Amazon Reviews Viz

August 18, 2018

1 t- SNE for Amazon food reviews

This is a randomly sampled dataset consisting of positive and negative points of 10K points each. This dataset is preprocessed and has 20K Rows of data.

"""df1= final[final.Score == 'positive'] print(df1.shape) df2= final[final.Score == 'negative'] sampled_df1 = df1.sample(10000) sampled_df2 = df2.sample(10000) final = pd.concat([sampled_df1,sampled_df2]) print (final.Score.value_counts())"""

1.1 Visualising Reviews with t-SNE

The vectors generated from the various methods like BoW, TF - IDF, Avg W2V and TF-IDF W2V will not produce a good result. Since the features are not related to each other.

Eg. I have a ball. 4 Features Hi how are you. 4 Features

These features are not correlated to each other and hence they do not produce a good visualisation result.

1.2 Loading the dataset

```
In [8]: import pandas as pd
        import sqlite3

        con = sqlite3.connect('sampled.sqlite')

        final = pd.read_sql_query("""
        SELECT *
        FROM final
        """, con)
```

1.3 BoW(Bag of Words)

1.3.1 Bi-Gram

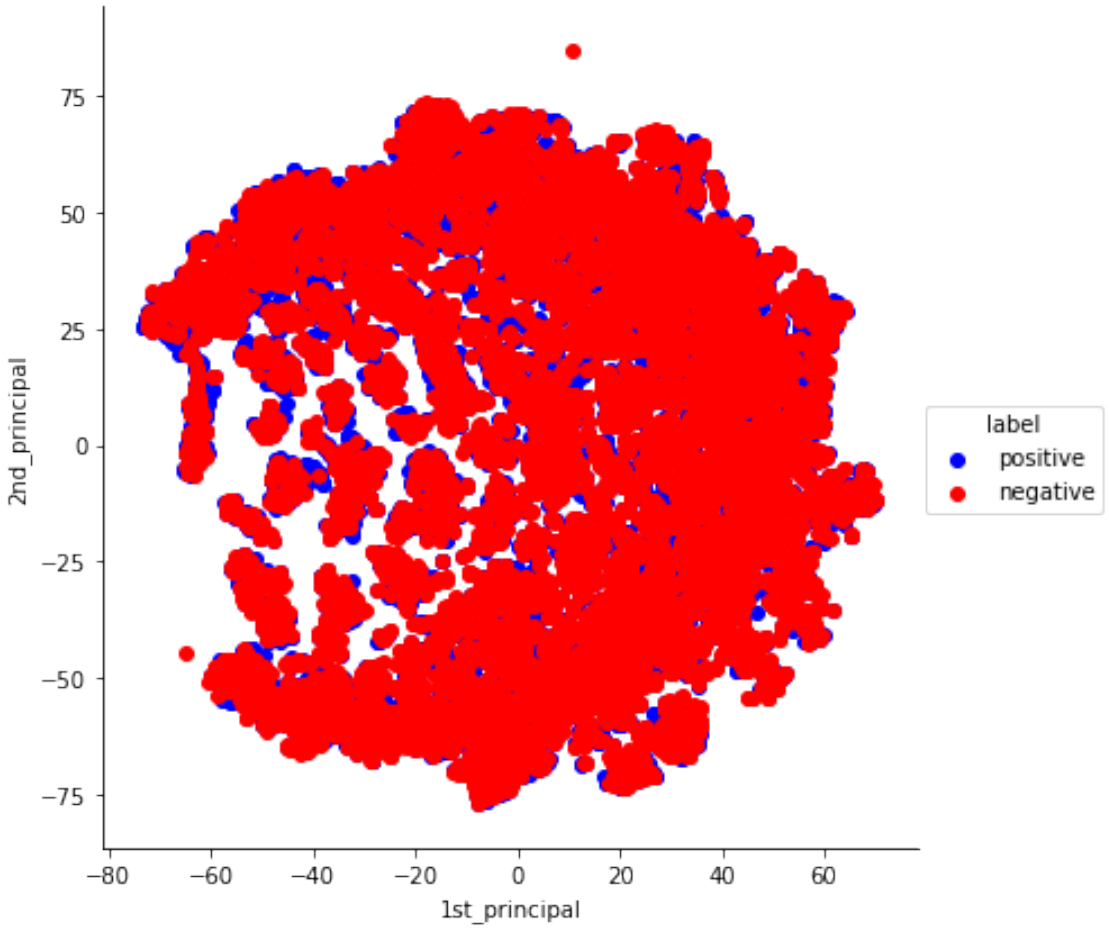
```
In [9]: from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.manifold import TSNE
        from sklearn.decomposition import TruncatedSVD
        from sklearn.preprocessing import StandardScaler
        import seaborn as sn
        import matplotlib.pyplot as plt
```

```
        import numpy as np
        #bi-gram, tri-gram and n-gram

        count_vect = CountVectorizer(ngram_range=(1,2))
        final_bigram_counts = count_vect.fit_transform(final['Text'].values)
        standardized_data = StandardScaler(with_mean = False).fit_transform(final_bigram_counts)
        labels = final.Score.values
```

```
/usr/local/anaconda/envs/py36/lib/python3.6/site-packages/sklearn/utils/validation.py:475: DataC
  warnings.warn(msg, DataConversionWarning)
```

