

LSTM on Amazon fine food review

```
In [1]: import sqlite3
import pandas as pd
import numpy as np
import re
import string
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer as sno
setofstopwords=set(stopwords.words('english'))
nltk.download('stopwords')

[nltk_data] Downloading package stopwords to C:\Users\Ankit Kumar
[nltk_data] Singh\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!

Out[1]: True
```

Loading from database

```
In [2]: conn= sqlite3.connect('database.sqlite')
data= pd.read_sql_query('''
SELECT * FROM Reviews WHERE Score!=3
''',conn)
data.shape
```

Out[2]: (525814, 10)

Removing not helpful reviews

```
In [3]: data['Score']=data['Score'].map(lambda x: 'Positive' if x>3 else 'Negative')
sorteddata= data.sort_values('ProductId',axis=0)
finaldata= sorteddata.drop_duplicates(subset={'UserId','ProfileName',\
      'Time','Text'}, keep='first',inplace=False)

finaldata= finaldata[finaldata['HelpfulnessNumerator'] <= finaldata['HelpfulnessDenominator']]
data= finaldata.sort_values('Time',axis=0)
data.shape
```

Out[3]: (364171, 10)

Cleaning HTML, punctuations, apply stemming, lowercasing etc without removing stopwords

```
In [4]: def cleanhtml(sentence): #substitute expression contained in <> with ' '
        cleaned= re.sub(re.compile('<.*?>'),' ',sentence)
        return cleaned
#function for removing punctuations chars
def cleanpunc(sentence):
    cleaned= re.sub(r'[?|!|\'|\"|#]',r'',sentence)
    cleaned= re.sub(r'[.,,)]|(|\\|/]',r'',sentence)
    return cleaned
snowstem= sno('english')

i=0
str1=' '
final_string=[]
all_positive_words=[] # store words from +ve reviews here
all_negative_words=[] # store words from -ve reviews here.
```

```

for sent in data['Text'].values:
    filtered_sentence=[]
    #print(sent);
    sent=cleanhtml(sent) # remove HTML tags
    for w in sent.split():
        # we have used cleanpunc(w).split(), one more split function here
        # because consider w="abc.def", cleanpunc(w) will return "abc def"
        # if we dont use .split() function then we will be considering
        "abc def"
        # as a single word, but if you use .split() function we will get
        "abc", "def"
        for cleaned_words in cleanpunc(w).split():
            if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
                s=(snowstem.stem(cleaned_words.lower())).encode('utf8')
                filtered_sentence.append(s)
                if(data['Score'].values[i] == 'Positive':
                    all_positive_words.append(s)
                if(data['Score'].values[i] == 'Negative':
                    all_negative_words.append(s)
            else:
                continue
        str1 = b" ".join(filtered_sentence) #final string of cleaned words
        final_string.append(str1)

# storing data till now
data['CleanedText']=final_string
#adding a column of CleanedText which displays the data after pre-processing of the review
data['CleanedText']=data['CleanedText'].str.decode("utf-8")
    # store final table into an SQLite table for future.
conn = sqlite3.connect('cleanedTextData.sqlite')
c=conn.cursor()
conn.text_factory = str
data.to_sql('Reviews', conn, schema=None, if_exists='replace', \
            index=True, index_label=None, chunksize=None, dtype=None)
conn.close()

```

In [5]: `data.head()`

Out[5]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator
138706	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0
138683	150501	0006641040	AJ46FKXOVC7NR	Nicholas A Mesiano	2	2
417839	451856	B00004CXX9	AIUWLEQ1ADEG5	Elizabeth Medina	0	0
346055	374359	B00004CI84	A344SMIA5JECGM	Vincent P. Ross	1	2
417838	451855	B00004CXX9	AJH6LUC1UT1ON	The Phantom of the Opera	0	0

In [7]: `data['Text'][2]`

```
Out[7]: 'This is a confection that has been around a few centuries. It is a light, pillowy citrus gelatin with nuts - in this case Filberts. And it is cut into tiny squares and then liberally coated with powdered sugar. And it is a tiny mouthful of heaven. Not too chewy, and very flavorful. I highly recommend this yummy treat. If you are familiar with the story of C.S. Lewis\' "The Lion, The Witch, and The Wardrobe" - this is the treat that seduces Edmund into selling out his Brother and Sisters to the Witch.'
```

```
In [8]: data['CleanedText'][2]
```

```
Out[8]: 'this confect that has been around few centuri light pillowi citrus gelatin with nut this case filbert and cut into tini squar and then liber coat with powder sugar and tini mouth heaven not too chewi and veri flavor high recommend this yummi treat you are familiar with the stori lion the witch and the this the treat that seduc edmund into sell out his brother and sister the witch'
```

Taking 100k datapoints

```
In [20]: Data= data[:100000]
Data= Data[['CleanedText', 'Score']]
Data['Score']= Data['Score'].map(lambda x:1 if x=='Positive' else 0)

Data_x= Data['CleanedText']
Data_y= Data['Score']
```

```
In [82]: Data_x.index= [i for i in range(0,10**5)]
Data_x
```

```
Out[82]: 0      this witti littl book make son laugh loud reci...
1      can rememb see the show when air televis year ...
2      beetlejuic well written movi everyth about fro...
3      twist rumplestiskin captur film star michael k...
4      beetlejuic excel and funni movi keaton hilari ...
5      this one movi that should your movi collect fi...
6      myself alway enjoy this movi veri funni and en...
```

7 bought few these after apart was infest with f...
8 what happen when you say his name three michae...
9 get look for beatlejuic french version video r...
10 get crazi realli imposs today not find the fre...
11 this was realli good idea and the final produc...
12 just receiv shipment and could hard wait tri t...
13 have just recent purchas the woodstream corp g...
14 this are much easier use than the wilson past ...
15 these are easi use they not make mess and offe...
16 this such great film even know how sum first a...
17 beetlejuic wonder amus comed romp that explor ...
18 sick scad nasti toothpick all over counter whe...
19 thought this movi was funni michael keaton bee...
20 mani movi have dealt with the figur death and ...
21 know whi anyon would ever use those littl liqu...
22 michael keaton bring distinguish characterist ...
23 continu amaz the shoddi treatment that some mo...
24 just warn you when tri trick you the widescree...
25 bought these decor some dia los muerto skull w...
26 winona ryder the gothic princess doom see for ...
27 this was favorit book mine when was littl girl...
28 for year have been tri simul truli italian esp...
29 when vacat adam and barbara maitland meet thei...

