Importing Libraries

In [75]:
```
import pandas as pd
import numpy as np

import re
import string
from nltk.corpus import stopwords
import spacy
from nltk.tokenize import word_tokenize
import gensim
from gensim import corpora

# libraries for visualization
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [76]:
```
#!pip install pyLDAvis
```

In [84]:
```
data=pd.read_csv("K8 Reviews v0.2.csv")
```

In [85]:
```
data.head()
```

Out[85]:

	sentiment	review
0	1	Good but need updates and improvements
1	0	Worst mobile i have bought ever, Battery is dr...
2	1	when I will get my 10% cash back.... its alrea...
3	1	Good
4	0	The worst phone everThey have changed the last...

Tokenizing data and filtering rows which have more than one word

In [86]:
```
data['new_review']=data['review'].map(lambda x : word_tokenize(x))
```

In [87]:
```
data.head()
```

Out[87]:

	sentiment	review	new_review
0	1	Good but need updates and improvements	[Good, but, need, updates, and, improvements]

	sentiment	review	new_review
1	0	Worst mobile i have bought ever, Battery is dr...	[Worst, mobile, i, have, bought, ever, ,, Batt...
2	1	when I will get my 10% cash back.... its alrea...	[when, I, will, get, my, 10, %, cash, back, ...
3	1	Good	[Good]
4	0	The worst phone everThey have changed the last...	[The, worst, phone, everThey, have, changed, t...

In [88]:
```
data=data[data['new_review'].map(lambda x: len(x)) > 1].reset_index(drop=True)
# Keeping records with more than single words
```

In [89]:
```
data.head()
```

Out[89]:

	sentiment	review	new_review
0	1	Good but need updates and improvements	[Good, but, need, updates, and, improvements]
1	0	Worst mobile i have bought ever, Battery is dr...	[Worst, mobile, i, have, bought, ever, ,, Batt...
2	1	when I will get my 10% cash back.... its alrea...	[when, I, will, get, my, 10, %, cash, back, ...
3	0	The worst phone everThey have changed the last...	[The, worst, phone, everThey, have, changed, t...
4	0	Only I'm telling don't buyI'm totally disappoi...	[Only, I, 'm, telling, do, n't, buyI, 'm, tota...

Normalized data

In [90]:
```
stop_words=stopwords.words('english')
def normalize(text):
    text=text.lower()
    textarr=text.split(' ')
    remtext=" ".join(i for i in textarr if (i not in stop_words and len(i) > 3 and i
    return remtext
```

In [91]:
```
data['cleaned_review']=data['review'].apply(normalize)
```

In [92]:
```
data.head()
```

Out[92]:

	sentiment	review	new_review	cleaned_review
0	1	Good but need updates and improvements	[Good, but, need, updates, and, improvements]	good need updates improvements
1	0	Worst mobile i have bought ever, Battery is dr...	[Worst, mobile, i, have, bought, ever, ,, Batt...	worst mobile bought battery draining like back...
2	1	when I will get my 10% cash back.... its alrea...	[when, I, will, get, my, 10, %, cash, back, ...	cash already
3	0	The worst phone everThey have changed the last...	[The, worst, phone, everThey, have, changed, t...	worst phone everthey changed last phone proble...

	sentiment	review	new_review	cleaned_review
4	0	Only I'm telling don't buyI'm totally disappoi...	[Only, I, 'm, telling, do, n't, buyI, 'm, tota...	telling totally disappointedpoor batterypoor c...

In [93]:
```
data.shape
```

Out[93]:
```
(13759, 4)
```

lemmatization and pos tag with filtering nouns and adj

In [94]:
```
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
```

In [95]:
```
def lemmatization(rev,allowed_postags=['NOUN', 'ADJ']):
    output = []
    for sent in rev:
        doc = nlp(sent)
        output.append([token.lemma_ for token in doc if token.pos_ in allowed_postag
    return output
```

In [97]:
```
reviews=data['cleaned_review'].tolist()
```

In [98]:
```
tokenized_reviews=lemmatization(reviews)
```

In [99]:
```
#data['cleaned_review']=data['cleaned_review'].apply(lemmatization)
```

In [100…
```
print(tokenized_reviews[0:5])
```

