In [13]:
```
pip install -U gensim
```

```
Collecting gensim
  Downloading gensim-4.2.0-cp39-cp39-win_amd64.whl (23.9 MB)
Requirement already satisfied: scipy>=0.18.1 in c:\users\skand\anaconda3\lib\si
te-packages (from gensim) (1.7.1)
Requirement already satisfied: numpy>=1.17.0 in c:\users\skand\anaconda3\lib\si
te-packages (from gensim) (1.20.3)
Collecting smart-open>=1.8.1
  Downloading smart_open-6.0.0-py3-none-any.whl (58 kB)
Collecting Cython==0.29.28
  Downloading Cython-0.29.28-py2.py3-none-any.whl (983 kB)
Installing collected packages: smart-open, Cython, gensim
  Attempting uninstall: Cython
    Found existing installation: Cython 0.29.24
    Uninstalling Cython-0.29.24:
      Successfully uninstalled Cython-0.29.24
Successfully installed Cython-0.29.28 gensim-4.2.0 smart-open-6.0.0
Note: you may need to restart the kernel to use updated packages.
```

In [25]:
```
import nltk
```

In [26]:
```
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\skand\AppData\Roaming\nltk_data...
```

Out[26]:
```
True
```

In [15]:
```
pip show gensim
```

```
Name: gensim
Version: 4.2.0
Summary: Python framework for fast Vector Space Modelling
Home-page: http://radimrehurek.com/gensim (http://radimrehurek.com/gensim)
Author: Radim Rehurek
Author-email: me@radimrehurek.com
License: LGPL-2.1-only
Location: c:\users\skand\anaconda3\lib\site-packages
Requires: smart-open, scipy, Cython, numpy
Required-by: 
Note: you may need to restart the kernel to use updated packages.
```

In [17]:
```
%matplotlib inline
import re
import sqlite3
import pandas as pd
import numpy as np
import nltk
import tqdm as tqdm
import string
from nltk.corpus import stopwords
stop = stopwords.words("english")
import matplotlib.pyplot as plt
import numpy as np
import datetime as dt
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer
english_stemmer=nltk.stem.SnowballStemmer('english')


from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from math import floor,ceil

from sklearn.svm import LinearSVC

from keras.models import Sequential
from keras.layers import LSTM, Dense, Embedding
import seaborn as sns
```

In [2]:
```
df = pd.read_excel(r'C:\Users\skand\Downloads\interns-tagging_Skandha.xlsx')
print (df)
```

```
     cluster_id                                id  \
0             0  0334a0d055104e9a931c079e338be9a1   
1             0  796d6c25ab8849cbba427f1f3e250d80   
2             0  661f5299cd8944a8a3841fd4f049dee9   
3             0  da831e4bc58d4505aec3c583f0248f8b   
4             0  0ea997675e7344419d1540d3e0bc26c3   
..          ...                               ...   
599          10  93f874167d11473f8d36d1cda0a0081c   
600          10  d50fe37fab064408a891aa9ef45dcd70   
601          10  3e1e8901d5ab4fc9b602ecfdca1220cb   
602          10  c84e1b1196a242d18938af6c60403afc   
603          10  fd4c71f399104d59ad6c1013fc414c67   

                                                phrase     common idea  
0          Would use the product again if needed Joe .  loyal customer  
1           Have been using the product for a week now  loyal customer  
2     Will continue to use this product when I have ...  loyal customer  
3         Have always had good luck with this product .  loyal customer  
4     Will continue to use This product as This prod...  loyal customer  
..                                                 ...             ...  
599                          Spray has no strong odor      good smell  
600   Spray is nice to keep out on the porch on a su...      good smell  
601             Spray does not leave any oily stinky stains      good smell  
602   Love that the scent of this spray is not chemi...      good smell  
603               Bug spray does not smell nauseating .      good smell  

[604 rows x 4 columns]
```