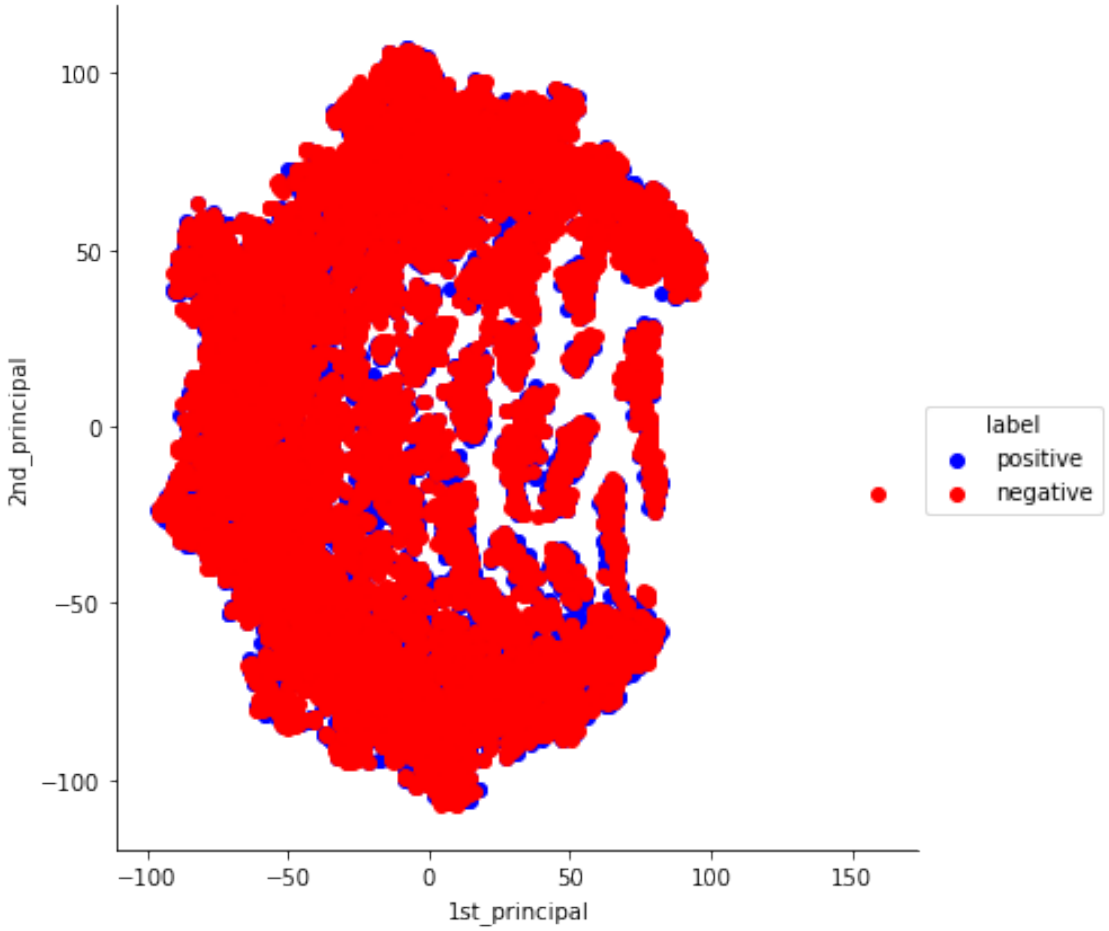
Perplexity 30 & n_iter = 2000

```
In [4]: tSVD = TruncatedSVD(n_components=10, random_state=0).fit_transform(standardized_data)
        viz = TSNE(n_components=2,perplexity = 30,n_iter = 2000).fit_transform(tSVD)
        import seaborn as sn
        # attaching the label for each 2-d data point
        viz_data = np.vstack((viz.T, labels)).T

        # creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","ne
        viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label")
        pal = dict(positive="blue", negative="red")
        sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negative
        plt.show()
```



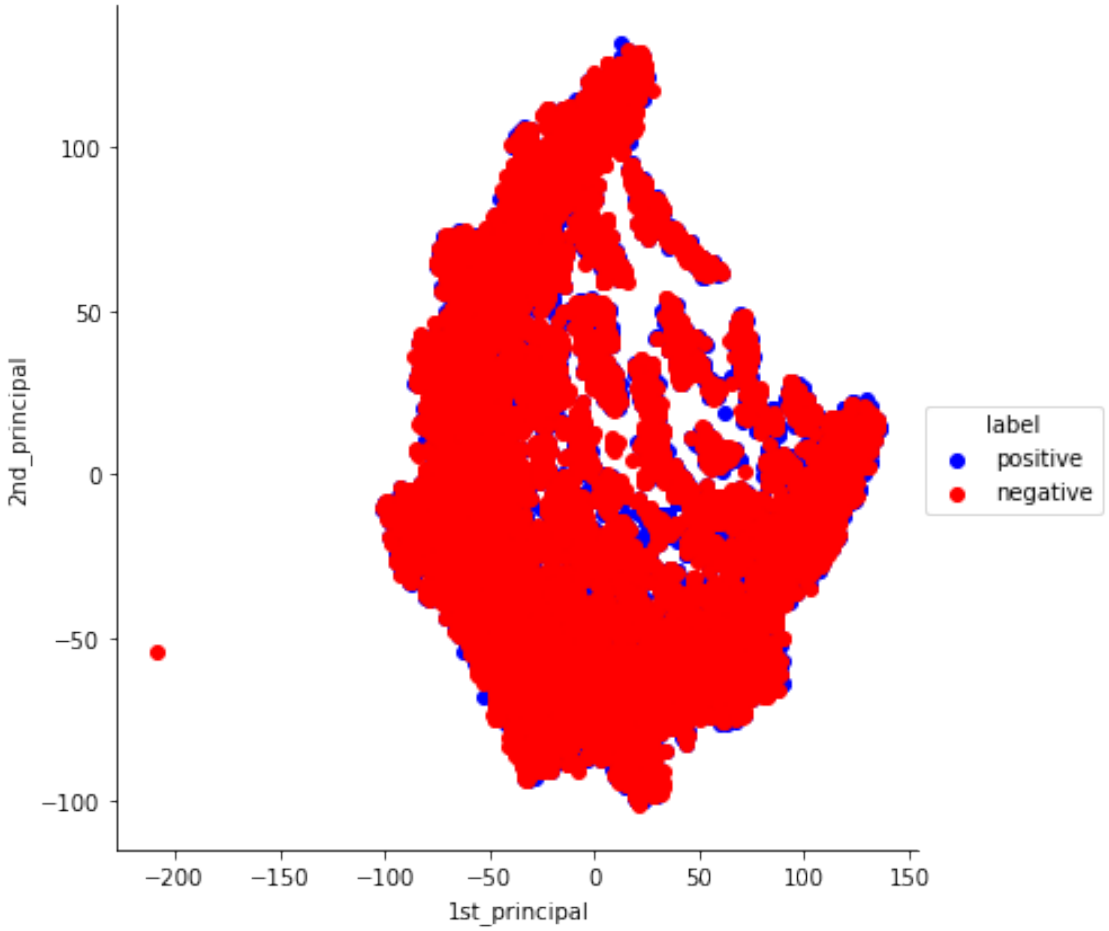
Perplexity 40 & n_iter = 3500

```
In [5]: tSVD = TruncatedSVD(n_components=10, random_state=0).fit_transform(standardized_data)
        viz = TSNE(n_components=2,perplexity = 40,n_iter = 3500).fit_transform(tSVD)

        # attaching the label for each 2-d data point
        viz_data = np.vstack((viz.T, labels)).T

        # creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","ne
        viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label")
        pal = dict(positive="blue", negative="red")
        sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negative
        plt.show()
```



Perplexity 50 & n_iter = 5000

```
In [10]: tSVD = TruncatedSVD(n_components=10, random_state=0).fit_transform(standardized_data)
         viz = TSNE(n_components=2,perplexity = 50,n_iter = 5000).fit_transform(tSVD)

         # attaching the label for each 2-d data point
         viz_data = np.vstack((viz.T, labels)).T

         # creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","n
         viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label"
         pal = dict(positive="blue", negative="red")
         sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negativ
         plt.show()
```



1.3.2 BoW conclusion

BoW did not produce a good result

There are too much of overlapping

Increasing the perplexity value and the number of iterations produced the same result with different angles of the projection of the chart

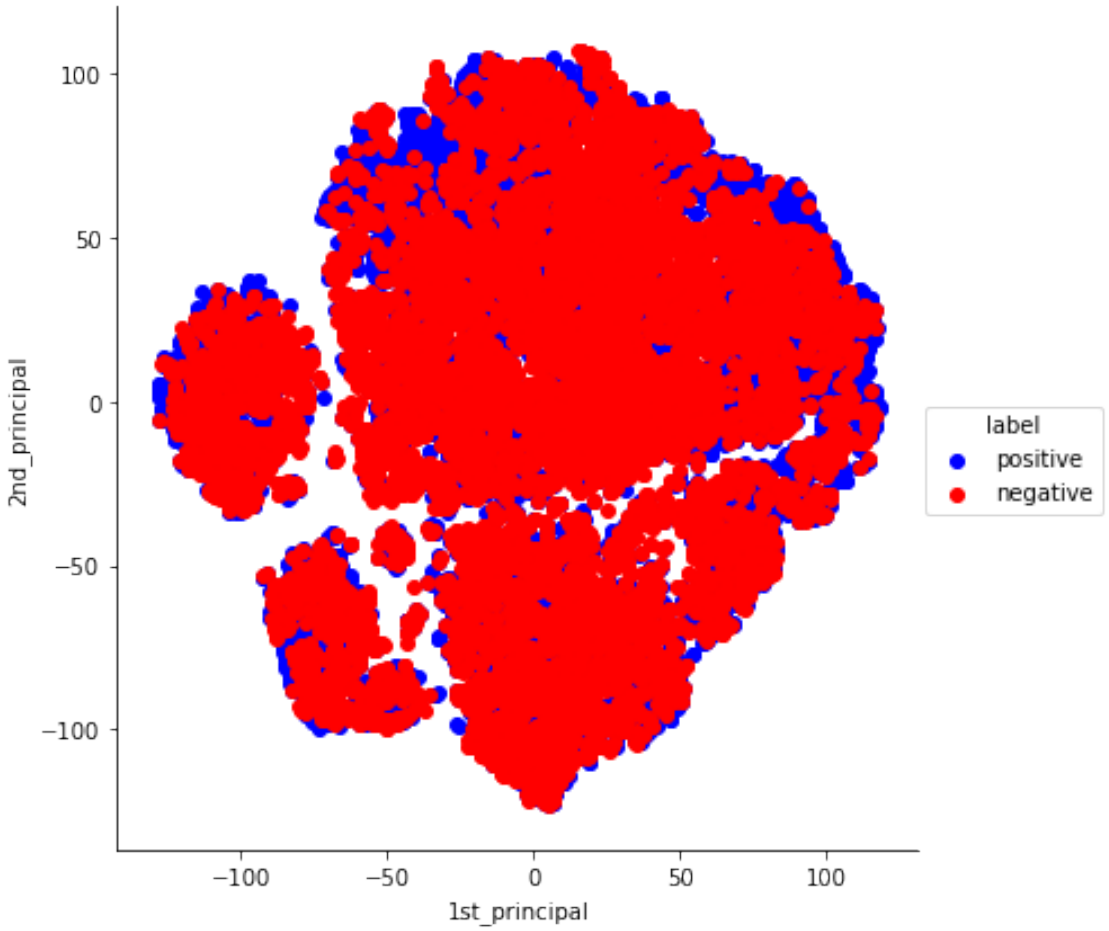
1.4 TF-IDF

```
In [11]: from sklearn.feature_extraction.text import TfidfVectorizer
         tf_idf_vect = TfidfVectorizer(ngram_range=(1,2))
         final_tf_idf = tf_idf_vect.fit_transform(final['Text'].values)
```

1.4.1 Perplexity 30 & n_iter = 2000

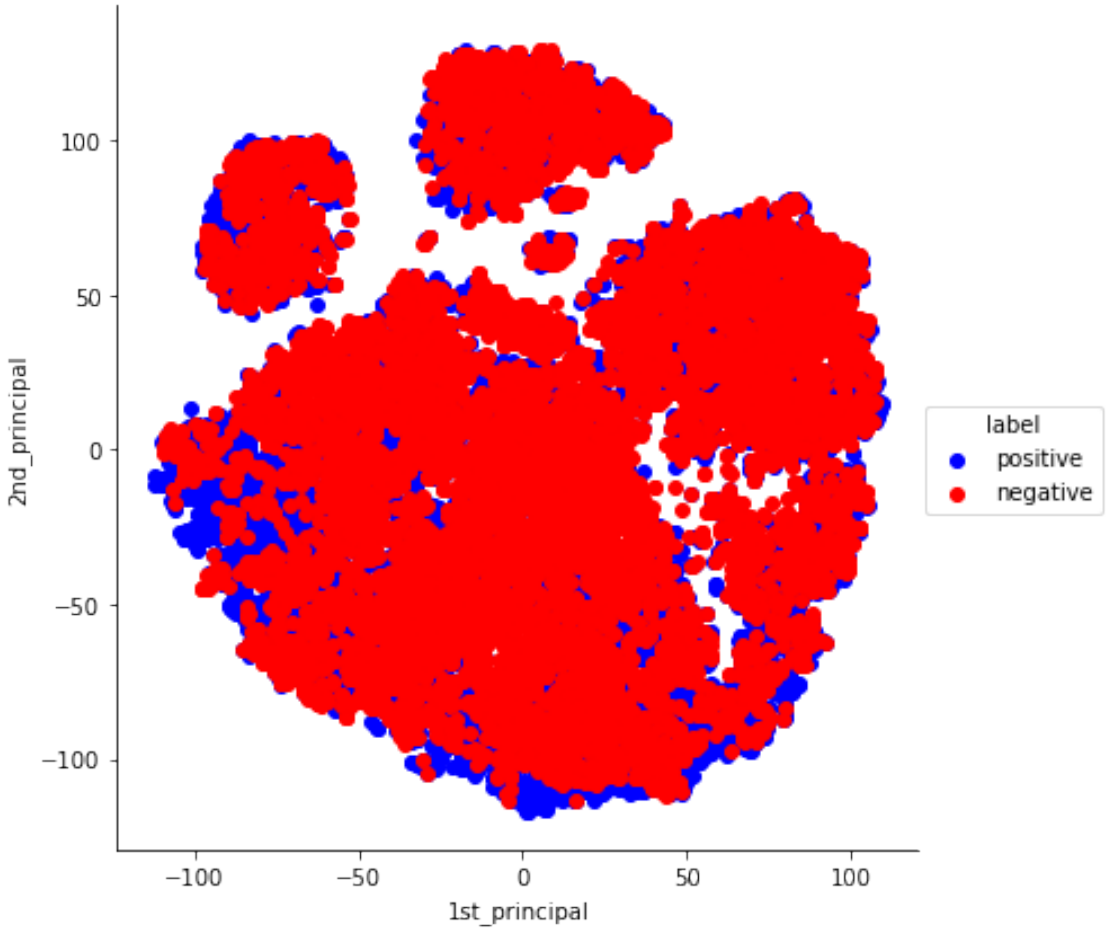
```
In [12]: tSVD = TruncatedSVD(n_components=10, random_state=0).fit_transform(final_tf_idf)
         viz = TSNE(n_components=2,perplexity = 30,n_iter = 2000).fit_transform(tSVD)
         # attaching the label for each 2-d data point
```

```
viz_data = np.vstack((viz.T, labels)).T
# creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","n
viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label"
pal = dict(positive="blue", negative="red")
sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negativ
plt.show()
```