```
[['good', 'need', 'improvement'], ['bad', 'mobile', 'battery', 'backup', 'hour', 'in
ternet', 'mobile', 'idle', 'big', 'amazon', 'lenove', 'full', 'battery', 'booster',
'charger', 'least', 'hour', 'full', 'else', 'regret'], ['cash'], ['bad', 'phone', 'l
ast', 'phone', 'problem', 'amazon', 'phone', 'disappointing', 'amazon'], ['disappoin
tedpoor', 'batterypoor', 'camerawaste', 'money']]
```

LDA MODEL CREATION

In [101…
```
dictionary=corpora.Dictionary(tokenized_reviews)
```

In [102…
```
print(dictionary)
```

```
Dictionary(5512 unique tokens: ['good', 'improvement', 'need', 'amazon', 'backu
p']...)
```

In [103…
```
matrix=[dictionary.doc2bow(rev) for rev in tokenized_reviews]
```

In [104…
```
print(matrix[0:5])
```

```
[[(0, 1), (1, 1), (2, 1)], [(3, 1), (4, 1), (5, 1), (6, 2), (7, 1), (8, 1), (9, 1),
(10, 1), (11, 2), (12, 2), (13, 1), (14, 1), (15, 1), (16, 1), (17, 2), (18, 1)],
[(19, 1)], [(3, 2), (5, 1), (20, 1), (21, 1), (22, 3), (23, 1)], [(24, 1), (25, 1),
(26, 1), (27, 1)]]
```

In [105…

```
LDA = gensim.models.ldamodel.LdaModel
lda_model = LDA(corpus=matrix, id2word=dictionary, num_topics=12, random_state=100,
                chunksize=2000, passes=50,iterations=100)
```

In [106…

```
lda_model.print_topics()
```

Out[106…

```
[(0,
  '0.093*"feature" + 0.060*"option" + 0.049*"work" + 0.043*"many" + 0.035*"call" +
0.033*"screen" + 0.031*"app" + 0.029*"video" + 0.025*"datum" + 0.021*"basic"'),
 (1,
  '0.210*"nice" + 0.157*"problem" + 0.155*"heating" + 0.098*"phone" + 0.083*"issue"
+ 0.024*"super" + 0.011*"bluetooth" + 0.011*"headset" + 0.010*"much" + 0.008*"hangin
g"'),
 (2,
  '0.196*"camera" + 0.101*"quality" + 0.057*"battery" + 0.035*"great" + 0.034*"dual"
+ 0.033*"performance" + 0.029*"poor" + 0.027*"front" + 0.020*"depth" + 0.019*"mod
e"'),
 (3,
  '0.099*"note" + 0.081*"speaker" + 0.048*"available" + 0.037*"working" + 0.032*"sou
nd" + 0.031*"experience" + 0.030*"headphone" + 0.029*"function" + 0.027*"killer" +
0.025*"superb"'),
 (4,
  '0.088*"phone" + 0.050*"excellent" + 0.048*"update" + 0.037*"software" + 0.034*"an
droid" + 0.032*"heat" + 0.032*"amazing" + 0.024*"review" + 0.023*"stock" + 0.021*"is
sue"'),
 (5,
  '0.279*"phone" + 0.099*"screen" + 0.036*"glass" + 0.033*"month" + 0.031*"smart" +
0.029*"gorilla" + 0.027*"fine" + 0.019*"perfect" + 0.019*"contact" + 0.014*"warrant
y"'),
 (6,
  '0.086*"phone" + 0.042*"network" + 0.042*"bad" + 0.034*"issue" + 0.033*"time" + 0.
032*"service" + 0.030*"amazon" + 0.026*"problem" + 0.021*"day" + 0.017*"customer"'),
 (7,
  '0.212*"battery" + 0.063*"phone" + 0.038*"hour" + 0.037*"backup" + 0.036*"charger"
+ 0.036*"charge" + 0.032*"turbo" + 0.032*"awesome" + 0.031*"money" + 0.030*"drai
n"'),
 (8,
  '0.091*"waste" + 0.070*"call" + 0.064*"little" + 0.054*"happy" + 0.038*"voice" +
0.037*"volte" + 0.033*"support" + 0.025*"item" + 0.023*"application" + 0.022*"aut
o"'),
 (9,
  '0.451*"good" + 0.138*"phone" + 0.057*"price" + 0.049*"camera" + 0.023*"performanc
e" + 0.022*"overall" + 0.017*"range" + 0.014*"budget" + 0.013*"smartphone" + 0.010
*"thing"'),
 (10,
  '0.357*"mobile" + 0.096*"good" + 0.053*"worth" + 0.022*"device" + 0.019*"sensor" +
0.018*"side" + 0.015*"love" + 0.014*"earphone" + 0.013*"awesome" + 0.012*"quick"'),
 (11,
  '0.310*"product" + 0.053*"amazon" + 0.033*"delivery" + 0.030*"lenovo" + 0.026*"ret
urn" + 0.022*"bad" + 0.018*"defective" + 0.017*"thank" + 0.017*"much" + 0.013*"mone
y"')]
```

In [107…

