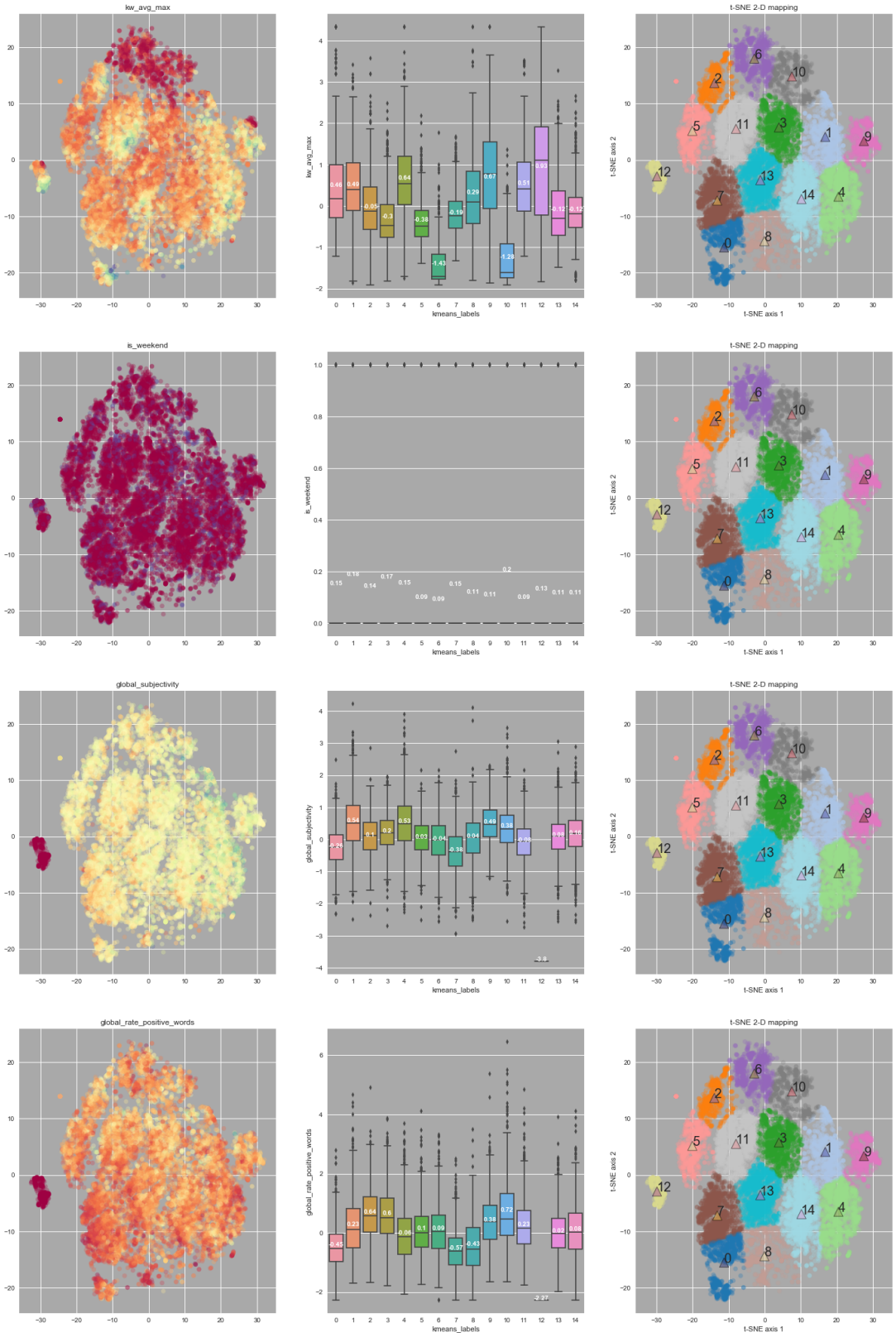
In [73]:
```
col_names = X_all_together.columns.values.tolist()
```

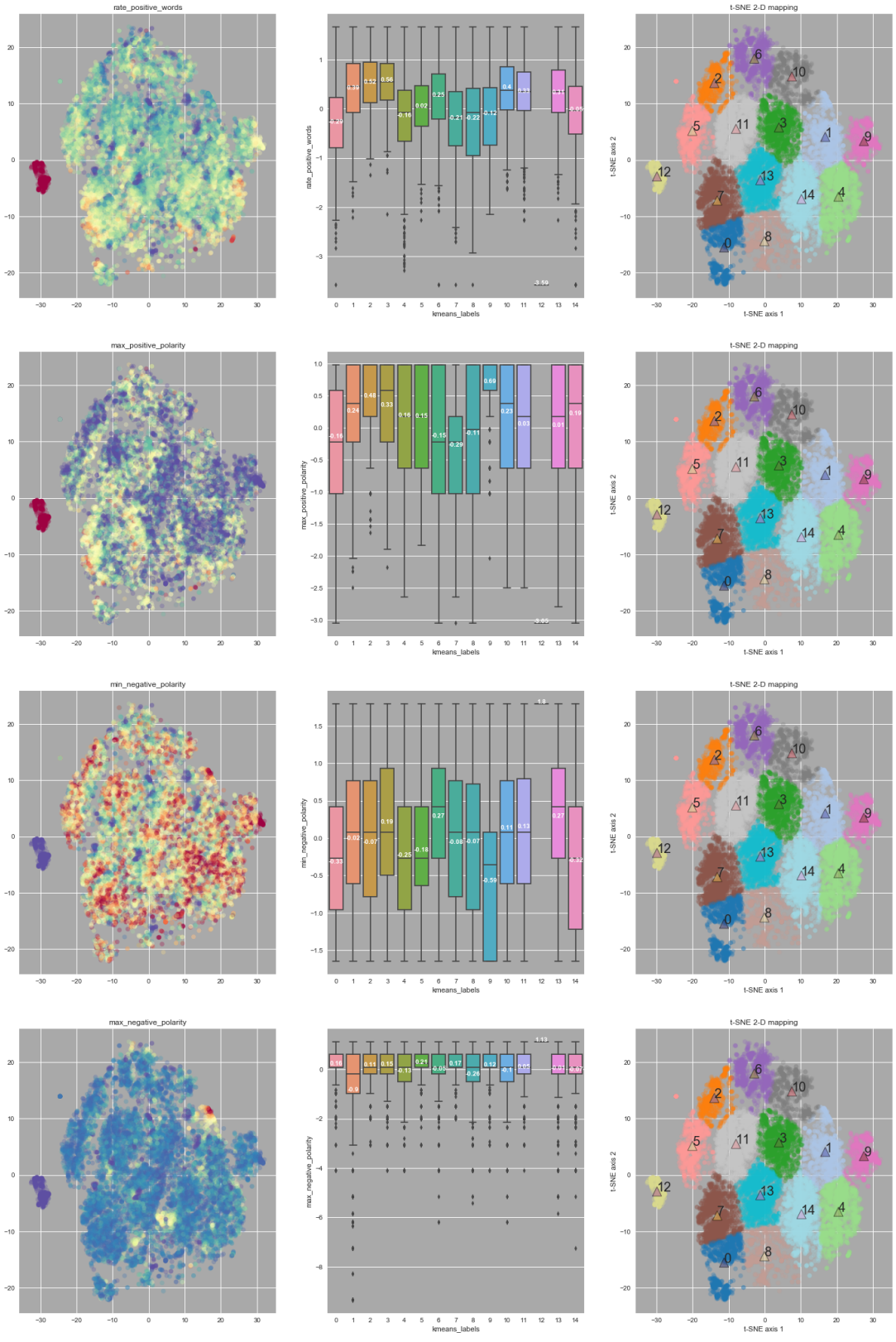
In [75]:

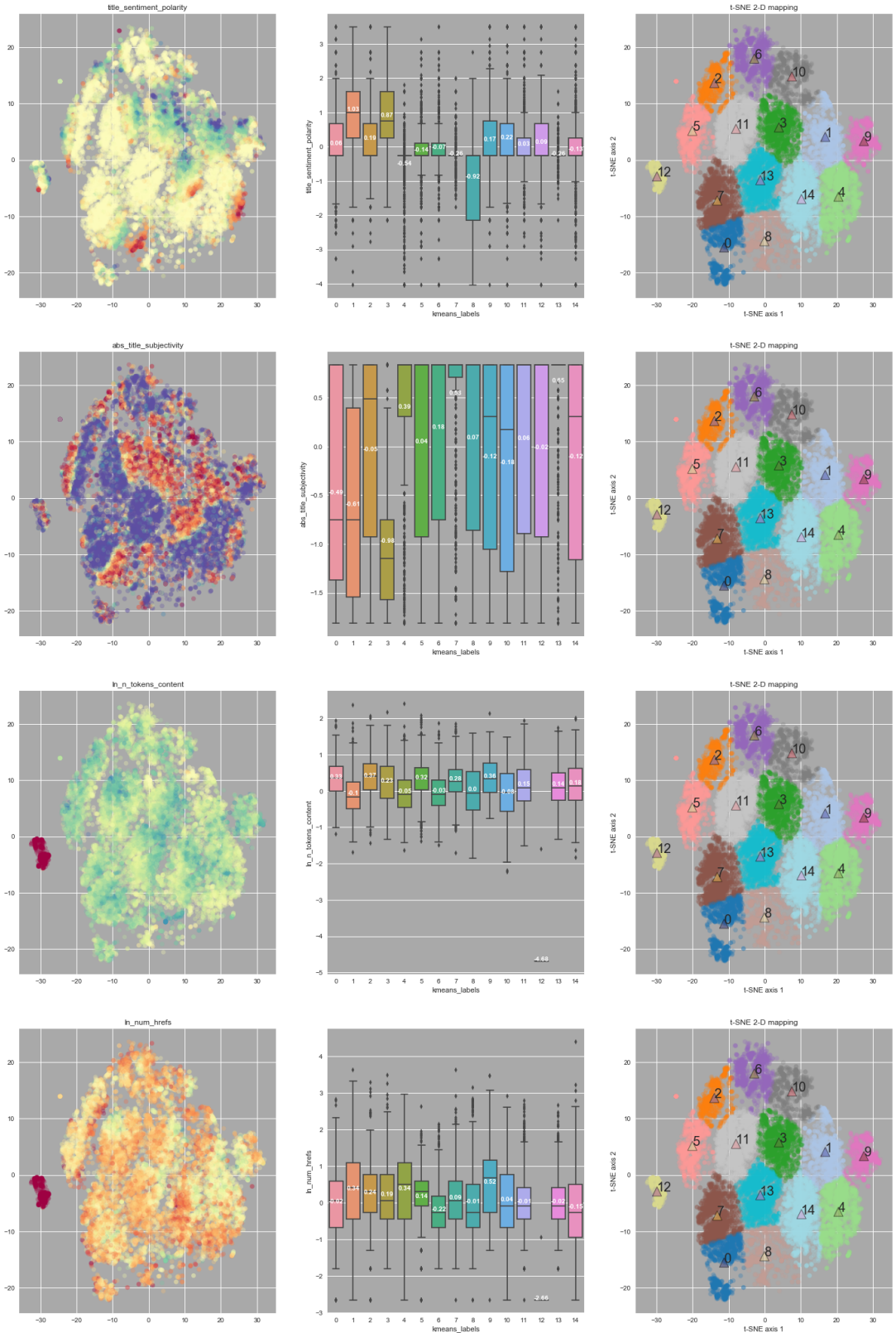
```
# boxplot across clusters for each feature ...
import seaborn as sns

col_names = X_all_together.columns.values.tolist()

for col in col_names :

    _ = plt.figure(figsize=(24, 8));

# ... feature distribution color map

    _ = plt.subplot(131, facecolor = 'darkgrey');

    _ = plt.scatter(X_all_together['x-tsne'], X_all_together['y-tsne'],
                    c = X_all_together[col],
                    cmap = plt.cm.Spectral,
                    s = 50,
                    linewidths = 0,
                    alpha = 0.30)
    _ = plt.title(col)

# ... feature boxplots

    _ = plt.subplot(132, facecolor = 'darkgrey');
    ax = sns.boxplot(x = "kmeans_labels", y = col, data = X_all_together);


    average_values = X_all_together.groupby(['kmeans_labels'])[col].mean().values
    average_labels = [str(np.round(s, 2)) for s in average_values]

    pos = range(len(average_values))
    for tick, label in zip(pos, ax.get_xticklabels()):

        _ = ax.text(pos[tick], average_values[tick], average_labels[tick],
                horizontalalignment = 'center', size = 'small', color = 'w', weight = 'semibol
d')

# ... cluster color map

    _ = plt.subplot(133, facecolor = 'darkgrey');

    _ = plt.scatter(X_all_together['x-tsne'], X_all_together['y-tsne'],
                    c = kmeans_labels,
                    cmap = plt.cm.tab20,
                    s = 50,
                    linewidths = 0,
                    alpha = 0.30)
    _ = plt.scatter(kmeans_centers[:, 0], kmeans_centers[:, 1],
                c = range(n_lda),
                cmap = plt.cm.tab20b,
                s = 200,
                linewidths = 1.0,
                marker = '^',
                edgecolors = 'black',
                alpha = 0.50);
```

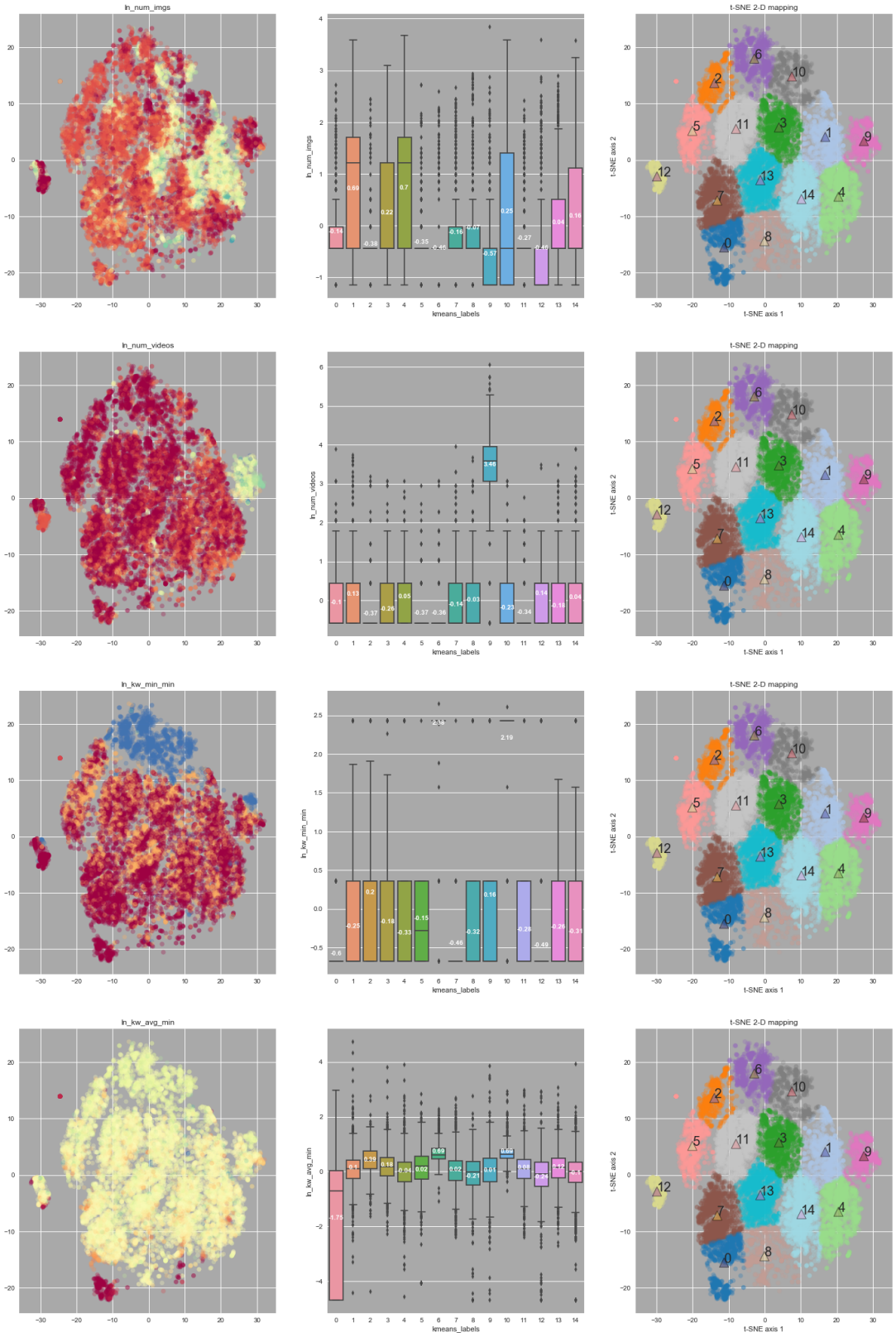

x-tsne
x-tsne
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
y-tsne
y-tsne
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
sample_index
sample_index
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
n_tokens_title
n_tokens_title
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2

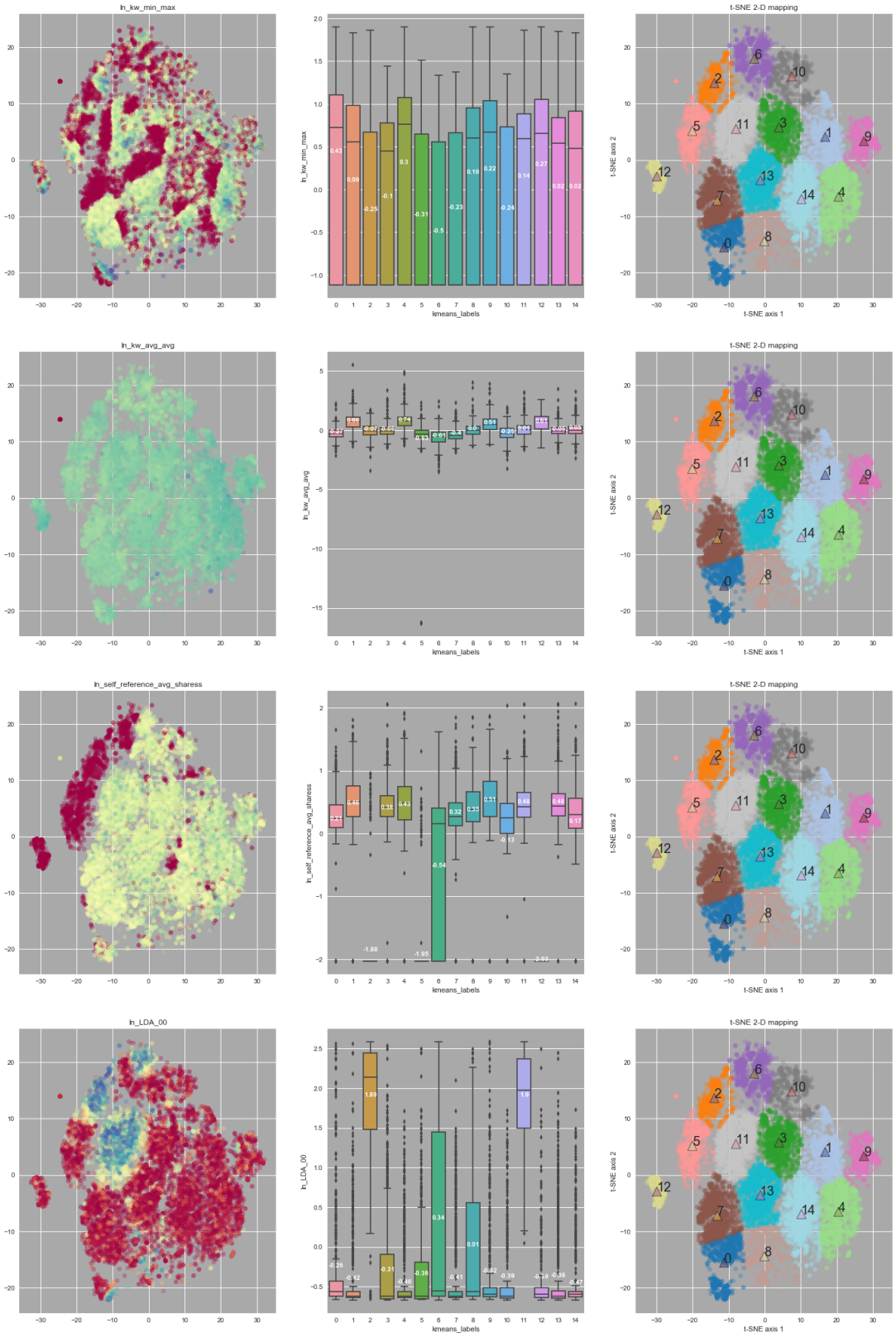

num_keywords
num_keywords
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
data_channel_is_lifestyle
data_channel_is_lifestyle
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
data_channel_is_entertainment
data_channel_is_entertainment
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
data_channel_is_socmed
data_channel_is_socmed
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2