...
99970 like strong black coffe like tast strong have ...
99971 absolut love this tea drink tea everyday and h...
99972 about year age our cat would have bladder infe...
99973 bought the turkish delight and then read the r...
99974 husband recent ate outsid cafe geneva was late...
99975 was suffer from cold for almost year spent lot...
99976 neighbor and love this candi and difficult fin...
99977 dog absolut love yummi chummi use for train tr...
99978 this second purchas and plan stay stock with t...
99979 was excit when found whole wheat isra couscous...
99980 bought dozen these monkey lollipop parti favor...
99981 order three set these from entirelypet even ca...
99982 special protein plus far favorit cereal the on...
99983 was given the opportun review this product bel...
99984 gloria jean hit out the park this whi not ther...

```
99985    yuck this coffe tast terribl doe not compet th...
99986    order this item becaus when purchas from local...
99987    compar other nutrit bar out there find these a...
99988    love peanut butter love that they contain ton ...
99989    two young adult siberian huski love these they...
99990    love teaand realli hate write bad review mayb ...
99991    husband and were alway afraid make fish like m...
99992    ice breaker ice cube peppermint sugar free gum...
99993    say that name this coffe blend appropri found ...
99994    was look for coffe replac and this fit the bil...
99995    this veri tasti protein shake and give you nic...
99996    this the best tast oliv and healthi for you to...
99997    sack melitta coffe label fine grind blanc noir...
99998    final babi food that tast love that great prot...
99999    first ventur salt and happi bought this and sa...
Name: CleanedText, Length: 100000, dtype: object
```

Making Vocabulary set and Frequency dictionary of words

```
In [45]: # collecting all words in single list
list_ = []
for i in Data_x:
    list_ += i
list_ = ''.join(list_)
allWords=list_.split()
```

```
In [56]: vocabulary= set(allWords)
```

```
In [60]: vocabulary_list= list(vocabulary)
```

```
In [62]: #frequency dictionary
freq_dict= {}
for word in vocabulary_list:
    freq_dict[word]= allWords.count(word)
```

```
In [64]: freq_dict
```

```
Out[64]: {'weightthis': 7,  
          'againdentist': 1,  
          'milibar': 1,  
          'almondopen': 1,  
          'needuse': 2,  
          'getcha': 1,  
          'casealthough': 1,  
          'excelhusband': 3,  
          'glycemix': 1,  
          'pricesuch': 1,  
          'tastestri': 1,  
          'enforc': 4,  
          'adamhave': 1,  
          'eitherheard': 1,  
          'gravel': 16,  
          'fluffihave': 1,  
          'giftmake': 2,  
          'futil': 3,  
          'ratewhen': 1,  
          'statego': 1,  
          'dealorvill': 1,  
          'mcdougalorder': 1,  
          'art': 123,  
          'coppicino': 1,  
          'intensmuch': 1,  
          'turkeynot': 1,  
          'prais': 116,  
          'caseshare': 1,  
          'lovenormal': 3,  
          'saltlive': 1,  
          'railway': 1,  
          'othercrazi': 1,  
          'wazu': 1,  
          'weekcent': 1,  
          'coloraft': 1,  
          'asthamt': 1,  
          'adbig': 1,  
          'outragtri': 1,  
          'thougheveryon': 1,
```



```
'owenlike': 1,  
'stravecchio': 1,  
'eatcultiv': 1,  
'believthe': 3,  
'cartthis': 1,  
'issuesat': 1,  
'theobromin': 9,  
'bussi': 4,  
'simiar': 2,  
'immediwas': 1,  
'promishow': 1,  
'detoxifiuse': 1,  
'foamread': 1,  
'breadmix': 3,  
'thesedoctor': 1,  
'productcoffe': 1,  
'chickenan': 1,  
'getorder': 7,  
'morecavali': 1,  
'youunfroast': 1,  
'alikkid': 1,  
'totalfirst': 1,  
'equalthis': 1,  
'carcinogendaughter': 1,  
'marketanoth': 1,  
'rushbought': 1,  
'soot': 2,  
'quisp': 9,  
'luckbox': 1,  
'goeuse': 1,  
'kennewicktrust': 1,  
'decaffeni': 1,  
'stockconveni': 1,  
'raenot': 1,  
'unspeak': 1,  
'andone': 2,  
'stratospher': 1,  
'pleasedto': 1,  
'two': 7369,
```

```
'flourand': 4,  
'chitosan': 1,  
'comparlove': 1,  
'gratinstumbl': 1,  
'onlilove': 2,  
'lolthe': 3,  
'dairiuse': 2,  
'mooshyso': 1,  
'nosenon': 1,  
'loofah': 3,  
'greatexcel': 3,  
'alasken': 1,  
'themstop': 1,  
'whitebread': 1,  
'seedwas': 1,  
'menueven': 1,  
'flavorokay': 1,  
'usualpleasant': 1,  
'alternlayla': 1,  
'laterveri': 1,  
'algaecid': 1,  
'revisit': 5,  
'flexse': 1,  
'muchprobabl': 1,  
'closettri': 1,  
'differet': 1,  
'awhiluse': 1,  
'cornnavi': 1,  
'helen': 8,  
'choc': 103,  
'justbasic': 1,  
'casegood': 2,  
'strawberrifamili': 1,  
'spalsh': 1,  
'coarsen': 1,  
'nondescript': 2,  
'mycup': 1,  
'waspackag': 1,  
'surf': 19,
```

```
'guilthave': 2,  
'notall': 1,  
'clothingbed': 1,  
'suretast': 1,  
'truliold': 1,  
'learnenjoy': 1,  
'seizur': 31,  
'workother': 1,  
'pricehorizon': 1,  
'espeschi': 1,  
'teethyou': 2,  
'kidand': 3,  
'dealership': 4,  
'amaretti': 6,  
'carnpuppi': 1,  
'açai': 2,  
'grip': 20,  
'thinkhusband': 1,  
'chanakra': 1,  
'itbought': 1,  
'gooooood': 3,  
'eitherwas': 2,  
'cellophan': 51,  
'truththis': 2,  
'stape': 1,  
'slavethought': 1,  
'disgus': 1,  
'shotsammi': 1,  
'deliciouscrunchyand': 1,  
'popcornwith': 1,  
'oldthere': 1,  
'fromrealli': 2,  
'pitathis': 1,  
'stickthese': 2,  
'youwholesom': 1,  
'beathave': 7,  
'fresht Trident': 1,  
'kebabbabi': 1,  
'glassput': 1,
```

'waythese': 15,
'containwent': 1,
'activthis': 2,
'inelig': 2,
'sugaranoth': 1,
'bearalway': 1,
'treatbeen': 5,
'flavorsw': 1,
'dipgot': 1,
'videogam': 1,
'jampreserv': 3,
'tautlove': 1,
'teethgluten': 1,
'andwatch': 1,
'noël': 1,
'tastwhen': 2,
'oventhe': 1,
'horriblewors': 1,
'trirotini': 1,
'urologist': 1,
'presentthis': 5,
'pareil': 3,
'anddeliv': 1,
'etcson': 2,
'model': 81,
'deliverynop': 1,
'coincid': 27,
'ymmvwas': 1,
'recommendwhen': 13,
'zeus': 3,
'purhas': 1,
'formi': 1,
'twiggl': 1,
'mainsteam': 1,
'catechu': 1,
'articifici': 2,
'pekinges': 5,
'balsalm': 4,
'potatowith': 1,

'raini': 37,
'ceylones': 3,
'mesa': 18,
'excelprobabl': 1,
'quantitihonest': 1,
'beefthen': 1,
'sweetentri': 1,
'terrific': 1,
'betterkds': 1,
'caffinate': 1,
'glutensnapea': 1,
'coffebefor': 1,
'erwachsen': 2,
'disloc': 3,
'whatthes': 1,
'elli': 4,
'stufforder': 1,
'greatknow': 1,
'availhad': 8,
'shipwont': 1,
'independ': 53,
'perkinknow': 1,
'controlcreami': 1,
'madebengal': 1,
'nibthe': 1,
'likelihoodonli': 1,
'flavoryear': 1,
'horrif': 18,
'wafersher': 1,
'packagepopcorn': 1,
'proliflove': 1,
'alonlhasa': 1,
'seiyo': 1,
'band': 66,
'neverbean': 1,
'menu': 81,
'growhave': 2,
'restraint': 5,
'huckthought': 1,

```
'riskthis': 1,  
'balsamica': 1,  
'amountvolum': 1,  
'shouldjack': 1,  
'rocklove': 1,  
'bight': 1,  
'nativwife': 1,  
'frozeif': 1,  
'milkno': 1,  
'dribbl': 15,  
'anniv': 3,  
'nothwas': 1,  
'catorder': 1,  
'clamliquid': 1,  
'ordertoo': 1,  
'productyerbam': 1,  
'snackrealli': 4,  
'poorlove': 1,  
'cabingreat': 1,  
'preservrecent': 1,  
'gook': 2,  
'causfirst': 1,  
'joop': 1,  
'closegreat': 1,  
'shippiong': 1,  
'shoppfamili': 1,  
'inositol': 6,  
'rightth': 4,  
'teayummi': 2,  
'packagegood': 1,  
'lovecould': 4,  
'nutritionistthis': 1,  
'finead': 1,  
'farfall': 3,  
'francais': 1,  
'otherwhen': 1,  
'heavenhave': 2,  
'convo': 2,  
'youbake': 1,
```

```
'helllllooooooooooooo': 1,  
'reddrink': 1,  
'healthierorder': 1,  
'antimicrobi': 5,  
'difficultth': 1,  
'fullactual': 1,  
'croy': 1,  
'hitchthis': 2,  
'daycrunchi': 1,  
'lovefat': 1,  
'lifeokthey': 1,  
'plantthese': 3,  
'fix': 546,  
'herehey': 1,  
'tablet': 239,  
'be': 4098,  
'lunchstrong': 1,  
'trailerfirst': 1,  
'enjoyableeveri': 1,  
'yourcan': 1,  
'patient': 137,  
'downswitch': 1,  
'juiceit': 2,  
'tealove': 31,  
'yumolemon': 1,  
'driveour': 1,  
'fordefinit': 2,  
'comelocal': 1,  
'fruitveggiegrain': 1,  
'drooli': 1,  
'ahitunaa': 1,  
'flaketook': 1,  
'muffaletta': 5,  
'bjs': 12,  
'wooden': 52,  
'separ': 528,  
'cookiwonder': 1,  
'carbontire': 1,  
'flieswith': 1,
```

```
'scoobi': 9,  
'pleez': 1,  
'nagoya': 1,  
'trailbraz': 1,  
'competituse': 1,  
'diceddri': 1,  
'quiver': 1,  
'changgood': 1,  
'regetta': 1,  
'sellerall': 1,  
'yugoslaviathese': 1,  
'smalllove': 2,  
'instalthis': 1,  
'againgot': 12,  
'pricehave': 52,  
'fantastanyon': 2,  
'szechwan': 2,  
'replacwant': 1,  
'bedtimcitric': 1,  
'happiresearch': 1,  
'shipmentyou': 1,  
'protienlove': 2,  
'embarras': 4,  
'teacokewinemilkb': 1,  
'yesth': 3,  
'etcverithis': 1,  
'composthave': 1,  
'wound': 85,  
'anothtook': 1,  
'megaglia': 1,  
'storetast': 5,  
'splendalbs': 1,  
'mate': 207,  
'muffindonut': 1,  
'bisquwas': 1,  
'odorold': 1,  
'mixgive': 1,  
'truehas': 1,  
'teasent': 1,
```



```
'math': 76,  
'syropsand': 1,  
'welcom': 165,  
'tootetley': 1,  
'shipalthough': 2,  
'preplan': 1,  
'reviewgreat': 3,  
'shroomlici': 1,  
'lessbought': 2,  
'thanknot': 1,  
'cakeinvest': 1,  
'favoritafter': 1,  
'richgoat': 1,  
'lightish': 1,  
'gretchen': 2,  
'cavatelli': 2,  
'assurtill': 1,  
'hannaford': 3,  
'microwavdaughter': 1,  
'similac': 106,  
'coarser': 19,  
'strick': 1,  
'haveare': 1,  
'mealgluten': 1,  
'stockabout': 1,  
'calisparra': 1,  
'andadmit': 1,  
'happitook': 1,  
'priceyi': 1,  
'brita': 6,  
'shipabsolut': 1,  
'terrificveri': 1,  
'teafamili': 6,  
'thansk': 1,  
'thosegreat': 1,  
'foodbar': 2,  
'uncurl': 1,  
'supplimentnot': 1,  
'xylo': 1,
```

```
'compair': 10,  
'durban': 4,  
'shahi': 2,  
'suffertri': 1,  
'pankosix': 1,  
'goopyso': 1,  
'measurthis': 1,  
'semana': 1,  
'msgnow': 1,  
'misolove': 1,  
'minedeploy': 1,  
'cookreceiv': 1,  
'mealnice': 1,  
'unbrand': 1,  
'wolfbe': 1,  
'matchaalso': 1,  
'fireplac': 13,  
'entourag': 1,  
'lightboy': 1,  
'thingfound': 1,  
'jewelosco': 1,  
'veriall': 1,  
'makerthis': 3,  
'packbuy': 3,  
'favoritlike': 2,  
'packheard': 1,  
'thankyou': 15,  
'cajun': 74,  
'dond': 1,  
'groceridog': 1,  
'sky': 24,  
'packageor': 1,  
'wtfyou': 1,  
'hiz': 1,  
'pincer': 1,  
'guiltgood': 1,  
'economalreadi': 1,  
'nymph': 1,  
'recommendlong': 1,
```

```
'staph': 6,  
'deservnestl': 1,  
'sandwhich': 14,  
'stockveri': 1,  
'sayorder': 1,  
'hassl': 141,  
'aftertastwhen': 1,  
'rangbuy': 1,  
'storesomewhat': 1,  
'souc': 3,  
'tower': 27,  
'cupbought': 1,  
'sweetnot': 4,  
'herepurchas': 2,  
'watersh': 1,  
'iceblend': 1,  
'freezerafter': 1,  
'abd': 2,  
'excelfinal': 1,  
'tastegum': 1,  
'cookiyou': 2,  
'boccherini': 1,  
'mundan': 7,  
'toorex': 1,  
'customtime': 1,  
'kennel': 18,  
'itif': 6,  
'siteknow': 1,  
'wellbeagl': 1,  
'wate': 2,  
'kestekidi': 4,  
'rid': 261,  
'grillgood': 2,  
'goebelthe': 1,  
'sugarlow': 2,  
'bitterorder': 1,  
'crackerrealli': 1,  
'negligexcel': 1,  
'flatthese': 1,
```

```
'licorich': 1,  
'alaea': 3,  
'ordereveryon': 1,  
'timeorigin': 1,  
'exposur': 35,  
'swiss': 195,  
'againcustom': 1,  
'bedsid': 8,  
'scotland': 50,  
'zincuse': 1,  
'piccalilli': 1,  
'potao': 2,  
'mmmmmmmmmmmmmmmmmmmm': 1,  
'cheapideal': 1,  
'cupcafix': 1,  
'thanth': 1,  
'flourorgan': 1,  
'packaag': 1,  
'seaport': 1,  
'viger': 2,  
'snapthey': 1,  
'officfeel': 1,  
'rocki': 37,  
'wayfriend': 1,  
'nostaglia': 1,  
'thighship': 1,  
'zingsad': 1,  
'oilsaltpopcorn': 1,  
'brainer': 27,  
'tou': 3,  
'hostag': 1,  
'tastiunfortun': 1,  
'currenc': 3,  
'where': 3080,  
'intriqu': 2,  
'weakdrink': 1,  
'breaksimpli': 1,  
'eatwater': 1,  
'whileit': 2,
```

```
'sitethey': 1,  
'passibl': 1,  
'lovegood': 10,  
'spaghetti': 4,  
'ironi': 3,  
'notthen': 1,  
'oneshame': 1,  
'extractlove': 1,  
'glutengrain': 1,  
'twee': 2,  
'foundabsolut': 1,  
'jacksboro': 1,  
'taiwan': 32,  
'pastatasti': 1,  
'enoughbrooklynit': 1,  
'recommendaadmit': 3,  
'cape': 23,  
'johnhave': 2,  
'whilepound': 1,  
'pantriwere': 1,  
'conditionwhich': 1,  
'andwould': 1,  
'basiafter': 1,  
'syrupthink': 2,  
'unrinari': 1,  
'thiswell': 5,  
'heavithe': 1,  
'expel': 28,  
'myselfthey': 2,  
'rosell': 1,  
'guarantebegan': 1,  
'coffeshop': 1,  
'brandbet': 1,  
'eitherreceiv': 1,  
'fareit': 1,  
'poisen': 5,  
'easicould': 1,  
'agefor': 1,  
'herselfsent': 2,
```

```
'scali': 5,  
'soeth': 2,  
'fungi': 2,  
'pursthis': 1,  
'ceci': 1,  
'cookerlook': 1,  
'schultz': 4,  
'petit': 166,  
'enfamilgrow': 1,  
'musicor': 1,  
'justrare': 1,  
'favoritnormal': 1,  
'themlike': 17,  
'stockbeen': 1,  
'themhave': 86,  
'thinggrape': 1,  
'deali': 2,  
'nightnever': 1,  
'buzzlove': 1,  
'dailiexcel': 1,  
'eggplantnot': 1,  
'tabletson': 1,  
'itallian': 2,  
'chocki': 2,  
'serimonth': 1,  
'leastregular': 1,  
'toomin': 1,  
'alumnus': 1,  
'howbuy': 1,  
'evengrew': 1,  
'bob': 49,  
'stockcat': 2,  
'hereknow': 1,  
'pomegranhave': 1,  
'foundrica': 1,  
'starboy': 1,  
'likeamazon': 1,  
'zum': 1,  
'whatevwild': 1,
```

'cantthese': 1,
'packagproduct': 1,
'appetitpurchas': 2,
'looooov': 14,
'daythere': 1,
'rightcustom': 1,
'raeli': 1,
'awarthis': 1,
'mamma': 5,
'loner': 1,
'spicemouth': 1,
'producreceiv': 1,
'nutrientlove': 1,
'krispihas': 1,
'ointment': 5,
'liyaqat': 1,
'thanxwent': 1,
'irishdelici': 1,
'sharp': 276,
'verior': 2,
'yucko': 1,
'eatergrew': 1,
'choicbeen': 2,
'coffewould': 3,
'busi': 971,
'schoolsorri': 1,
'forador': 2,
'dilemmathis': 1,
'davisonnew': 1,
'molecul': 37,
'attributwhen': 1,
'homini': 5,
'toooo': 7,
'caserecal': 1,
'staplfirst': 1,
'fountain': 1,
'menus': 10,
'closewhen': 1,
'washno': 1,

```
'bestgive': 2,  
'aromaeasi': 1,  
'trapknow': 1,  
'tek': 5,  
'saymotor': 1,  
'gutter': 4,  
'chattanooga': 2,  
'itdo': 2,  
'waterbelt': 1,  
'hibachi': 2,  
'varihad': 1,  
'discriminatori': 1,  
'wet': 495,  
'overknead': 1,  
'readthis': 1,  
'bikefirst': 1,  
'upright': 27,  
'wormnecco': 1,  
'yourramun': 1,  
'coupl': 2404,  
'charit': 15,  
'thankssgave': 1,  
'thenquit': 1,  
'bought': 1,  
'tag': 136,  
'againonc': 3,  
'vitamineveryon': 1,  
'listsighsshockthese': 1,  
'dark': 3020,  
'toysrus': 1,  
'fatteningso': 1,  
'internetorder': 1,  
'blant': 1,  
'thhey': 1,  
'beatthre': 1,  
'herefound': 2,  
'munchkinread': 1,  
'teaagre': 1,  
'orderive': 1,
```



```
'timeferment': 1,  
'alohave': 3,  
'mealit': 3,  
'metabisulfit': 6,  
'afternoonmichael': 1,  
'yogurtkefir': 1,  
'hindenberg': 1,  
'granbabi': 1,  
'teaingredi': 1,  
'hardi': 34,  
'likehome': 1,  
'marintri': 1,  
'buildwhen': 1,  
'lookclassic': 1,  
'ricola': 5,  
'plusthese': 2,  
'cheesbest': 1,  
'tierra': 8,  
'hayguiltless': 1,  
'exam': 21,  
'mannercannot': 1,  
'diffeent': 1,  
'treatwow': 1,  
'addbought': 2,  
'absolutey': 1,  
'lean': 205,  
'theseediet': 1,  
'drinksent': 1,  
'mento': 2,  
'regress': 2,  
'twoand': 1,  
'stora': 1,  
'warmmayb': 1,  
'tjs': 4,  
'theoften': 1,  
'specialgreat': 1,  
'smoothivisit': 1,  
'lookneed': 1,  
'occasthorough': 1,
```

```
'heartilihave': 1,  
'freshlik': 1,  
'pst': 1,  
'gettingin': 1,  
'labelfirst': 1,  
'thse': 3,  
'nevella': 1,  
'potatoover': 1,  
'teahad': 6,  
'catthis': 10,  
'interestnot': 1,  
'productveryy': 1,  
'lessd': 1,  
'cheapso': 2,  
'sunrischild': 1,  
'bothertrident': 1,  
'triglyceridbought': 1,  
'sugarkid': 1,  
'horriblthis': 2,  
'inulabrador': 1,  
'holidaythis': 8,  
'bitbuy': 1,  
'calorigotten': 1,  
'spotlet': 1,  
'studentthere': 1,  
'flavourbought': 1,  
'othersorri': 1,  
'lotsalt': 1,  
'bluemountain': 1,  
'disappointedthi': 1,  
'alreadirealli': 1,  
'dick': 24,  
'beanwell': 1,  
'rumplestiskin': 1,  
'dayultim': 1,  
'texturspice': 1,  
'pacehave': 1,  
'anipurchas': 2,  
'aris': 11,
```

```
'flatulencega': 1,  
'withgrate': 1,  
'likesit': 1,  
'outthose': 1,  
'alternthought': 1,  
'moldhave': 1,  
'nightwhen': 2,  
'formade': 2,  
'happenednev': 1,  
'healthirealli': 1,  
'situatdark': 1,  
'lostthey': 1,  
'blazekbulli': 1,  
'moreset': 1,  
'recommendani': 2,  
'eventwo': 1,  
'reasontri': 2,  
'contentjust': 1,  
'madeenah': 1,  
'mixfolger': 1,  
'tukey': 1,  
'watercombin': 1,  
'outerwear': 1,  
'againi': 20,  
'donepleas': 1,  
'effectarrowhead': 1,  
'companiwell': 1,  
'beatjust': 1,  
'decisbeen': 1,  
'craver': 1,  
'herbalearthi': 1,  
'collielab': 2,  
'traveleight': 1,  
'mixturewhich': 1,  
'anymor': 805,  
'specialeveri': 1,  
'amazoncomdog': 1,  
'hunt': 181,  
'toghterit': 1,
```

```
'bellino': 1,  
'uniqueif': 1,  
'courscclear': 1,  
'elswell': 1,  
'amazonfriend': 2,  
'moisturlive': 1,  
'lessdisappoint': 1,  
'nexium': 4,  
'carryng': 1,  
'chartreus': 8,  
'handnever': 2,  
'teir': 1,  
'foodfightgroceri': 1,  
'asda': 1,  
'nubbut': 1,  
'rivalsanyon': 1,  
'couton': 1,  
'blendyourself': 1,  
'winebase': 1,  
'cleardri': 1,  
'wubba': 1,  
'itzaman': 1,  
'peachmango': 3,  
'controlyou': 1,  
'satisfisome': 1,  
'pickiyou': 1,  
'harrar': 3,  
'chilliyet': 1,  
'dogthe': 9,  
'stalest': 1,  
'detoxif': 3,  
'schnere': 2,  
'mandatori': 8,  
'alwayrecent': 1,  
'ozbottl': 1,  
'teaour': 2,  
'andoill': 1,  
'realthank': 1,  
'continthis': 1,
```

```
'loacker': 1,  
'onecare': 1,  
'personyou': 1,  
'fraenkisch': 1,  
'thebegin': 1,  
'octopus': 29,  
'mileafter': 1,  
'candithank': 1,  
'sobat': 1,  
'sacrifici': 1,  
'messfinal': 1,  
'hugest': 3,  
'shipmentit': 2,  
'expectare': 1,  
'surfacwonder': 1,  
'availchees': 1,  
'getwhi': 1,  
'wellahmad': 1,  
'anythlove': 9,  
'applebutt': 1,  
'thisown': 2,  
'saucett': 5,  
'themfar': 3,  
'huski': 38,  
'miscommun': 1,  
'superstit': 1,  
'availablehmmhidmade': 1,  
'picturgreat': 1,  
'freshbare': 1,  
'tastewil': 3,  
'bon': 31,  
'cajol': 6,  
'cokki': 1,  
'cupit': 2,  
'futurafter': 2,  
'themithese': 1,  
'theirfirst': 2,  
'usaand': 1,  
'aupperltri': 1,
```

```
'frichikagre': 1,  
'thoughtri': 6,  
'floursmelltast': 1,  
'tablhard': 1,  
'setfast': 1,  
'stapltwo': 1,  
'expectdog': 2,  
'gummithe': 1,  
'monthwhen': 2,  
'deliciamerican': 1,  
'thatthought': 2,  
'brandsecond': 1,  
'daffodil': 1,  
'schooler': 1,  
'groceriuse': 2,  
'epa': 2,  
'sandwichdid': 1,  
'drainmaxwel': 1,  
'waikato': 2,  
'sryup': 1,  
'aromatthese': 1,  
'crowdtazo': 1,  
'competitthis': 4,  
'bakenoth': 1,  
'proteinlove': 2,  
'goldbox': 7,  
'minerwas': 1,  
'mile': 279,  
'butterthank': 1,  
'originthis': 2,  
'portionwas': 1,  
'insectthis': 1,  
'foreign': 1,  
'themay': 3,  
'recievthis': 2,  
'duperealli': 1,  
'simon': 3,  
'recommendhope': 1,  
'scant': 23,
```

'counteract': 19,
'excelhave': 12,
'monthrealli': 3,
'thankdecid': 2,
'disillusion': 1,
'indig': 2,
'availwas': 1,
'rydermich': 1,
'smoother': 155,
'rotessir': 1,
'nutrientsnot': 1,
'thoughwellthi': 1,
'theuse': 16,
'exceedanoth': 1,
'foodtwo': 1,
'teawhen': 5,
'instantyou': 1,
'grandeur': 1,
'packfine': 1,
'altinba': 1,
'breakfastfound': 2,
'theirour': 1,
'anymorrealli': 4,
'noodljust': 1,
'potaot': 1,
'lifethese': 6,
'printmi': 1,
'earlier': 217,
'muniecthish': 1,
'storewow': 1,
'enjoyexpect': 1,
'localmccormick': 1,
'benzodiazepin': 2,
'programthese': 2,
'weakest': 12,
'unecessaripuppi': 1,
'confectionari': 12,
'goneknow': 2,
'easpurchas': 1,

```
'shotthis': 3,  
'tenia': 1,  
'saucproduct': 2,  
'sneek': 2,  
'bulkour': 1,  
'debilitatingi': 1,  
'sweetenwow': 1,  
'moneyfamili': 1,  
'barsbut': 1,  
'kos': 1,  
'nicesuper': 1,  
'poundhusband': 1,  
'mustwas': 1,  
'greec': 33,  
'availproduct': 1,  
'beat': 1067,  
'tastfirst': 12,  
'thisfinicki': 2,  
'costpup': 1,  
'alluse': 8,  
'yougrown': 1,  
'bigot': 1,  
'shatilause': 1,  
'repack': 1,  
'marihave': 1,  
'perdaughter': 1,  
'triwonder': 3,  
'coffeewithout': 1,  
'bodyand': 1,  
'experipath': 1,  
'muahahaha': 1,  
'fromveget': 1,  
'farmwas': 1,  
'pickey': 4,  
'toocant': 1,  
'theheard': 1,  
'swellew': 1,  
'grainluv': 1,  
'marketgave': 1,
```



```
'paythese': 4,  
'farro': 14,  
'sticksbranch': 1,  
'diffenc': 2,  
'costlike': 1,  
'afternoonread': 1,  
'vinegarwhat': 1,  
'chuppa': 1,  
'enjoyyear': 3,  
'sellerbunnibig': 1,  
'abbott': 3,  
'themalmond': 1,  
'zabuca': 1,  
'baradd': 1,  
'usetoo': 2,  
'storealthough': 2,  
'jerkithese': 5,  
'herecreami': 1,  
'dairymen': 1,  
'oatblueberri': 1,  
'honeyorder': 1,  
'jacket': 28,  
'impulsethey': 1,  
'cranberriswitch': 1,  
'binghamton': 1,  
...}
```

```
In [66]: import pickle  
with open('freq_dict.pkl','wb') as file:  
    pickle.dump(freq_dict,file)
```

Creating rank list of frequent words upto 5000

```
In [73]: from operator import itemgetter  
sorted_list= []  
for k, v in sorted(freq_dict.items(), key=itemgetter(1),reverse=True):  
    sorted_list.append(k)
```

```
In [74]: sorted_list
```

```
Out[74]: ['the',  
          'and',  
          'this',  
          'for',  
          'that',  
          'with',  
          'you',  
          'have',  
          'but',  
          'are',  
          'not',  
          'they',  
          'was',  
          'like',  
          'tast',  
          'flavor',  
          'them',  
          'these',  
          'good',  
          'tea',  
          'one',  
          'use',  
          'can',  
          'product',  
          'veri',  
          'great',  
          'just',  
          'tri',  
          'all',  
          'from',  
          'love',  
          'make',  
          'has',  
          'when',  
          'get',  
          'more',  
          'other',
```

'will',
'than',
'coffe',
'had',
'out',
'would',
'some',
'buy',
'food',
'onli',
'eat',
'about',
'time',
'your',
'find',
'realli',
'also',
'best',
'much',
'too',
'littl',
'order',
'even',
'amazon',
'becaus',
'drink',
'which',
'were',
'price',
'bag',
'there',
'store',
'been',
'mix',
'what',
'chocol',
'ani',
'better',
'well',

'box',
'sugar',
'now',
'year',
'their',
'after',
'sweet',
'found',
'day',
'dog',
'want',
'then',
'high',
'look',
'our',
'give',
'cup',
'over',
'first',
'add',
'water',
'brand',
'recommend',
'most',
'she',
'made',
'think',
'packag',
'way',
'who',
'treat',
'two',
'nice',
'work',
'mani',
'enjoy',
'sinc',
'favorit',
'need',

'thing',
'know',
'bar',
'keep',
'bit',
'come',
'differ',
'milk',
'could',
'purchas',
'say',
'snack',
'still',
'lot',
'free',
'delici',
'pack',
'ship',
'hot',
'her',
'take',
'never',
'review',
'organ',
'into',
'without',
'perfect',
'wonder',
'fresh',
'everi',
'doe',
'ever',
'befor',
'how',
'ingredi',
'local',
'sauc',
'cook',
'cat',

'few',
'alway',
'easi',
'bought',
'put',
'natur',
'someth',
'stuff',
'seem',
'cooki',
'it',
'oil',
'whole',
'healthi',
'green',
'contain',
'did',
'got',
'enough',
'hard',
'while',
'ad',
'right',
'qualiti',
'rice',
'those',
'same',
'back',
'regular',
'less',
'dri',
'last',
'candi',
'small',
'salt',
'cereal',
'here',
'calori',
'again',

'howev',
'long',
'serv',
'fruit',
'groceri',
'each',
'actual',
'size',
'tasti',
'quick',
'quit',
'feel',
'see',
'far',
'sure',
'old',
'strong',
'excel',
'definit',
'off',
'though',
'both',
'month',
'textur',
'his',
'peopl',
'chip',
'bread',
'juic',
'bottl',
'be',
'help',
'chicken',
'bean',
'problem',
'start',
'anoth',
'big',
'real',

'through',
'down',
'fat',
'smell',
'item',
'butter',
'open',
'bad',
'case',
'soup',
'almost',
'famili',
'blend',
'kid',
'chees',
'may',
'pretti',
'sever',
'happi',
'usual',
'per',
'thought',
'friend',
'low',
'should',
'go',
'top',
'black',
'new',
'bake',
'gluten',
'diet',
'amount',
'anyth',
'avail',
'varieti',
'compani',
'pasta',
'thank',

'worth',
'light',
'recip',
'minut',
'peanut',
'own',
'around',
'expens',
'onc',
'arriv',
'ice',
'kind',
'nut',
'home',
'chew',
'prefer',
'reason',
'full',
'protein',
'where',
'week',
'health',
'morn',
'half',
'especi',
'abl',
'came',
'dark',
'syrup',
'meal',
'carri',
'receiv',
'until',
'powder',
'probabl',
'white',
'spice',
'save',
'cost',

'expect',
'bitter',
'gift',
'him',
'leav',
'fill',
'piec',
'brew',
'pleas',
'honey',
'might',
'said',
'breakfast',
'such',
'hand',
'roast',
'star',
'away',
'vanilla',
'corn',
'call',
'fact',
'rich',
'place',
'larg',
'compar',
'extra',
'absolut',
'cream',
'three',
'instead',
'live',
'whi',
'coconut',
'sweeten',
'hope',
'cake',
'noth',
'read',

'wheat',
'type',
'disappoint',
'red',
'surpris',
'least',
'let',
'season',
'coupl',
'market',
'cracker',
'slight',
'wish',
'care',
'satisfi',
'smooth',
'ago',
'fine',
'decid',
'husband',
'meat',
'jar',
'end',
'turn',
'soda',
'second',
'addit',
'yet',
'next',
'deal',
'longer',
'pepper',
'chang',
'plus',
'stop',
'although',
'offer',
'run',
'includ',

'color',
'tell',
'went',
'list',
'must',
'sell',
'anyon',
'money',
'either',
'person',
'packet',
'cherri',
'spici',
'altern',
'fiber',
'noodl',
'orang',
'dish',
'gave',
'almond',
'glad',
'mayb',
'salad',
'side',
'appl',
'oliv',
'part',
'hous',
'sometim',
'amaz',
'stick',
'notic',
'etc',
'fast',
'soft',
'version',
'heat',
'seed',
'cheaper',

'past',
'soy',
'cold',
'everyth',
'origin',
'believ',
'fan',
'everyon',
'cut',
'els',
'rather',
'mouth',
'pay',
'pod',
'choic',
'myself',
'crunchi',
'flour',
'stock',
'hour',
'conveni',
'brown',
'oatmeal',
'experi',
'took',
'tomato',
'gum',
'potato',
'bowl',
'weight',
'ounc',
'special',
'lemon',
'close',
'often',
'popcorn',
'son',
'nutrit',
'direct',

'espresso',
'cinnamon',
'exact',
'total',
'energi',
'egg',
'clean',
'easili',
'pound',
'valu',
'grain',
'ask',
'prepar',
'mean',
'consist',
'granola',
'plain',
'goe',
'onlin',
'base',
'normal',
'feed',
'strawberri',
'switch',
'artifici',
'set',
'machin',
'result',
'vitamin',
'caffein',
'life',
'mild',
'becom',
'combin',
'near',
'pot',
'recent',
'particular',
'cours',

'bite',
'microwav',
'simpli',
'pop',
'cannot',
'stay',
'final',
'label',
'bodi',
'instant',
'complet',
'replac',
'carb',
'beef',
'date',
'ground',
'salti',
'similar',
'wait',
'certain',
'sodium',
'dure',
'babi',
'process',
'night',
'search',
'teeth',
'substitut',
'between',
'plastic',
'pleasant',
'lunch',
'suggest',
'aroma',
'ginger',
'consid',
'continu',
'shop',
'line',

'bring',
'cocoa',
'super',
'effect',
'benefit',
'name',
'check',
'content',
'fair',
'pick',
'four',
'daughter',
'veget',
'except',
'servic',
'chewi',
'bulk',
'gram',
'finish',
'deliv',
'smaller',
'pancak',
'chai',
'extrem',
'rate',
'oat',
'daili',
'along',
'individu',
'pure',
'addict',
'guess',
'rememb',
'state',
'fish',
'left',
'point',
'eaten',
'provid',

'sit',
'yummi',
'pill',
'berri',
'christma',
'note',
'delight',
'entir',
'overal',
'given',
'formula',
'decaf',
'custom',
'singl',
'gummi',
'itself',
'someon',
'follow',
'jerki',
'wife',
'acid',
'area',
'glass',
'alreadi',
'carbon',
'insid',
'raw',
'deliveri',
'pet',
'huge',
'interest',
'fantast',
'coat',
'world',
'mint',
'thick',
'bold',
'later',
'sold',

'french',
'pour',
'supermarket',
'idea',
'aftertast',
'mine',
'truli',
'refresh',
'licoric',
'wrong',
'discov',
'chili',
'watch',
'toy',
'roll',
'throw',
'cover',
'import',
'beverag',
'garlic',
'dinner',
'larger',
'boil',
'true',
'due',
'hold',
'creami',
'share',
'sourc',
'liquid',
'consum',
'ate',
'saw',
'hit',
'togeth',
'allergi',
'rest',
'caus',
'wast',

'hint',
'dress',
'preserv',
'maker',
'restaur',
'yes',
'issu',
'general',
'miss',
'mind',
'balanc',
'five',
'produc',
'clear',
'break',
'gone',
'slice',
'sour',
'sale',
'herb',
'difficult',
'sent',
'fun',
'told',
'suppos',
'lower',
'melt',
'mention',
'warm',
'loos',
'simpl',
'done',
'within',
'drop',
'return',
'worri',
'stomach',
'crunch',
'unfortun',

'veggi',
'awesom',
'plant',
'option',
'dip',
'bottom',
'beat',
'anyway',
'wrap',
'excit',
'crave',
'amazoncom',
'impress',
'healthier',
'tradit',
'plan',
'thin',
'batch',
'starbuck',
'easier',
'allow',
'picki',
'seen',
'unlik',
'onion',
'level',
'anywher',
'shape',
'bone',
'under',
'vet',
'raisin',
'twice',
'banana',
'grow',
'fit',
'figur',
'drinker',
'immedi',

'avoid',
'fri',
'beauti',
'soon',
'yogurt',
'stir',
'happen',
'busi',
'test',
'grey',
'opinion',
'kitchen',
'diabet',
'yourself',
'possibl',
'tuna',
'toast',
'italian',
'websit',
'show',
'crisp',
'unless',
'number',
'sort',
'none',
'chemic',
'tend',
'blueberri',
'cool',
'form',
'requir',
'seal',
'hate',
'moist',
'sandwich',
'refriger',
'children',
'remind',
'skin',

'concern',
'do',
'vinegar',
'send',
'sampl',
'uniqu',
'digest',
'six',
'matter',
'browni',
'homemad',
'shake',
'today',
'perhap',
'pie',
'stand',
'tin',
'cheap',
'mapl',
'site',
'basic',
'medium',
'manufactur',
'agre',
'wine',
'blue',
'describ',
'subscrib',
'short',
'trip',
'blood',
'earl',
'touch',
'condit',
'suppli',
'flower',
'keurig',
'lover',
'main',

'shipment',
'tini',
'extract',
'trap',
'standard',
'sprinkl',
'typic',
'previous',
'stevia',
'biscuit',
'gourmet',
'steep',
'tart',
'appreci',
'splenda',
'expir',
'hook',
'broken',
'improv',
'realiz',
'muffin',
'readi',
'american',
'bland',
'concentr',
'instruct',
'complaint',
'travel',
'mill',
'anymor',
'move',
'imagin',
'overpow',
'raspberri',
'tree',
'bear',
'higher',
'remov',
'word',

'somewhat',
'research',
'quantiti',
'portion',
'reduc',
'visit',
'dessert',
'mom',
'whatev',
'offic',
'incred',
'caramel',
'shelf',
'doubl',
'english',
'count',
'creat',
'pan',
'pocket',
'otherwis',
'seller',
'learn',
'straight',
'sound',
'pizza',
'leaf',
'stale',
'decent',
'frozen',
'subtl',
'broth',
'sea',
'tablespoon',
'heavi',
'thrill',
'spoon',
'grape',
'waffl',
'kept',

'spread',
'claim',
'lose',
'senseo',
'summer',
'cashew',
'pictur',
'weak',
'brought',
'ball',
'forward',
'dollar',
'salmon',
'parti',
'control',
'system',
'chunk',
'train',
'kick',
'mess',
'curri',
'serious',
'flake',
'puppi',
'spend',
'across',
'mustard',
'terribl',
'vegan',
'increas',
'knew',
'felt',
'appear',
'charg',
'inform',
'various',
'pricey',
'chanc',
'chop',

'plenti',
'lack',
'sick',
'hazelnut',
'room',
'outsid',
'stronger',
'teaspoon',
'alon',
'known',
'age',
'afternoon',
'herbal',
'relat',
'fall',
'heart',
'heard',
'freez',
'yeast',
'present',
'play',
'job',
'troubl',
'vegetarian',
'comment',
'paper',
'descript',
'crust',
'celiac',
'power',
'style',
'smoke',
'pouch',
'burn',
'jelli',
'whether',
'grill',
'premium',
'late',

```
'sensit',  
'prompt',  
'fridg',  
'mother',  
'crazi',  
'pretzel',  
'book',  
'school',  
'pass',  
'crispi',  
'everyday',  
'paid',  
'indian',  
'aw',  
'understand',  
'nutti',  
'child',  
'grind',  
'safe',  
'quinoa',  
'introduc',  
'sticki',  
'allerg',  
'supplement',  
'agav',  
'remain',  
'equal',  
...]
```

```
In [83]: Data_x[1]
```

```
Out[83]: 'can rememb see the show when air televis year ago when was child siste  
r later bought the which have this day thirti somethingi use this seri  
book song when did student teach for preschool turn the whole school no  
w purchas along with the book for children the tradit live'
```

```
In [91]: top_words= 5000  
sorted_list= sorted_list[:5000]
```

Transforming Sentences of words to sequence of rank number of words

```
In [92]: column=[]  
         for sent in Data_x:  
             lis=[]  
             for word in sent.split():  
                 if word in sorted_list:  
                     lis.append(word)  
             column.append(' '.join(lis))
```

```
In [94]: with open('column.pkl','wb') as file:  
         pickle.dump(column,file)
```

```
In [97]: final_x=[]  
         for sent in Data_x:  
             lis=[]  
             for word in sent.split():  
                 if word in sorted_list:  
                     lis.append(sorted_list.index(word)+1)  
             final_x.append(lis)
```

```
In [99]: Xtest= final_x[:30000]  
         Ytest= Data_y[:30000]  
         Xtrain= final_x[30000:]  
         Ytrain= Data_y[30000:]
```

```
In [105]: print(Xtrain[1])
```

```
[8, 216, 209, 1980, 106, 8, 368, 25, 3960, 528, 3217, 3217, 721, 17, 48  
8, 1101, 2, 583, 127, 5, 1071, 474, 5, 11, 697, 8, 47, 28, 1, 557, 16,  
3217, 3217, 9, 322, 1, 31, 89, 99, 12, 10, 156, 622, 42, 91, 151, 69, 5  
41, 35, 1134, 17, 61, 5, 8, 352, 1, 69]
```

Applying LSTM models

Using TensorFlow backend.

[illegible]

```

0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 8 216 209
1980 106 8 368 25 3960 528 3217 3217 721 17 488 1101 2
583 127 5 1071 474 5 11 697 8 47 28 1 557 16
3217 3217 9 322 1 31 89 99 12 10 156 622 42 91
151 69 541 35 1134 17 61 5 8 352 1 69]

```

```
In [114]: from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```

[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 9021134197561777072
, name: "/device:GPU:0"
 device_type: "GPU"
 memory_limit: 3206820659
 locality {
   bus_id: 1
   links {
 }
 }
 incarnation: 17023240268385776602
 physical_device_desc: "device: 0, name: GeForce 940MX, pci bus id: 000
0:01:00.0, compute capability: 5.0"
]

```

```
In [115]: import numpy
from keras.datasets import imdb
```

```

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM, Dropout
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)

```

Model 1 same as of IMDB

```

In [112]: # create the model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words+1, embedding_vecor_length, input_length=m
ax_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['a
ccuracy'])
print(model.summary())

```

WARNING:tensorflow:From C:\Anaconda3\lib\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version. Instructions for updating:
Colocations handled automatically by placer.

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 600, 32)	160032
lstm_1 (LSTM)	(None, 100)	53200
dense_1 (Dense)	(None, 1)	101
Total params: 213,333		
Trainable params: 213,333		
Non-trainable params: 0		

None

```
In [121]: history=model.fit(Xtrain, Ytrain, nb_epoch=10, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(Xtest, Ytest, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

WARNING:tensorflow:From C:\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.

C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: The `nb_epoch` argument in `fit` has been renamed `epochs`.
"""Entry point for launching an IPython kernel.

```
Epoch 1/10
70000/70000 [=====] - 1607s 23ms/step - loss:
0.2394 - acc: 0.9088
Epoch 2/10
70000/70000 [=====] - 1620s 23ms/step - loss:
0.1763 - acc: 0.9321
Epoch 3/10
70000/70000 [=====] - 1715s 24ms/step - loss:
0.1579 - acc: 0.9392
Epoch 4/10
70000/70000 [=====] - 1468s 21ms/step - loss:
0.1422 - acc: 0.9458
Epoch 5/10
70000/70000 [=====] - 1459s 21ms/step - loss:
0.1260 - acc: 0.9525
Epoch 6/10
70000/70000 [=====] - 1518s 22ms/step - loss:
0.1114 - acc: 0.9583
Epoch 7/10
70000/70000 [=====] - 1522s 22ms/step - loss:
0.0993 - acc: 0.9631
Epoch 8/10
```



```

70000/70000 [=====] - 1554s 22ms/step - loss:
0.0873 - acc: 0.9674
Epoch 9/10
70000/70000 [=====] - 1459s 21ms/step - loss:
0.0784 - acc: 0.9716
Epoch 10/10
70000/70000 [=====] - 1445s 21ms/step - loss:
0.0668 - acc: 0.9756
Accuracy: 92.31%

```

Model 2

```

In [120]: # create the model
embedding_vecor_length = 32
model2 = Sequential()
model2.add(Embedding(top_words+1, embedding_vecor_length, input_length=
max_review_length))
model2.add(LSTM(100,return_sequences=True))
model2.add(Dropout(0.25))
model2.add(LSTM(80))
model2.add(Dropout(0.5))
model2.add(Dense(1, activation='sigmoid'))
model2.compile(loss='binary_crossentropy', optimizer='adam', metrics=[
'accuracy'])
print(model2.summary())

```

Layer (type)	Output Shape	Param #
embedding_6 (Embedding)	(None, 600, 32)	160032
lstm_9 (LSTM)	(None, 600, 100)	53200
dropout_4 (Dropout)	(None, 600, 100)	0
lstm_10 (LSTM)	(None, 80)	57920
dropout_5 (Dropout)	(None, 80)	0

dense_2 (Dense)	(None, 1)	81
-----------------	-----------	----

Total params: 271,233
 Trainable params: 271,233
 Non-trainable params: 0

None

```
In [122]: history2= model2.fit(Xtrain, Ytrain, nb_epoch=10, batch_size=64)-# Final evaluation of the model
scores = model2.evaluate(Xtest, Ytest, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: The `nb_epoch` argument in `fit` has been renamed `epochs`.

"""Entry point for launching an IPython kernel.

```
Epoch 1/10
70000/70000 [=====] - 2961s 42ms/step - loss:
0.2402 - acc: 0.9088
Epoch 2/10
70000/70000 [=====] - 2958s 42ms/step - loss:
0.1802 - acc: 0.9316
Epoch 3/10
70000/70000 [=====] - 2961s 42ms/step - loss:
0.1613 - acc: 0.9386
Epoch 4/10
70000/70000 [=====] - 2958s 42ms/step - loss:
0.1429 - acc: 0.9459
Epoch 5/10
70000/70000 [=====] - 2964s 42ms/step - loss:
0.1254 - acc: 0.9528
Epoch 6/10
70000/70000 [=====] - 3007s 43ms/step - loss:
0.1108 - acc: 0.9592
Epoch 7/10
70000/70000 [=====] - 2996s 43ms/step - loss:
0.0970 - acc: 0.9653
Epoch 8/10
```

```

70000/70000 [=====] - 3001s 43ms/step - loss:
0.0839 - acc: 0.9706
Epoch 9/10
70000/70000 [=====] - 3001s 43ms/step - loss:
0.0764 - acc: 0.9736
Epoch 10/10
70000/70000 [=====] - 3011s 43ms/step - loss:
0.0667 - acc: 0.9778
Accuracy: 92.96%

```

```

In [123]: with open('model.pkl','wb') as file:
          pickle.dump(model,file)
          with open('model2.pkl','wb') as file:
            pickle.dump(model2,file)

```

Testing our model on self made review sentence

```

In [165]: # making a function which convert sentence to required vectorized format
           that will feed well in model

def cleanhtml(sentence): #substitute expression contained in <> with '
'
    cleaned= re.sub(re.compile('<.*?>'),' ',sentence)
    return cleaned
#function for removing punctuations chars
def cleanpunc(sentence):
    cleaned= re.sub(r'[?|!|\'|\'|\'|#]',' ',sentence)
    cleaned= re.sub(r'[.,,]|(|\|/|)',' ',sentence)
    return cleaned
snowstem= sno('english')

def predict_this(sentence):
    i=0
    str1=' '
    final_string=[]
    all_positive_words=[] # store words from +ve reviews here
    all_negative_words=[] # store words from -ve reviews here.

```

```

sent= sentence
filtered_sentence=[]
#print(sent);
sent=cleanhtml(sent) # remove HTML tags
for w in sent.split():
    # we have used cleanpunc(w).split(), one more split function here
    # because consider w="abc.def", cleanpunc(w) will return "abc def"
    # if we dont use .split() function then we will be considering
    "abc def"
    # as a single word, but if you use .split() function we will get
    "abc", "def"
    for cleaned_words in cleanpunc(w).split():
        if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
            s=(snowstem.stem(cleaned_words.lower())).encode('utf8')
            filtered_sentence.append(s)
            if(data['Score'].values[i] == 'Positive':
                all_positive_words.append(s)
            if(data['Score'].values[i] == 'Negative':
                all_negative_words.append(s)
            else:
                continue
    str1 = b" ".join(filtered_sentence) #final string of cleaned words
    final_string.append(str1)

    final_string
    for i in final_string:
        final_string=i.decode("utf-8")

    lis=[]
    for word in final_string.split():
        if word in sorted_list:
            lis.append(sorted_list.index(word)+1)

    final_string= lis
    final_string = sequence.pad_sequences([final_string], maxlen=max_review_length)
    print(final_string)

```

```
what= ''
if (round(float(model2.predict(final_string)))==1):
    what= 'Positive'
    acc= round(float(model2.predict(final_string))*100,2)
else:
    what='Negative'
    acc= 100- round(float(model2.predict(final_string))*100,2)
print(what,'review with',acc,'% Accuracy')
```

```
In [174]: sentence= 'food was very bad in taste'
          predict_this(sentence)
```

[illegible]

[illegible]

```
0 46 13 25 240 15]]
Negative review with 97.99 % Accuracy
```

```
In [175]: sentence= 'taste of chocolate was fantastic'
          predict_this(sentence)
```

[illegible]

```

0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 15 73 13 615]]
Positive review with 98.01 % Accuracy

```

```

In [176]: sentence= 'food was medium tasty'
          predict_this(sentence)

```


[illegible]

```

0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
0 0 46 13 800 202]]
Positive review with 97.55 % Accuracy

```

```

In [179]: print(data['Text'][1])
          predict_this(data['Text'][1])

```

```

Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actu
ally small sized unsalted. Not sure if this was an error or if the vend
or intended to represent the product as "Jumbo".
[[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0  0  0  0  0

```

[illegible]

```
2751 11 208 3 13 2452 1 1185 1402 2550 1 24]]  
Negative review with 71.84 % Accuracy
```

Summary

- Accuracy of First model is 92.31%
- Accuracy of Second model is 92.96%