```
pyLDAvis.enable_notebook()
vis = gensimvis.prepare(lda_model,matrix,dictionary)
```

In [ ]:

```
vis
```

Coherence Score

In [109…

```
from gensim.models.coherencemodel import CoherenceModel
coherence_model_lda = CoherenceModel(model=lda_model, texts=tokenized_reviews, dicti
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Coherence Score:  0.5147637975475261
```

Creating best LDA model aka optimal number of topics

In [110…

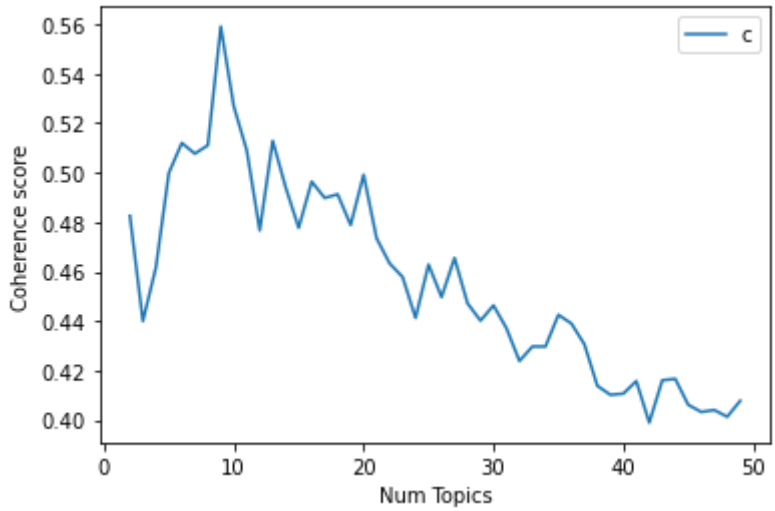
```
def compute_different_coherent_values(dictionary, corpus, texts, limit, start, step)
    coherence_values = []
    model_list = []
    for num_topics in range(start, limit, step):
        model = gensim.models.ldamodel.LdaModel(corpus=corpus, num_topics=num_topics
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=diction
        coherence_values.append(coherencemodel.get_coherence())

    return model_list, coherence_values
```

In [111…

```
model_list, coherence_score=compute_different_coherent_values(dictionary=dictionary,
```

In [113…

```
limit=50; start=2; step=1;
x = range(start, limit, step)
plt.plot(x, coherence_score)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()# Print the coherence scores
```



In [114…

```
for m, cs in zip(x, coherence_score):
    print("Num Topics =", m, " has Coherence Value of", round(cs, 4))
```

```
Num Topics = 2  has Coherence Value of 0.4825
Num Topics = 3  has Coherence Value of 0.44
Num Topics = 4  has Coherence Value of 0.4616
Num Topics = 5  has Coherence Value of 0.4999
```

```
Num Topics = 6  has Coherence Value of 0.512
Num Topics = 7  has Coherence Value of 0.5077
Num Topics = 8  has Coherence Value of 0.511
Num Topics = 9  has Coherence Value of 0.5591
Num Topics = 10  has Coherence Value of 0.5269
Num Topics = 11  has Coherence Value of 0.509
Num Topics = 12  has Coherence Value of 0.4766
Num Topics = 13  has Coherence Value of 0.5129
Num Topics = 14  has Coherence Value of 0.494
Num Topics = 15  has Coherence Value of 0.4777
Num Topics = 16  has Coherence Value of 0.4964
Num Topics = 17  has Coherence Value of 0.4898
Num Topics = 18  has Coherence Value of 0.4912
Num Topics = 19  has Coherence Value of 0.4788
Num Topics = 20  has Coherence Value of 0.4991
Num Topics = 21  has Coherence Value of 0.4734
Num Topics = 22  has Coherence Value of 0.4633
Num Topics = 23  has Coherence Value of 0.4578
Num Topics = 24  has Coherence Value of 0.4413
Num Topics = 25  has Coherence Value of 0.4628
Num Topics = 26  has Coherence Value of 0.4497
Num Topics = 27  has Coherence Value of 0.4655
Num Topics = 28  has Coherence Value of 0.4472
Num Topics = 29  has Coherence Value of 0.4401
Num Topics = 30  has Coherence Value of 0.4463
Num Topics = 31  has Coherence Value of 0.4371
Num Topics = 32  has Coherence Value of 0.4238
Num Topics = 33  has Coherence Value of 0.4297
Num Topics = 34  has Coherence Value of 0.4297
Num Topics = 35  has Coherence Value of 0.4425
Num Topics = 36  has Coherence Value of 0.439
Num Topics = 37  has Coherence Value of 0.4306
Num Topics = 38  has Coherence Value of 0.4138
Num Topics = 39  has Coherence Value of 0.4102
Num Topics = 40  has Coherence Value of 0.4106
Num Topics = 41  has Coherence Value of 0.4157
Num Topics = 42  has Coherence Value of 0.3989
Num Topics = 43  has Coherence Value of 0.416
Num Topics = 44  has Coherence Value of 0.4166
Num Topics = 45  has Coherence Value of 0.4061
Num Topics = 46  has Coherence Value of 0.4032
Num Topics = 47  has Coherence Value of 0.404
Num Topics = 48  has Coherence Value of 0.4013
Num Topics = 49  has Coherence Value of 0.4077
```

From above result we can say 9 is the optimal topic number for best LDA model

Select the model and print the topics

In [121…

```
optimal_model = model_list[7]
model_topics = optimal_model.show_topics(formatted=False)
optimal_model.print_topics(num_words=10)
```

Out[121…

```
[(0,
  '0.033*"product" + 0.033*"phone" + 0.033*"speaker" + 0.031*"call" + 0.027*"many" +
0.024*"time" + 0.024*"camera" + 0.023*"problem" + 0.017*"screen" + 0.012*"app"'),
 (1,
  '0.107*"good" + 0.104*"nice" + 0.084*"product" + 0.069*"phone" + 0.056*"camera" +
0.040*"price" + 0.030*"great" + 0.023*"feature" + 0.020*"excellent" + 0.019*"batter
y"'),
 (2,
  '0.123*"battery" + 0.053*"camera" + 0.033*"phone" + 0.031*"poor" + 0.028*"backup"
+ 0.022*"mobile" + 0.017*"life" + 0.016*"device" + 0.014*"mode" + 0.013*"update"'),
 (3,
```

```
  '0.066*"phone" + 0.065*"awesome" + 0.029*"good" + 0.023*"glass" + 0.018*"gorilla"
+ 0.017*"product" + 0.017*"love" + 0.016*"headphone" + 0.014*"note" + 0.013*"volum
e"'),
 (4,
  '0.252*"good" + 0.128*"phone" + 0.061*"camera" + 0.050*"battery" + 0.039*"problem"
+ 0.038*"performance" + 0.031*"heating" + 0.025*"quality" + 0.014*"worth" + 0.013*"b
ackup"'),
 (5,
  '0.182*"mobile" + 0.040*"charger" + 0.038*"good" + 0.032*"phone" + 0.029*"charge"
+ 0.027*"turbo" + 0.021*"time" + 0.020*"full" + 0.020*"work" + 0.016*"hour"'),
 (6,
  '0.108*"phone" + 0.054*"bad" + 0.038*"amazon" + 0.026*"product" + 0.025*"service"
+ 0.024*"time" + 0.024*"battery" + 0.021*"month" + 0.017*"return" + 0.016*"proble
m"'),
 (7,
  '0.076*"battery" + 0.075*"heating" + 0.063*"issue" + 0.058*"network" + 0.024*"drai
n" + 0.023*"value" + 0.021*"fast" + 0.021*"problem" + 0.019*"camera" + 0.016*"mone
y"'),
 (8,
  '0.060*"camera" + 0.060*"phone" + 0.056*"quality" + 0.025*"good" + 0.015*"poor" +
0.014*"average" + 0.014*"front" + 0.013*"sound" + 0.012*"much" + 0.012*"well"')]
```

In [122…

```
coherence_model_lda = CoherenceModel(model=optimal_model, texts=tokenized_reviews, d
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Coherence Score:  0.5590789786425435
```

In [123…

```
pyLDAvis.enable_notebook()
vis = gensimvis.prepare(optimal_model,matrix,dictionary)
```

Visualization of topics

In [140…

```
vis
```

Out[140…

Topic Table creation for 9 optimal topics

In [126…

```
topic_table= pd.DataFrame((optimal_model.print_topics()),columns=['Topic_Number','To
#topic_table['Topic_Name'] = ['Camera, Sound','Mixed issues','Heating issue','turbo
#topic_lookup_data = topic_lookup_data[['Topic_Number','Topic_Name','Top_Keywords']]
```

In [129…

```
topic_table['Top_Keywords'] = topic_table.Top_Keywords.str.replace(r'[^a-z]',' ',reg
```

In [131…

```
topic_table.style.set_properties(subset=['Top_Keywords'], **{'width': '300px'})
```

Out[131…

	Topic_Number	Top_Keywords
0	0	['product', 'phone', 'speaker', 'call', 'many', 'time', 'camera', 'problem', 'screen', 'app']
1	1	['good', 'nice', 'product', 'phone', 'camera', 'price', 'great', 'feature', 'excellent', 'battery']
2	2	['battery', 'camera', 'phone', 'poor', 'backup', 'mobile', 'life', 'device', 'mode', 'update']

	Topic_Number	Top_Keywords
3	3	['phone', 'awesome', 'good', 'glass', 'gorilla', 'product', 'love', 'headphone', 'note', 'volume']
4	4	['good', 'phone', 'camera', 'battery', 'problem', 'performance', 'heating', 'quality', 'worth', 'backup']
5	5	['mobile', 'charger', 'good', 'phone', 'charge', 'turbo', 'time', 'full', 'work', 'hour']
6	6	['phone', 'bad', 'amazon', 'product', 'service', 'time', 'battery', 'month', 'return', 'problem']
7	7	['battery', 'heating', 'issue', 'network', 'drain', 'value', 'fast', 'problem', 'camera', 'money']
8	8	['camera', 'phone', 'quality', 'good', 'poor', 'average', 'front', 'sound', 'much', 'well']

In [138...
```
topic_table['Topic Name']=['speaker and camera thing issue','good feedback about pro
```

Final Table with Topic Name along with top 10 keywords for business analysis

In [139...
```
topic_table
```

Out[139...

	Topic_Number	Top_Keywords	Topic Name
0	0	[product, phone, speaker, call, many, time, ca...	speaker and camera thing issue
1	1	[good, nice, product, phone, camera, price, gr...	good feedback about product
2	2	[battery, camera, phone, poor, backup, mobile,...	bad review on phone performance
3	3	[phone, awesome, good, glass, gorilla, product...	about strong display
4	4	[good, phone, camera, battery, problem, perfor...	feature reviews
5	5	[mobile, charger, good, phone, charge, turbo, ...	about charger
6	6	[phone, bad, amazon, product, service, time, b...	bad reviews on service
7	7	[battery, heating, issue, network, drain, valu...	Heating issues
8	8	[camera, phone, quality, good, poor, average, ...	mixed issues

In [ ]:
```
```

In [ ]:
```
```

In [ ]:
```
'''for index,sent in enumerate(ldamodel[doc_term_matrix]):
  topic_num =[]
  topic_details = sorted(sent,key=lambda x: x[1], reverse=True)[:2] # Getting top 2
  topic_num.append(topic_details[0][0]) # Appending top topic
  if len(topic_details) > 1:
    if topic_details[1][1] > 0.35: # Appending second topic only if it has more than
      topic_num.append(topic_details[1][0])
  review_data.loc[index,'Topic_Number'] = ','.join(str(x) for x in sorted(topic_num)
```

In [ ]:

```
'''for index,topic_num in enumerate(review_data.Topic_Number):
  topic_name_list=[]
  for single_topic_num in topic_num.split(','):
    single_topic_num=int(single_topic_num)
    topic_name_list.append(topic_lookup_data.loc\
                           [topic_lookup_data.Topic_Number == single_topic_num,'Topi
  # Extracting topic names from lookup table
  review_data.loc[index,'Topic_Name'] =' & '.join(topic_name_list)'''
```