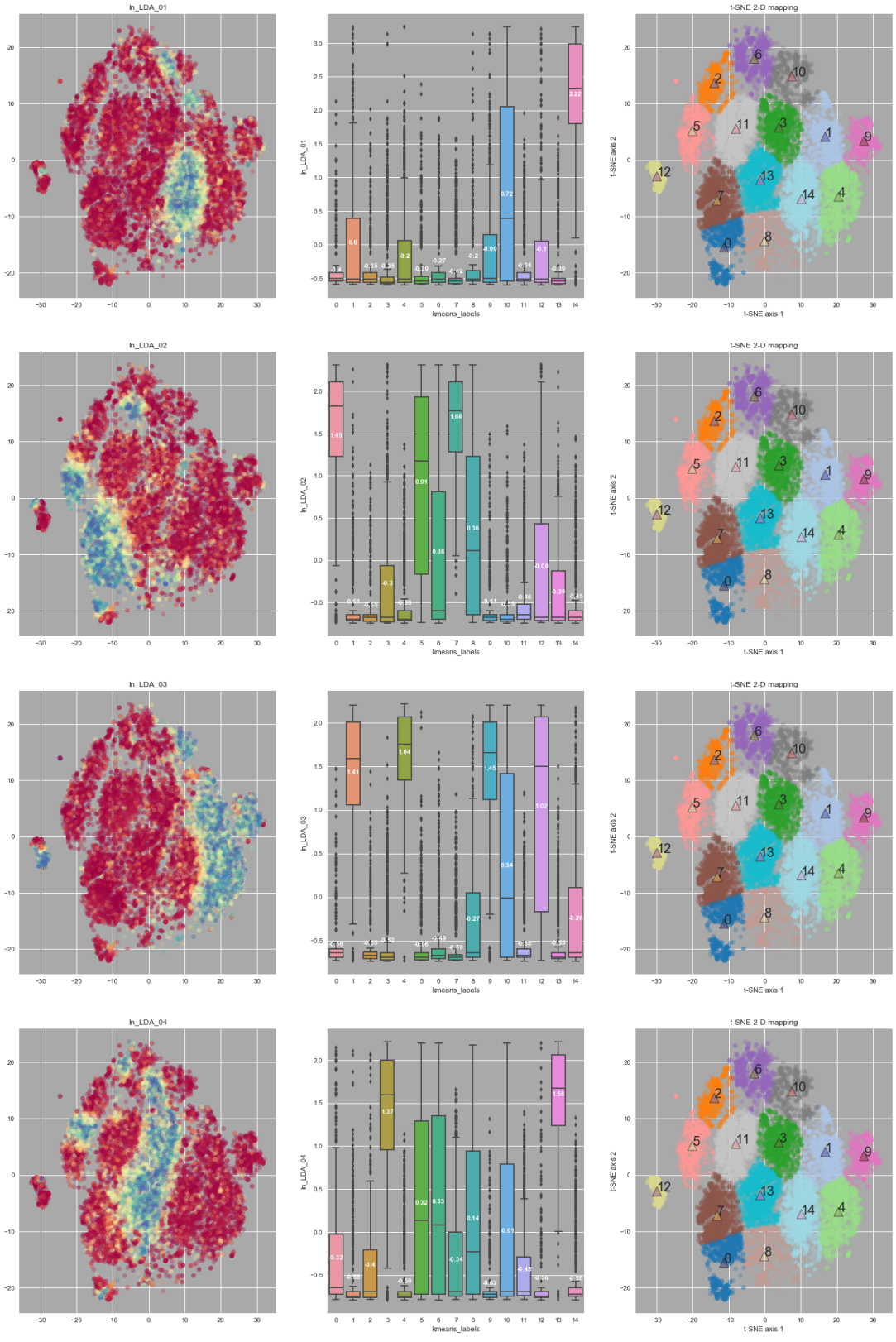

kw_avg_max
kw_avg_max
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 2
t-SNE axis 1
is_weekend
is_weekend
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 2
t-SNE axis 1
global_subjectivity
global_subjectivity
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 2
t-SNE axis 1
global_rate_positive_words
global_rate_positive_words
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 2
t-SNE axis 1

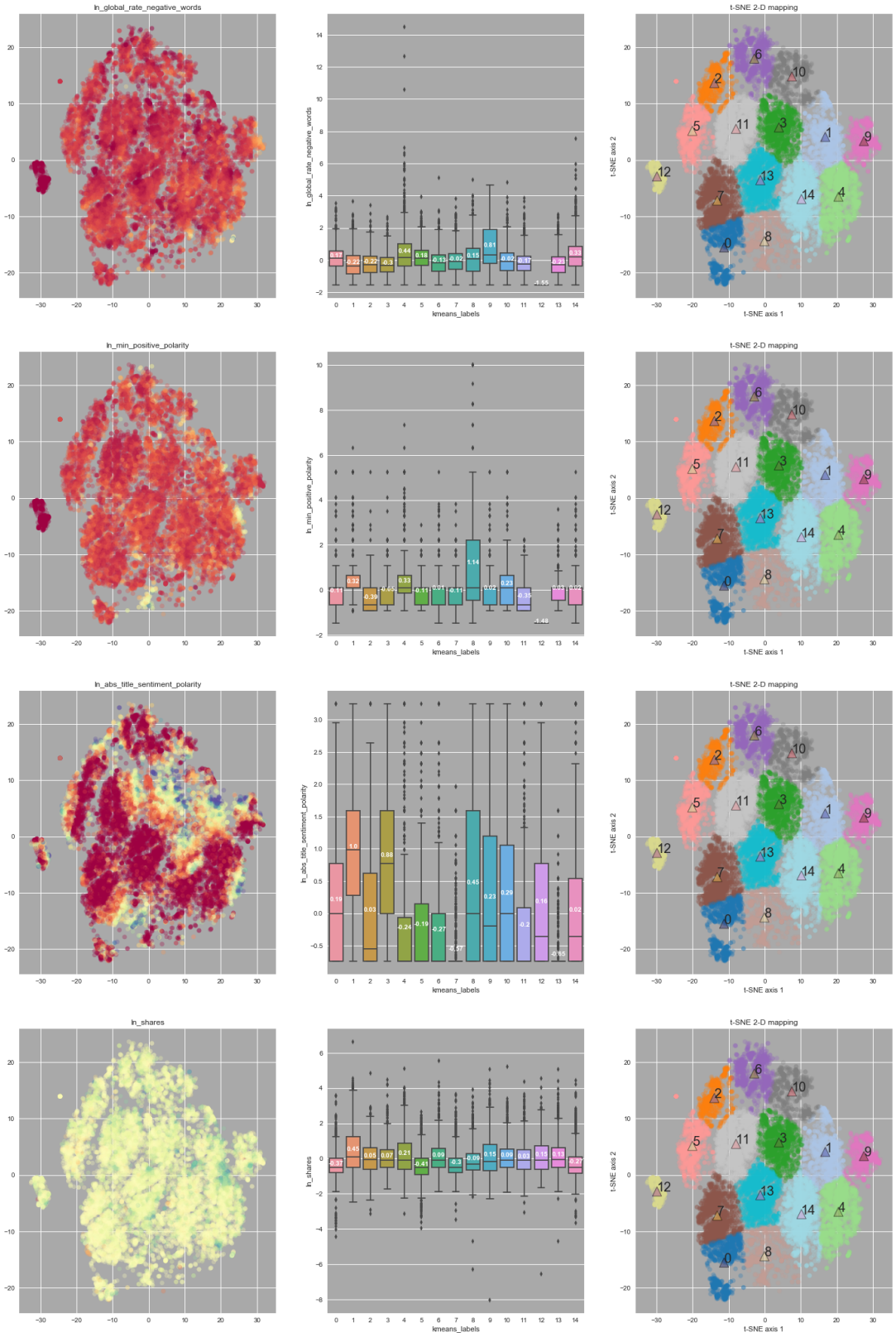

rate_positive_words
rate_positive_words
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
max_positive_polarity
max_positive_polarity
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
min_negative_polarity
min_negative_polarity
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
max_negative_polarity
max_negative_polarity
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2


title_sentiment_polarity
t-SNE 2-D mapping
abs_title_subjectivity
t-SNE 2-D mapping
ln_n_tokens_content
t-SNE 2-D mapping
ln_num_hrefs
t-SNE 2-D mapping
kmeans_labels
t-SNE axis 1
t-SNE axis 2


ln_num_imgs
ln_num_imgs
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
ln_num_videos
ln_num_videos
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
ln_kw_min_min
ln_kw_min_min
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
ln_kw_avg_min
ln_kw_avg_min
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2


ln_kw_min_max
ln_kw_min_max
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 2
t-SNE axis 1
ln_kw_avg_avg
ln_kw_avg_avg
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 2
t-SNE axis 1
ln_self_reference_avg_sharess
ln_self_reference_avg_sharess
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 2
t-SNE axis 1
ln_LDA_00
ln_LDA_00
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 2
t-SNE axis 1


ln_LDA_01
ln_LDA_01
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
ln_LDA_02
ln_LDA_02
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
ln_LDA_03
ln_LDA_03
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
ln_LDA_04
ln_LDA_04
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2


ln_global_rate_negative_words
ln_global_rate_negative_words
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
ln_min_positive_polarity
ln_min_positive_polarity
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
ln_abs_title_sentiment_polarity
ln_abs_title_sentiment_polarity
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2
ln_shares
ln_shares
kmeans_labels
t-SNE 2-D mapping
t-SNE axis 1
t-SNE axis 2

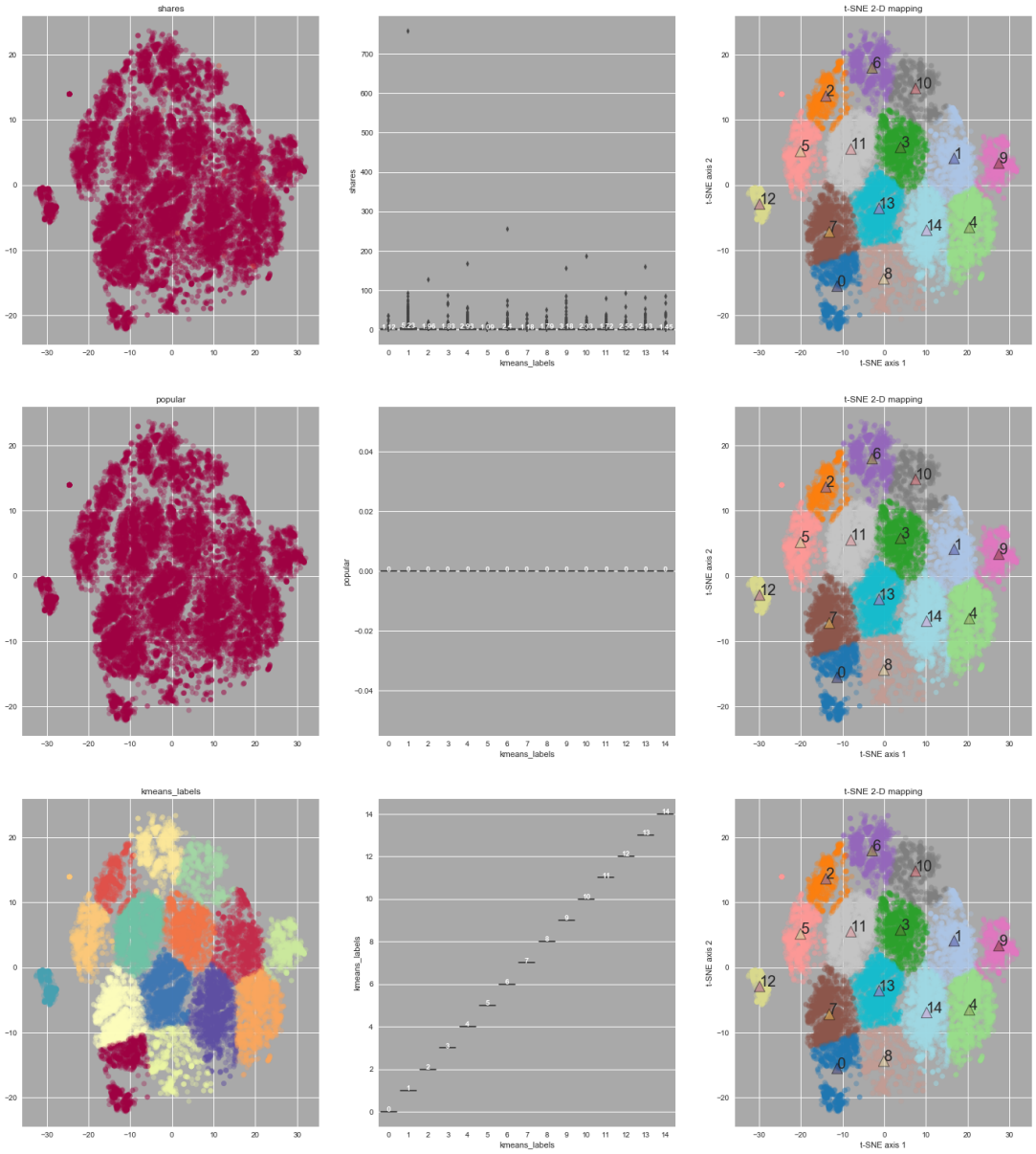


Table of Contents

end of file

In [ ]: