Amazon tf-idf

December 26, 2018

```
In [1]: %matplotlib inline

        import sqlite3
        import pandas as pd
        import numpy as np
        import nltk
        import string
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.feature_extraction.text import TfidfTransformer
        from sklearn.feature_extraction.text import TfidfVectorizer

        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.metrics import confusion_matrix
        from sklearn import metrics
        from sklearn.metrics import roc_curve, auc
        from nltk.stem.porter import PorterStemmer


        con = sqlite3.connect('./final.sqlite')

        review= pd.read_sql_query('''select * from Reviews''',con)
```

```
In [2]: labels=review.Score[0:10000]
        labels
```

```
Out[2]: 0        positive
        1        positive
        2        positive
        3        negative
        4        positive
        5        positive
        6        positive
        7        positive
        8        positive
        9        positive
        10       positive
        11       positive
```

```
12      positive
13      positive
14      positive
15      positive
16      positive
17      positive
18      positive
19      positive
20      positive
21      positive
22      positive
23      positive
24      positive
25      positive
26      positive
27      positive
28      positive
29      positive
          ...
9952    positive
9953    negative
9954    negative
9955    positive
9956    positive
9957    positive
9958    positive
9959    positive
9960    positive
9961    positive
9962    negative
9963    negative
9964    positive
9965    positive
9966    positive
9967    negative
9968    positive
9969    positive
9970    positive
9971    positive
9972    positive
9973    positive
9974    positive
9975    negative
9976    positive
9977    positive
9978    positive
9979    positive
9980    positive
```

```
        9981    positive
        Name: Score, Length: 9982, dtype: object
```

```
In [3]: tf_idf_vect = TfidfVectorizer(ngram_range=(1,2))
        final_tf_idf = tf_idf_vect.fit_transform(review['Text'].values)
```

```
In [4]: final_tf_idf.get_shape()
```

```
Out[4]: (9982, 284906)
```

```
In [5]: features = tf_idf_vect.get_feature_names()
        len(features)
```

```
Out[5]: 284906
```

```
In [6]: features[100000:100010]
```

```
Out[6]: ['gets it',
         'gets its',
         'gets just',
         'gets led',
         'gets lot',
         'gets lots',
         'gets low',
         'gets made',
         'gets me',
         'gets minus']
```

```
In [7]: # covnert a row in saprsematrix to a numpy array
        print(final_tf_idf[3,:].toarray()[0])
```

```
[0. 0. 0. ... 0. 0. 0.]
```

```
In [8]: # source: https://buhrmann.github.io/tfidf-analysis.html
        def top_tfidf_feats(row, features, top_n=25):
            ''' Get top n tfidf values in row and return them with their corresponding feature
            topn_ids = np.argsort(row)[::-1][:top_n]
            top_feats = [(features[i], row[i]) for i in topn_ids]
            df = pd.DataFrame(top_feats)
            df.columns = ['feature', 'tfidf']
            return df

        top_tfidf = top_tfidf_feats(final_tf_idf[1,:].toarray()[0],features,25)
```

```
In [9]: top_tfidf
```

```
Out[9]:                  feature     tfidf
        0                   book  0.245512
        1               the book  0.157524
```

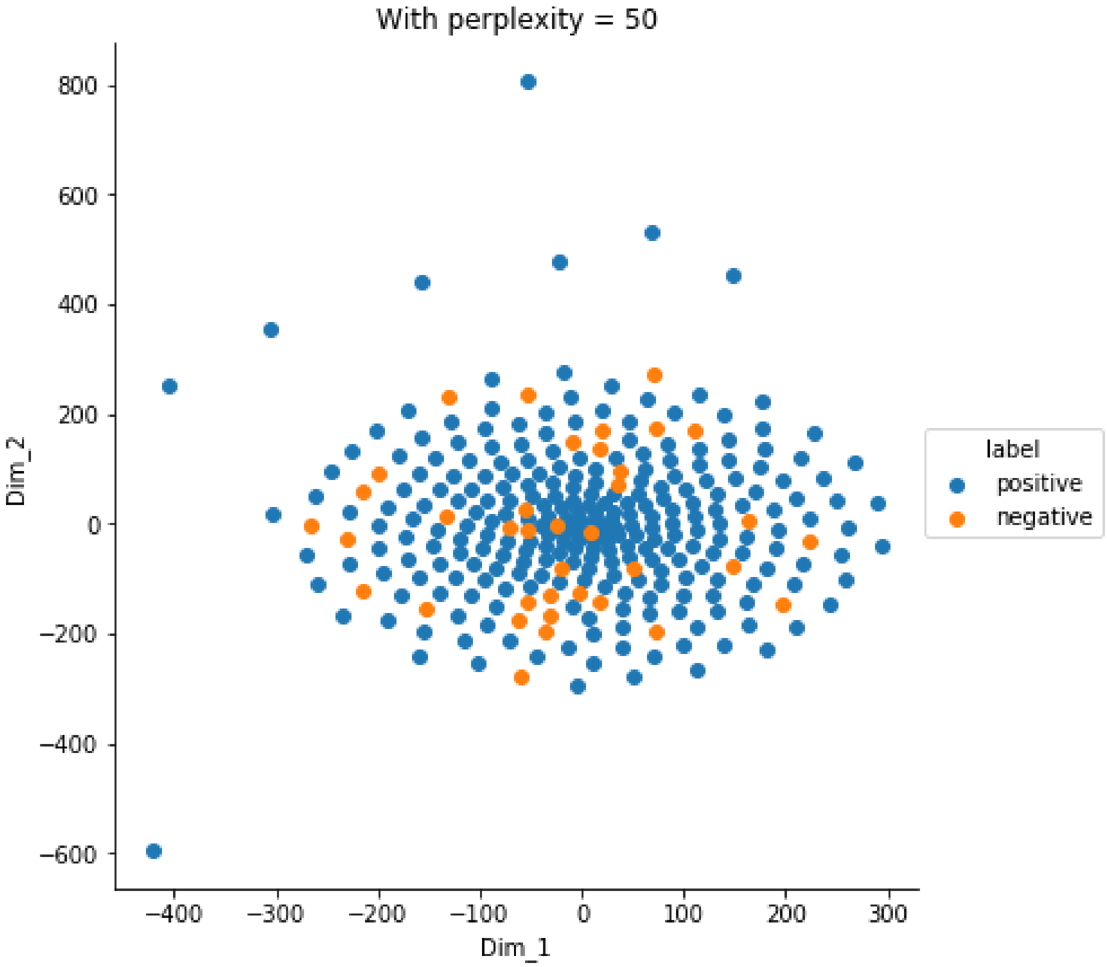
```
2                  month  0.148871
3              this book  0.146111
4                  going  0.129866
5               that she  0.125391
6                    she  0.114254
7     independently yet  0.097033
8                 grader  0.097033
9            sized books  0.097033
10     girls especially  0.097033
11            proud that  0.097033
12            going once  0.097033
13           twice going  0.097033
14            rhymes she  0.097033
15           the rhymes  0.097033
16         independently  0.097033
17      childhood highly  0.097033
18         singing going  0.097033
19           and singing  0.097033
20             month her  0.097033
21             she reads  0.097033
22  reads independently  0.097033
23          first grader  0.097033
24         among regular  0.097033
```

```
In [10]: from sklearn.preprocessing import StandardScaler
         standardized_data = StandardScaler(with_mean=False).fit_transform(final_tf_idf)
         print(standardized_data.shape)
```

```
(9982, 284906)
```

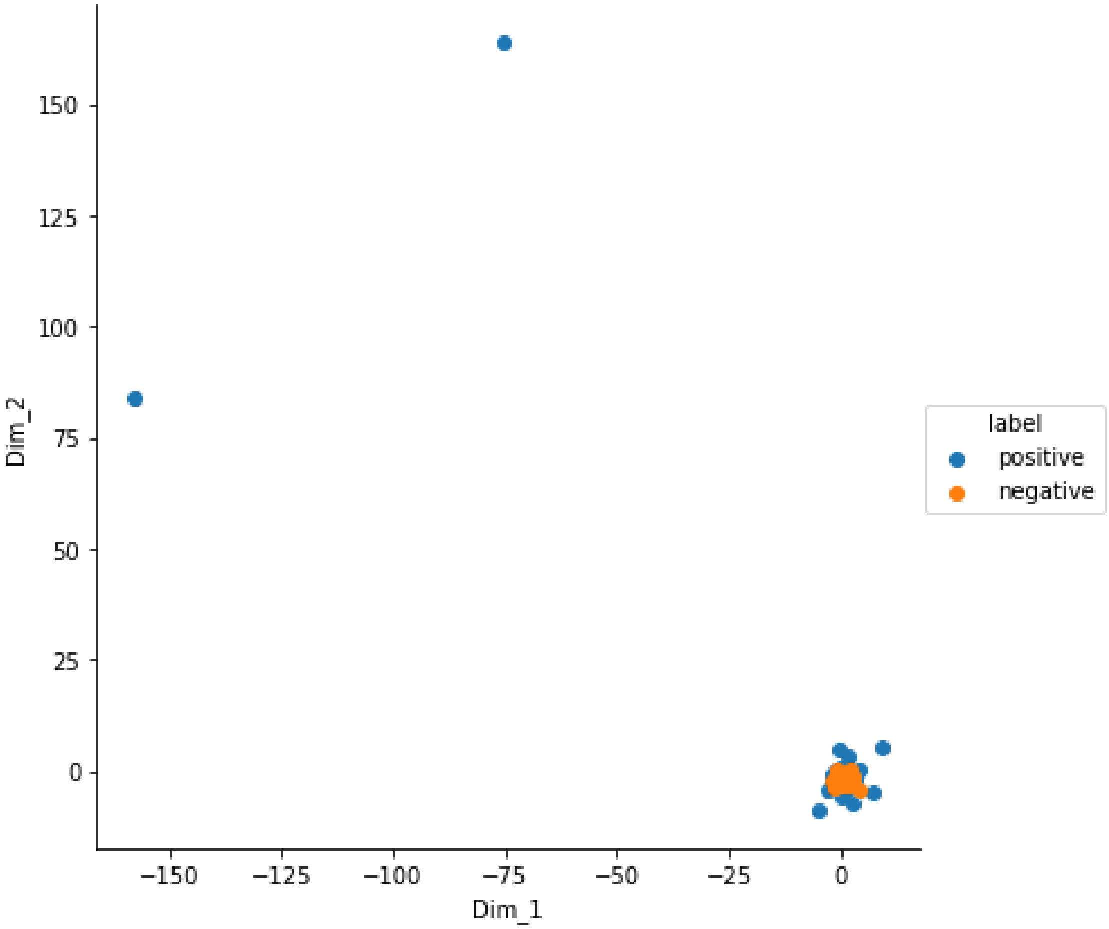
```
In [12]: # TSNE

         from sklearn.manifold import TSNE

         # Picking the top 1000 points as TSNE takes a lot of time for 15K points
         data_1000 = standardized_data[0:300].todense()
         labels_1000 = labels[0:300]

         model = TSNE(n_components=2, random_state=0,perplexity=30)
         tsne_data = model.fit_transform(data_1000)

         # creating a new data fram which help us in ploting the result data
         tsne_data = np.vstack((tsne_data.T, labels_1000)).T
         tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

         # Ploting the result of tsne
         sns.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_leg
```

```
plt.title('With perplexity = 50')
plt.show()
```

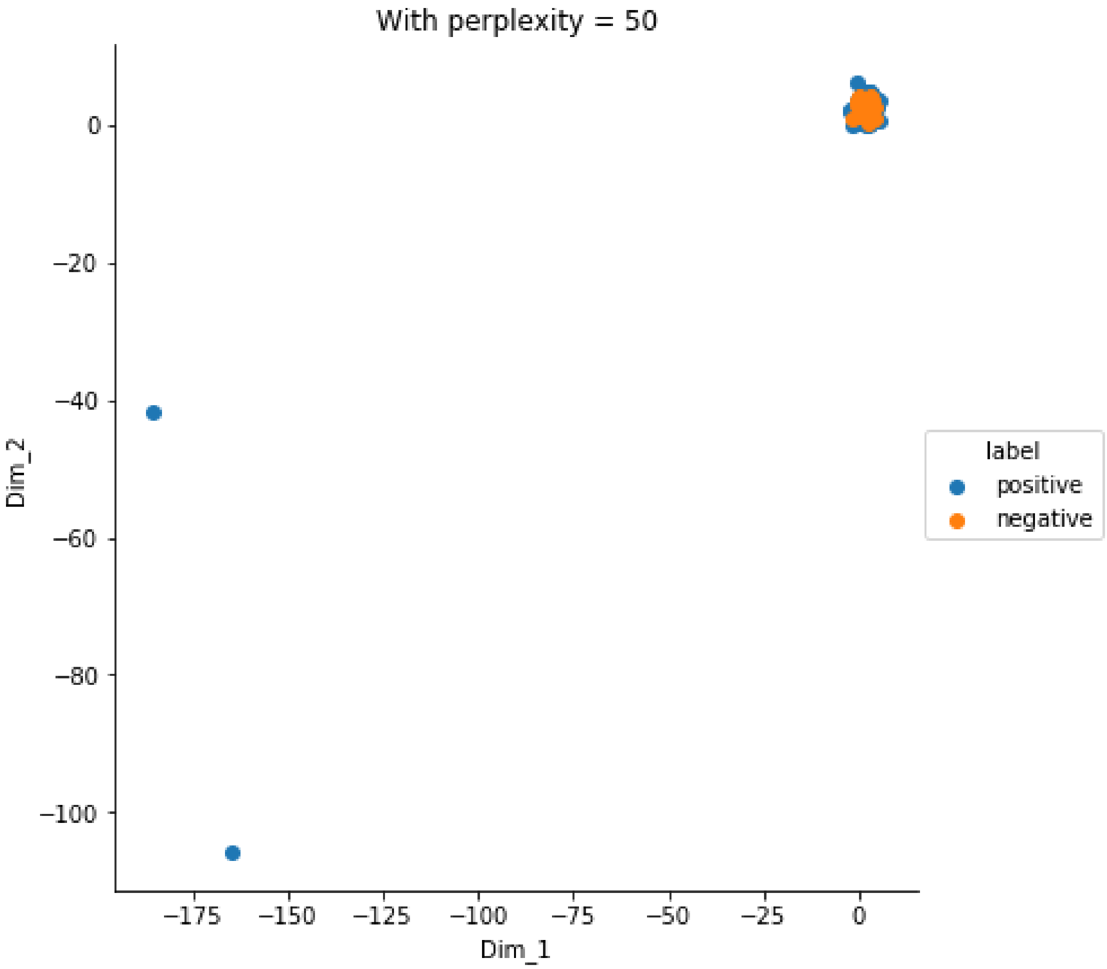


```
In [11]: model = TSNE(n_components=2, random_state=0,perplexity=50)
         # configuring the parameteres
         # the number of components = 2
         # default perplexity = 30
         # default learning rate = 200
         # default Maximum number of iterations for the optimization = 1000

         tsne_data = model.fit_transform(data_1000)


         # creating a new data frame which help us in ploting the result data
         tsne_data = np.vstack((tsne_data.T, labels_1000)).T
         tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))
```

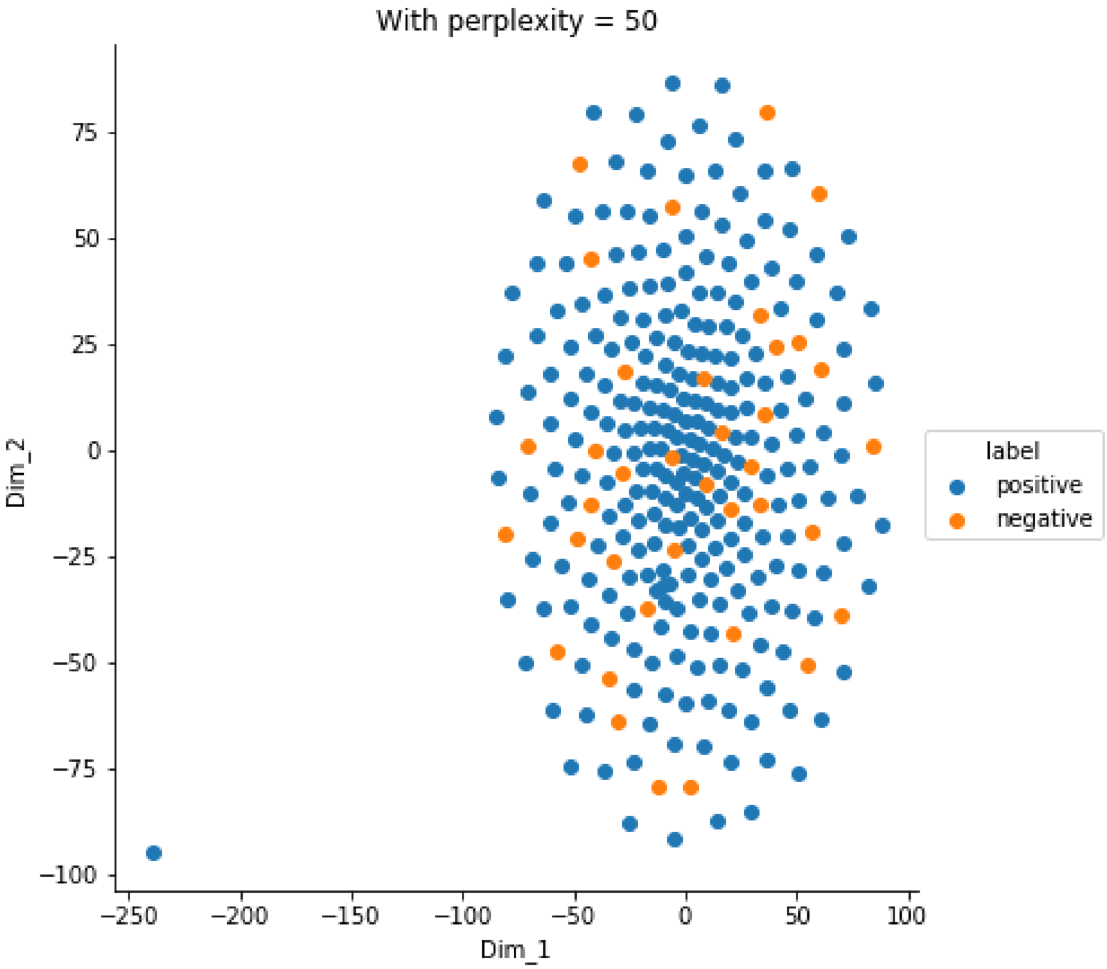
```
# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_leg
plt.show()
```



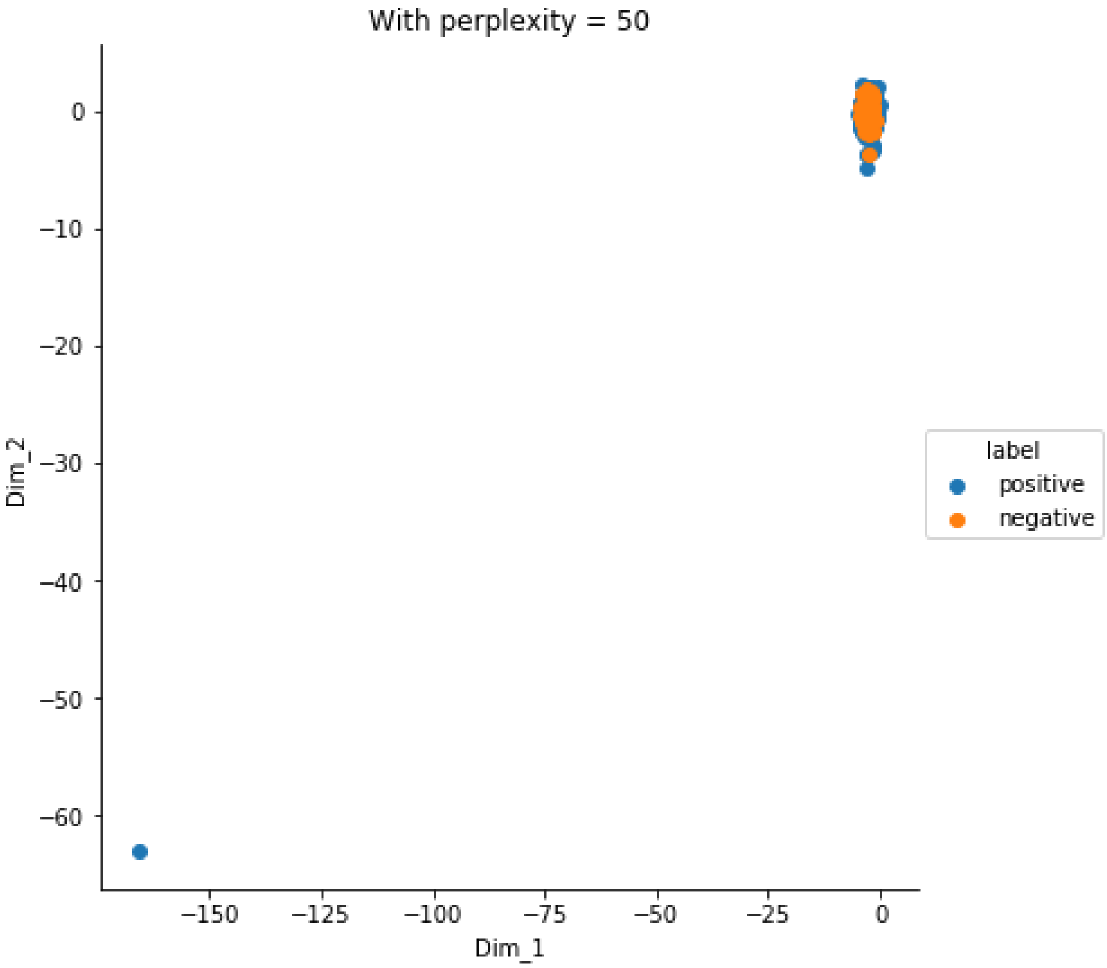
```
In [14]: model = TSNE(n_components=2, random_state=0,perplexity=60)
         tsne_data = model.fit_transform(data_1000)

         # creating a new data fram which help us in ploting the result data
         tsne_data = np.vstack((tsne_data.T, labels_1000)).T
         tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

         # Ploting the result of tsne
         sns.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_leg
         plt.title('With perplexity = 50')
         plt.show()
```



```
In [15]: model = TSNE(n_components=2, random_state=0,perplexity=90)
         tsne_data = model.fit_transform(data_1000)

         # creating a new data fram which help us in ploting the result data
         tsne_data = np.vstack((tsne_data.T, labels_1000)).T
         tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

         # Ploting the result of tsne
         sns.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_leg
         plt.title('With perplexity = 50')
         plt.show()
```



```
In [18]: model = TSNE(n_components=2, random_state=0,perplexity=95)
         tsne_data = model.fit_transform(data_1000)

         # creating a new data fram which help us in ploting the result data
         tsne_data = np.vstack((tsne_data.T, labels_1000)).T
         tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

         # Ploting the result of tsne
         sns.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_leg
         plt.title('With perplexity = 50')
         plt.show()
```


With perplexity = 50
Dim_2
0
−10
−20
−30
−40
−50
−60
Dim_1
−150
−125
−100
−75
−50
−25
0
label
● positive
● negative