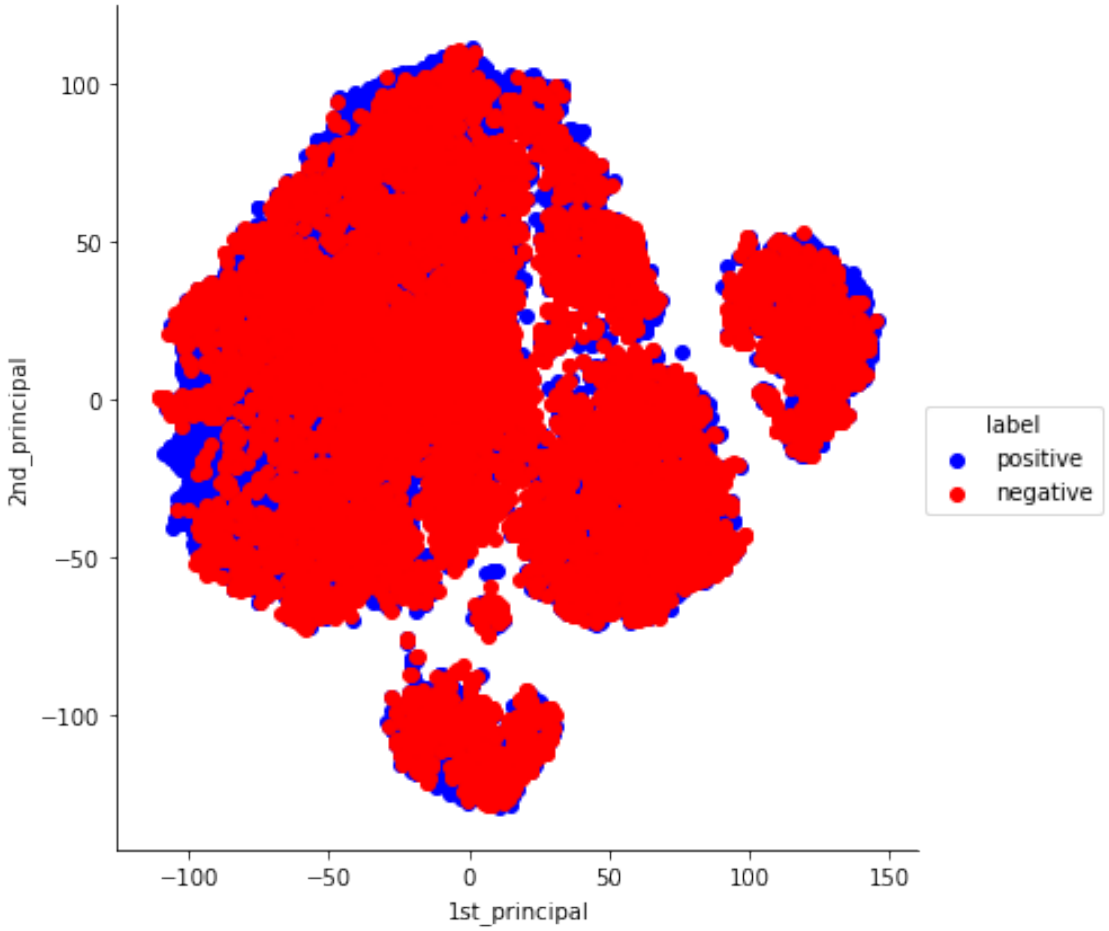
1.4.2 Perplexity 40 & n_iter = 3500

In [13]:
```
tSVD = TruncatedSVD(n_components=10, random_state=0).fit_transform(final_tf_idf)
viz = TSNE(n_components=2,perplexity = 40,n_iter = 3500).fit_transform(tSVD)
# attaching the label for each 2-d data point
viz_data = np.vstack((viz.T, labels)).T
# creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","n
viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label"
pal = dict(positive="blue", negative="red")
sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negativ
plt.show()
```



1.4.3 Perplexity 50 & n_iter = 5000

```
In [14]: tSVD = TruncatedSVD(n_components=10, random_state=0).fit_transform(final_tf_idf)
         viz = TSNE(n_components=2,perplexity = 50,n_iter = 5000).fit_transform(tSVD)
         # attaching the label for each 2-d data point
         viz_data = np.vstack((viz.T, labels)).T
         # creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","n
         viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label"
         pal = dict(positive="blue", negative="red")
         sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negativ
         plt.show()
```



1.4.4 TF-IDF conclusion

TF-IDF did not produce a good result

There are too much of overlapping

Increasing the perplexity value and the number of iterations produced the same result with different angles of the projection of the chart

1.5 W2V

```
In [15]: # Train your own Word2Vec model using your own text corpus
         import re
         def cleanhtml(sentence): #function to clean the word of any html-tags
             cleanr = re.compile('<.*?>')
             cleantext = re.sub(cleanr, ' ', sentence)
             return cleantext
         def cleanpunc(sentence): #function to clean the word of any punctuation or special char
             cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
             cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
```

```
    return  cleaned

import gensim
i=0
list_of_sent=[]
for sent in final['Text'].values:
    filtered_sentence=[]
    sent=cleanhtml(sent)
    for w in sent.split():
        for cleaned_words in cleanpunc(w).split():
            if(cleaned_words.isalpha()):
                filtered_sentence.append(cleaned_words.lower())
            else:
                continue
    list_of_sent.append(filtered_sentence)
```

```
In [18]: from gensim.models import Word2Vec
         from gensim.models import KeyedVectors
         import pickle
         import gensim
         import numpy as np

         w2v_model=gensim.models.Word2Vec(list_of_sent,min_count=5,size=50, workers=4)
         w2v_model.save("amazon_nlp_sampled.model")
```

1.5.1 Avg W2V

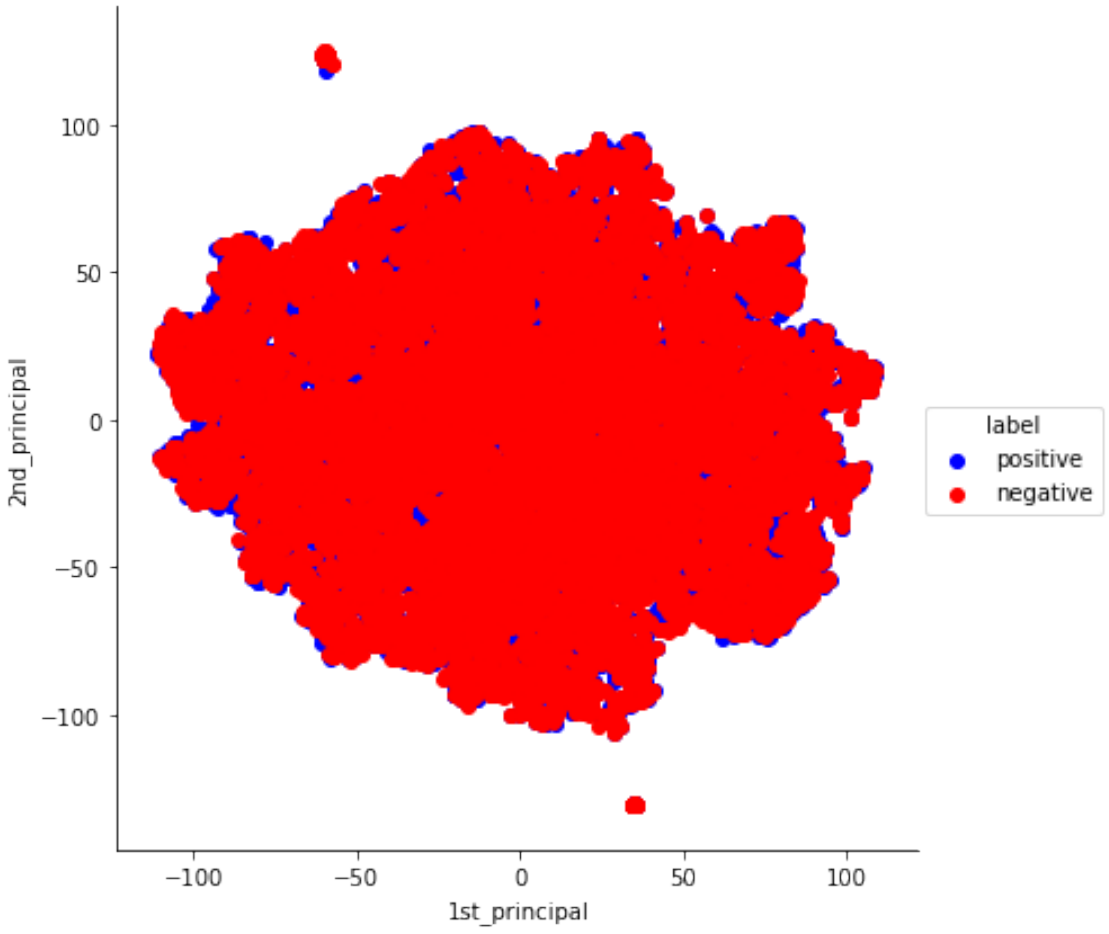
```
In [19]: # average Word2Vec
         # compute average word2vec for each review.
         sent_vectors = []; # the avg-w2v for each sentence/review is stored in this list
         for sent in final.Text.values: # for each review/sentence
             sent_vec = np.zeros(50) # as word vectors are of zero length
             cnt_words =0; # num of words with a valid vector in the sentence/review
             for word in sent: # for each word in a review/sentence
                 try:
                     vec = w2v_model.wv[word]
                     if len(vec) == 0:
                         sent_vec += np.fill(50)
                     else:
                         sent_vec += vec
                     cnt_words += 1
                 except:
                     pass
             sent_vec /= cnt_words
             sent_vectors.append(sent_vec)
         sent_vectors = np.nan_to_num(sent_vectors)
```

```
/usr/local/anaconda/envs/py36/lib/python3.6/site-packages/ipykernel_launcher.py:17: RuntimeWarni
```

```
In [22]: import pickle
         #dumps(sent_vectors, protocol=None, *, fix_imports=True)
         #dumps(sent_vectors, protocol=None, *, fix_imports=True)
         with open("avg_w2v.txt", "wb") as fp:   #Pickling
             pickle.dump(sent_vectors, fp)
```

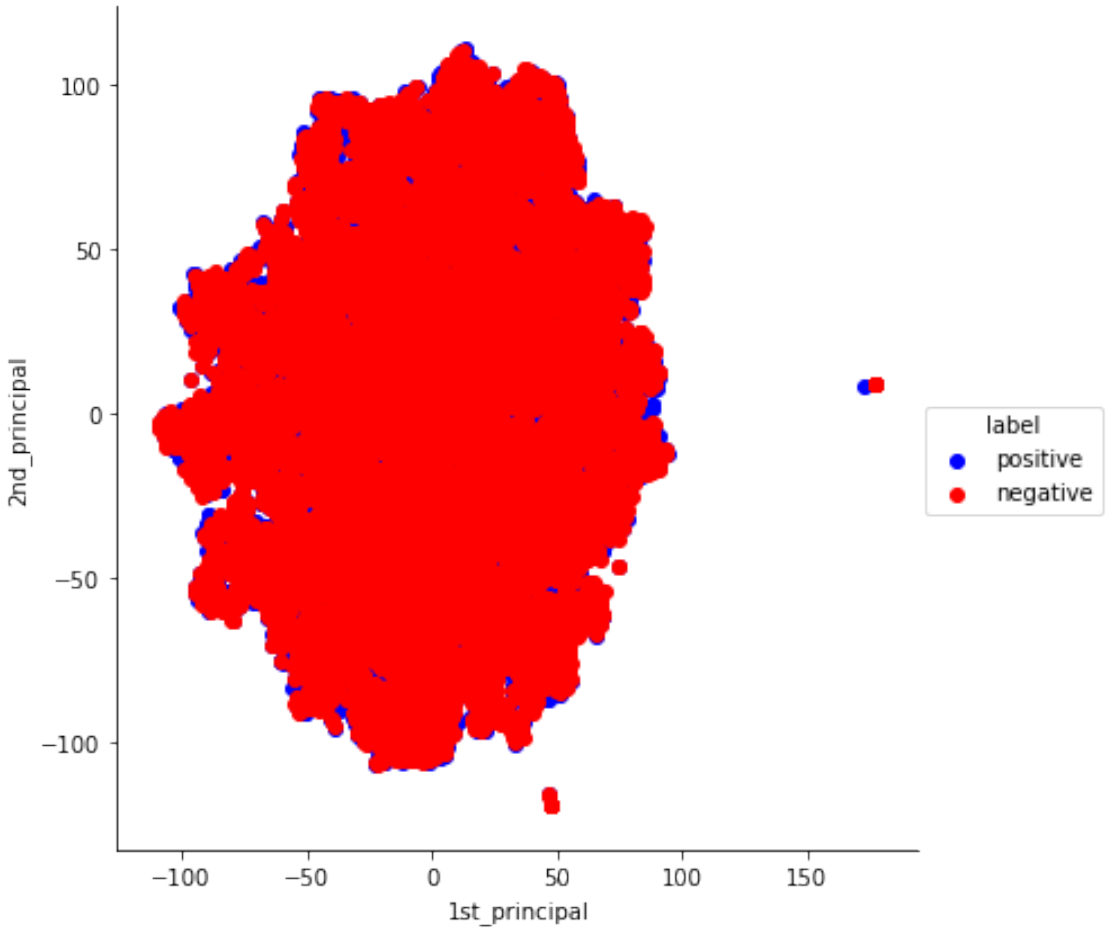
Perplexity 30 & n_iter = 2000

```
In [23]: standardized_data = StandardScaler(with_mean = False).fit_transform(sent_vectors)
         viz = TSNE(n_components=2,perplexity = 30,n_iter = 2000).fit_transform(standardized_dat
         # attaching the label for each 2-d data point
         viz_data = np.vstack((viz.T, labels)).T

         # creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","n
         viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label"
         pal = dict(positive="blue", negative="red")
         sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negativ
         plt.show()
```



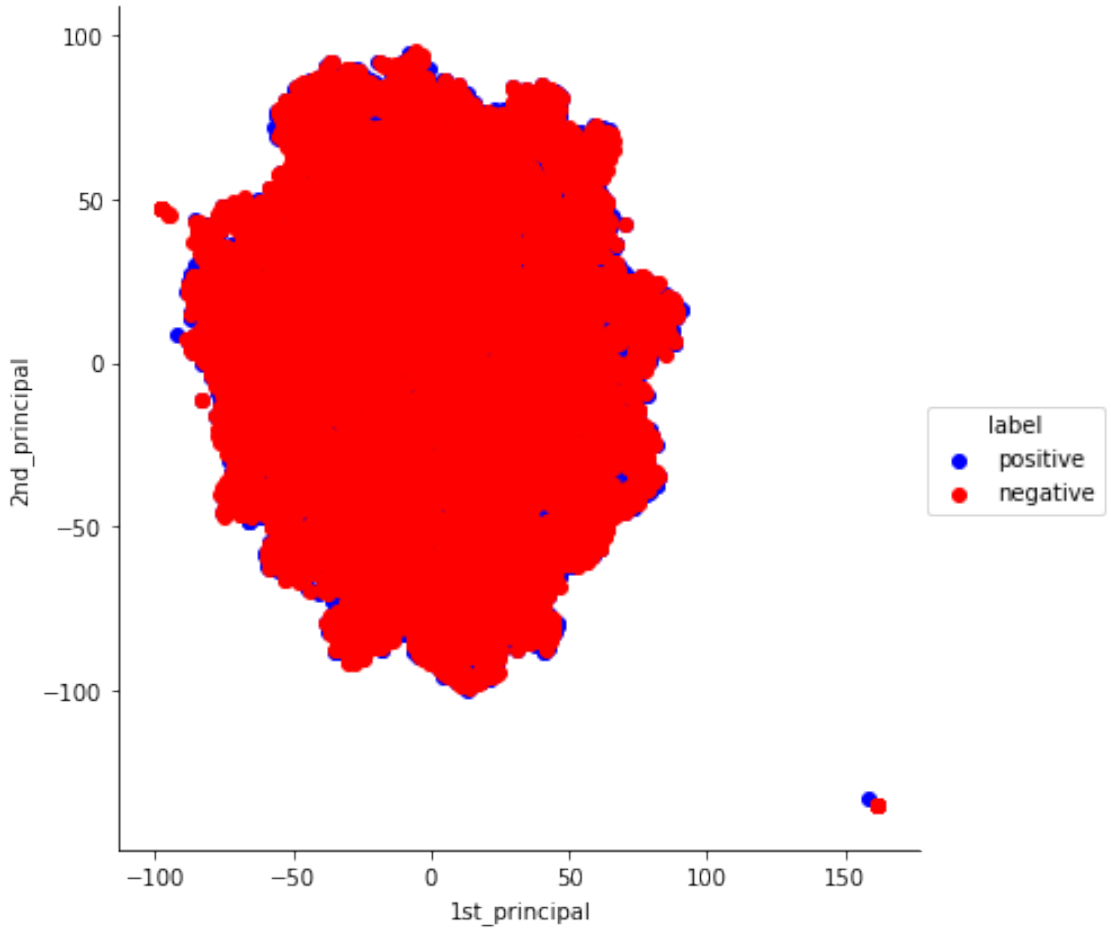
Perplexity 40 & n_iter = 3500

```
In [24]: standardized_data = StandardScaler(with_mean = False).fit_transform(sent_vectors)
         viz = TSNE(n_components=2,perplexity = 40,n_iter = 3500).fit_transform(standardized_dat
         # attaching the label for each 2-d data point
         viz_data = np.vstack((viz.T, labels)).T

         # creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","n
         viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label"
         pal = dict(positive="blue", negative="red")
         sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negativ
         plt.show()
```



Perplexity 50 & n_iter = 5000

In [25]:
```
standardized_data = StandardScaler(with_mean = False).fit_transform(sent_vectors)
viz = TSNE(n_components=2,perplexity = 50,n_iter = 5000).fit_transform(standardized_dat
# attaching the label for each 2-d data point
viz_data = np.vstack((viz.T, labels)).T

# creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","n
viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label"
pal = dict(positive="blue", negative="red")
sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negativ
plt.show()
```



1.5.2 Average W2V conclusion

Average W2V did not produce a good result

There are too much of overlapping

Increasing the perplexity value and the number of iterations produced the same result with different angles of the projection of the chart

1.5.3 TF-IDF W2V

In [26]:
```
# TF-IDF weighted Word2Vec
tfidf_feat = tf_idf_vect.get_feature_names() # tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and cell_val = tfidf

tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is stored in this lis
row=0;
for sent in list_of_sent: # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length
    weight_sum =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        try:
            vec = w2v_model.wv[word]
            # obtain the tf_idfidf of a word in a sentence/review
            tfidf = final_tf_idf[row, tfidf_feat.index(word)]
            sent_vec += (vec * tf_idf)
            weight_sum += tf_idf
        except:
            pass
    sent_vec /= weight_sum
    tfidf_sent_vectors.append(sent_vec)
    row += 1
```

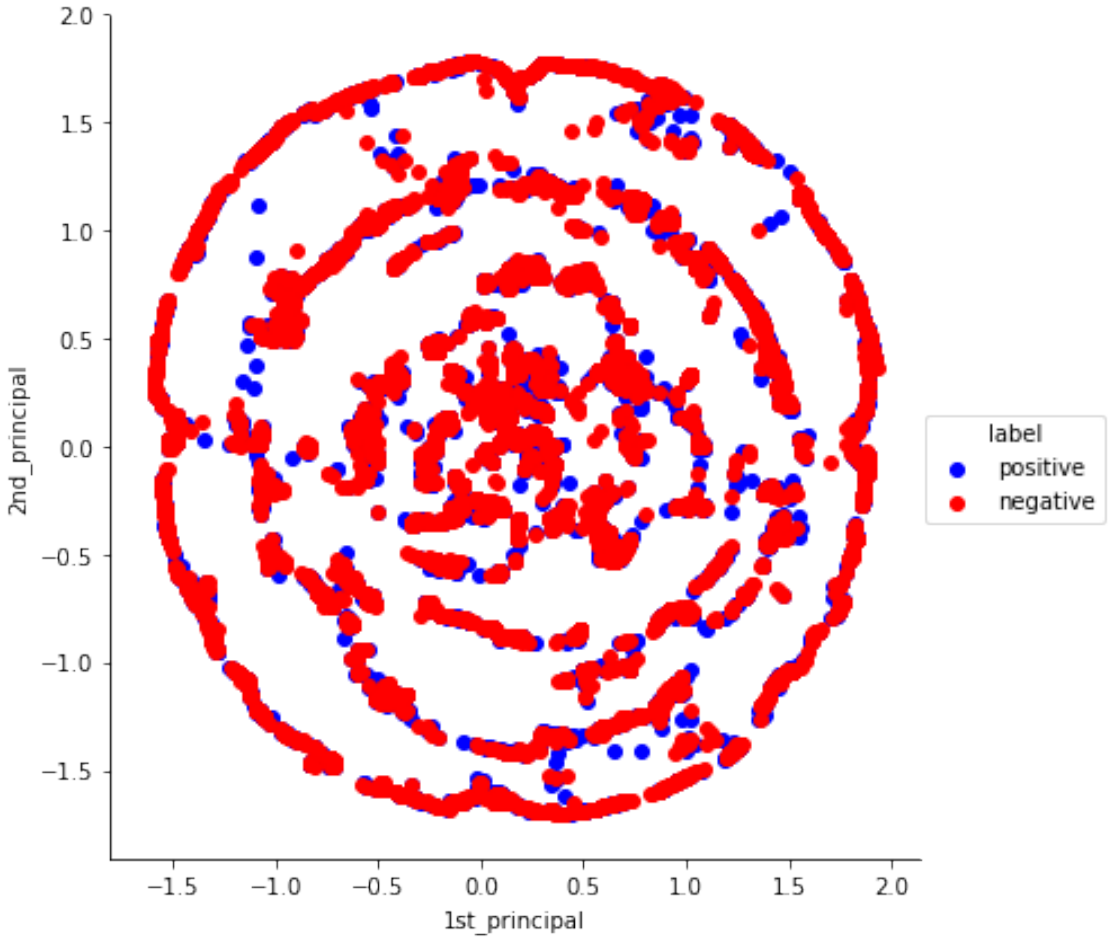
```
/usr/local/anaconda/envs/py36/lib/python3.6/site-packages/ipykernel_launcher.py:19: RuntimeWarni
```

In [27]:
```
import pickle
#dumps(sent_vectors, protocol=None, *, fix_imports=True)
#dumps(sent_vectors, protocol=None, *, fix_imports=True)
with open("avg_tfidf_w2v.txt", "wb") as fp:   #Pickling
    pickle.dump(tfidf_sent_vectors, fp)
```

In [29]:
```
tfidf_sent_vectors = np.nan_to_num(tfidf_sent_vectors)
```

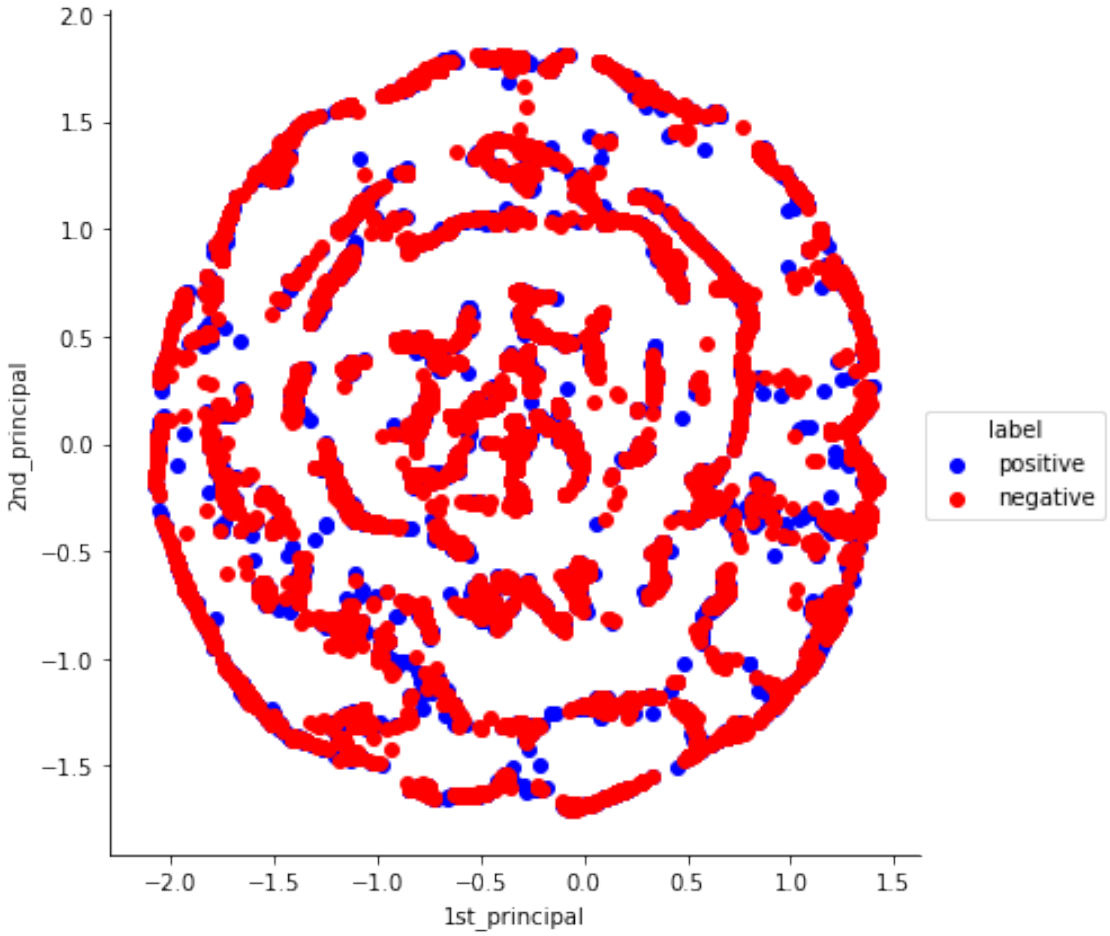
Perplexity 30 & n_iter = 2000

In [30]:
```
standardized_data = StandardScaler(with_mean = False).fit_transform(tfidf_sent_vectors)
viz = TSNE(n_components=2,perplexity = 30,n_iter = 2000).fit_transform(standardized_dat
# attaching the label for each 2-d data point
viz_data = np.vstack((viz.T, labels)).T

# creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","n
viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label"
pal = dict(positive="blue", negative="red")
sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negativ
plt.show()
```



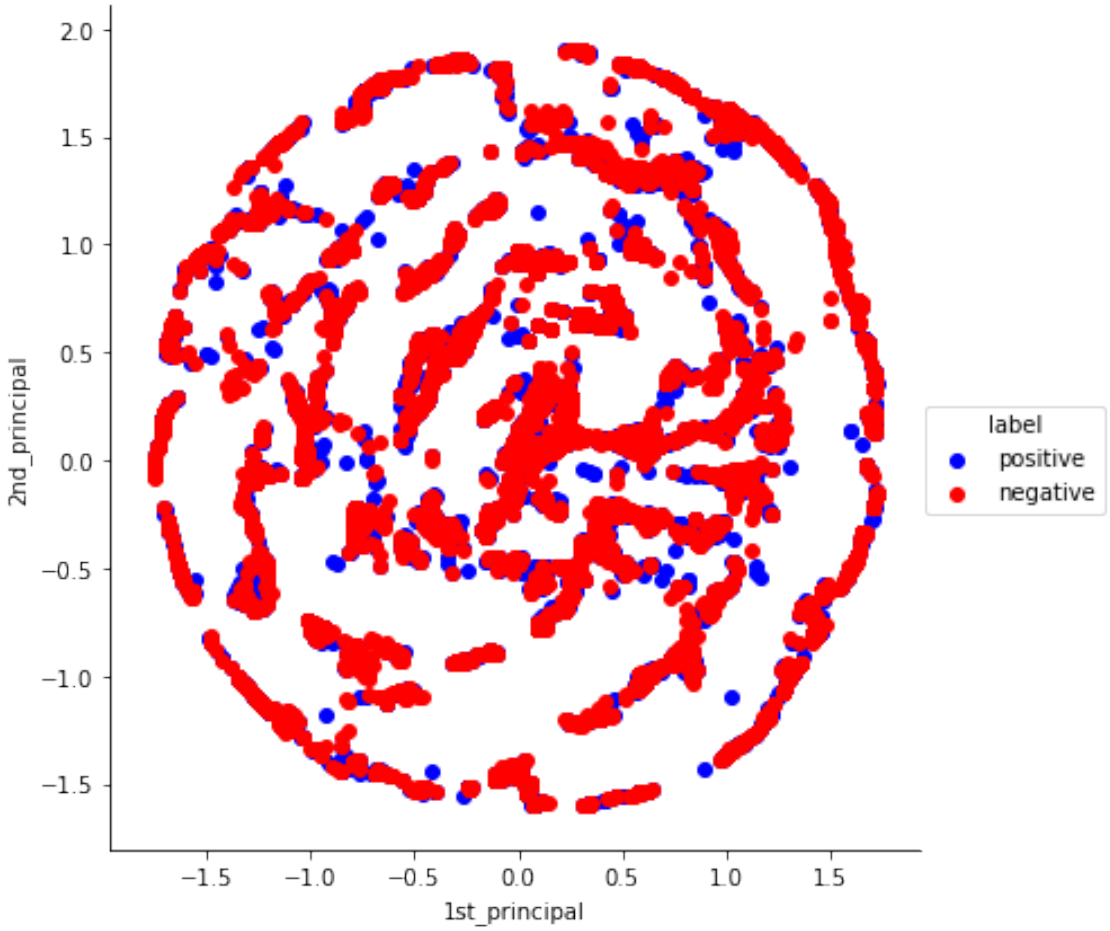
Perplexity 40 & n_iter = 3500

```
In [31]: standardized_data = StandardScaler(with_mean = False).fit_transform(tfidf_sent_vectors)
         viz = TSNE(n_components=2,perplexity = 40,n_iter = 3500).fit_transform(standardized_dat
         # attaching the label for each 2-d data point
         viz_data = np.vstack((viz.T, labels)).T

         # creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","n
         viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label"
         pal = dict(positive="blue", negative="red")
         sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negativ
         plt.show()
```



Perplexity 50 & n_iter = 5000

```
In [32]: standardized_data = StandardScaler(with_mean = False).fit_transform(tfidf_sent_vectors)
         viz = TSNE(n_components=2,perplexity = 50,n_iter = 5000).fit_transform(standardized_dat
         # attaching the label for each 2-d data point
         viz_data = np.vstack((viz.T, labels)).T

         # creating a new data fram which help ussize=5,palette = pal,hue_order = ["positive","n
         viz_df = pd.DataFrame(data=viz_data, columns=("1st_principal", "2nd_principal", "label"
         pal = dict(positive="blue", negative="red")
         sn.FacetGrid(viz_df, hue="label", size=6,palette = pal,hue_order = ["positive","negativ
         plt.show()
```



1.5.4 TF-IDF W2V conclusion

TF-IDF W2V did not produce a good result

There are too much of overlapping

Increasing the perplexity value and the number of iterations produced the same result with different angles of the projection of the chart

1.6 Facts & conclusions

Facts:

The vectors generated from the various methods like BoW, TF - IDF, Avg W2V and TF-IDF W2V will not produce a good result. Since the features are not related to each other.

Increasing the perplexity values generally not more than (50) and increasing the number of iterations will produce a good result

Observations:

Though increasing the perplexity and the number of iterations have not produced a good result to separate the positive and the negative results.

Lower values of perplexity and the number of iterations have almost produced the same result as higher values of perplexity and the number of iterations

Hence, dimensional reduction for word vectors is not useful and visualising it does not produce a fair result.