Data Pre Processing

In [7]:
```
def cleaning( review, remove_stopwords=True):



    review_text = re.sub("[^a-zA-Z]"," ", review)

    words = review_text.lower().split()

    if remove_stopwords:
        stops = set(stopwords.words("english"))
        words = [w for w in words if not w in stops]

    b=[]
    stemmer = english_stemmer
    for word in words:
        b.append(stemmer.stem(word))


    return(b)
```

Cleaning

```
In [18]: clean_Text = []
         for review in df['phrase']:
             clean_Text.append( " ".join(cleaning(review)))

         clean_summary = []
         for review in df['common idea']:
             clean_summary.append( " ".join(cleaning(review)))
```

Top Word Count In Text(Review)

```
In [19]: Top_Words_Review =pd.Series(' '.join(clean_Text).lower().split()).value_counts()[
         print ("Top Count Words Used In Review", Top_Words_Review)
```

```
Top Count Words Used In Review product      102
safe           82
use            79
smell          72
around         57
pest           47
recommend      46
pet            38
good           38
great          36
dtype: int64
```

```
In [20]: Top_Words_Summary = pd.Series(' '.join(clean_summary).lower().split()).value_cour
         print ("Top Count Words Used In Summary",Top_Words_Summary)
```

```
Top Count Words Used In Summary effici     142
product    142
good       141
smell      141
safe       117
kid        117
pet        117
loyal       62
custom      62
valu        61
dtype: int64
```

Tf-idf

```
In [23]: from sklearn.feature_extraction.text import TfidfVectorizer
         vectorizer = TfidfVectorizer(min_df=4, max_features = 10000)
         vz = vectorizer.fit_transform(clean_Text)
         tfidf = dict(zip(vectorizer.get_feature_names(), vectorizer.idf_))
```

Sentiment Analysis

In [27]:
```python
from nltk.sentiment.vader import SentimentIntensityAnalyzer
Senti = SentimentIntensityAnalyzer()
sample_review = clean_Text[:5]
for sentence in sample_review:
    sentence
    ss = Senti.polarity_scores(sentence)
    for k in sorted(ss):
        print('{0}: {1}, '.format(k, ss[k]))
    print(sentence)
```

```
compound: 0.0,
neg: 0.0,
neu: 1.0,
pos: 0.0,
would use product need joe
compound: 0.0,
neg: 0.0,
neu: 1.0,
pos: 0.0,
use product week
compound: 0.0,
neg: 0.0,
neu: 1.0,
pos: 0.0,
continu use product issu
compound: 0.7096,
neg: 0.0,
neu: 0.253,
pos: 0.747,
alway good luck product
compound: 0.0,
neg: 0.0,
neu: 1.0,
pos: 0.0,
continu use product product get job done
```

K means

```
In [28]: from sklearn.cluster import MiniBatchKMeans

         num_clusters = 10
         kmeans_model = MiniBatchKMeans(n_clusters=num_clusters, init='k-means++', n_init=
                                        init_size=1000, batch_size=1000, verbose=False, max_iter
         kmeans = kmeans_model.fit(vz)
         kmeans_clusters = kmeans.predict(vz)
         kmeans_distances = kmeans.transform(vz)
         sorted_centroids = kmeans.cluster_centers_.argsort()[:, ::-1]
         terms = vectorizer.get_feature_names()
         for i in range(num_clusters):
             print("Cluster %d:" % i)
             for j in sorted_centroids[i, :5]:
                 print(' %s' % terms[j])
             print()
```

```
Cluster 0:
 recommend
 would
 friend
 product
 anyon

Cluster 1:
 safe
 around
 kid
 children
 use

Cluster 2:
 smell
 chemic
 work
 bad
 spray

Cluster 3:
 odor
 product
 strong
 bad
 effect

Cluster 4:
 price
 great
 good
 worth
 afford

Cluster 5:
 pet
 friend
 safe
 stuff
 good

Cluster 6:
 scent
 strong
 lemon
 fresh
 nice

Cluster 7:
 use
 product
 year
 sever
 continu

Cluster 8:
 bug
 ant
 deliveri
 valu
 fast

Cluster 9:
 pest
 control
 awesom
 effect
 keep
```

In [ ]:

In [ ]:

In [ ]: