NLP Project - Topic Analysis of Review Data

1. Normalize case
2. Tokenize (using word_tokenize from NLTK)
3. POS tagging using the NLTK pos tagger
4. For the topic model, we would want to include only nouns
   - First, find out all the POS tags that correspond to nouns
   - Limit the data to only terms with these tags
5. Lemmatize (you want different forms of the terms to be treated as one, don't worry about providing POS tag to lemmatizer for now)
6. Remove stop words and punctuation (if there are any at all after the POS tagging)
7. Create a topic model using LDA on the cleaned up data with 12 topics
   - choose the topic model parameters carefully
   - what is the perplexity of the model?
   - what is the coherence of the model?
8. Analyze the topics, which pairs of topics can be combined?
9. Create topic model using LDA with what you think is the optimal number of topics
   - choose the topic model parameters carefully
   - is the perplexity better now?
   - is the coherence better now?
10. The business finally needs to be able to interpret the topics
    - name each of the identified topics
    - create a table with the topic name and the top 10 terms in each to present to business

In [1]:
```
import warnings
warnings.filterwarnings("ignore")

# Importing the usual utilities
import numpy as np, pandas as pd
import re, random, os, string

from pprint import pprint #pretty print
import matplotlib.pyplot as plt
%matplotlib inline

from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

Task 1. Read the .csv file using Pandas. Take a look at the top few records.

In [2]:
```
reviews0 = pd.read_csv(r"C:\Mary\Training\AI\NLP\Project - Topic Analysis\K8 Reviews v0.2.csv")
reviews0.head()
```

Out[2]:

	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...

Task 2. Normalize casings for the review text and extract the text into a list for easier manipulation.

In [3]:
```
reviews_lower = [sentence.lower() for sentence in reviews0.review.values]
reviews_lower[0]
```

Out[3]:
```
'good but need updates and improvements'
```

Task 3. Tokenize the reviews using NLTKs word_tokenize function.

In [4]:
```
reviews_token = [word_tokenize(sentence) for sentence in reviews_lower]
reviews_token[0]
```

Out[4]:
```
['good', 'but', 'need', 'updates', 'and', 'improvements']
```

Task 4. Perform parts-of-speech tagging on each sentence using the NLTK POS tagger.

In [5]:
```
import nltk
```

In [6]:
```
nltk.pos_tag(reviews_token[0])
```

Out[6]:
```
[('good', 'JJ'),
 ('but', 'CC'),
 ('need', 'VBP'),
 ('updates', 'NNS'),
 ('and', 'CC'),
 ('improvements', 'NNS')]
```

In [7]:
```
reviews_tagged = [nltk.pos_tag(tokens) for tokens in reviews_token]
reviews_tagged[0]
```

Out[7]:
```
[('good', 'JJ'),
 ('but', 'CC'),
 ('need', 'VBP'),
 ('updates', 'NNS'),
 ('and', 'CC'),
 ('improvements', 'NNS')]
```

Task 5. For the topic model, we would want to include only nouns

- First, find out all the POS tags that correspond to nouns
- Limit the data to only terms with these tags

In [8]:
```
# extract the noun tags from the POS tagger tuples
reviews_noun=[]
for sent in reviews_tagged:
    reviews_noun.append([token for token in sent if re.search("NN.*", token[1])])
reviews_noun[0]
```

Out[8]:
```
[('updates', 'NNS'), ('improvements', 'NNS')]
```

Task 6. Lemmatize

1. Different forms of the terms need to be treated as one.
2. No need to provide POS tag to lemmatizer for now.

In [9]:
```
lemm = WordNetLemmatizer()
reviews_lemm=[]
for sent in reviews_noun:
    reviews_lemm.append([lemm.lemmatize(word[0]) for word in sent])
```

In [10]:
```
reviews_lemm[0]
```

Out[10]:
```
['update', 'improvement']
```

Task 7. Remove stop words and punctuation (if there are any at all after the POS tagging)

Use NLTK standard stop word list and the punctuations

In [11]:
```
from string import punctuation
from nltk.corpus import stopwords
stop_nltk = stopwords.words("english")
```

In [12]:
```
stop_updated = stop_nltk + list(punctuation) + ["..."] + [".."]
reviews_sw_removed=[]
for sent in reviews_lemm:
    reviews_sw_removed.append([term for term in sent if term not in stop_updated])
```

In [13]:
```
reviews_lemm[1]
```

Out[13]:
```
['mobile',
 'i',
 'battery',
 'hell',
 'backup',
 'hour',
 'us',
 'idle',
 'discharged.this',
 'lie',
 'amazon',
 'lenove',
 'battery',
 'charger',
 'hour',
 'don']
```

In [14]:
```
reviews_sw_removed[1]
```

Out[14]:
```
['mobile',
 'battery',
 'hell',
 'backup',
 'hour',
 'us',
 'idle',
 'discharged.this',
 'lie',
 'amazon',
 'lenove',
 'battery',
 'charger',
 'hour']
```

Task 8. Create a topic model using LDA on the cleaned up data with 12 topics.

1. Print out the top terms for each topic.
2. What is the coherence of the model with the c_v metric?

In [15]:
```
import gensim
import gensim.corpora as corpora
from gensim.models import CoherenceModel
from gensim.models import ldamodel
```

In [16]:
```
id2word = corpora.Dictionary(reviews_sw_removed)
texts = reviews_sw_removed
corpus = [id2word.doc2bow(text) for text in texts]
```

In [17]:
```
print(corpus[1])
```

```
[(2, 1), (3, 1), (4, 2), (5, 1), (6, 1), (7, 1), (8, 2), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1)]
```

In [18]:
```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=12,
                                            random_state=42,
                                            passes=10,
                                            per_word_topics=True)
```

In [19]:

```
pprint(lda_model.print_topics())
```

```
[(0,
  '0.381*"mobile" + 0.023*"problem" + 0.023*"notification" + 0.017*"heat" + '
  '0.016*"cell" + 0.016*"message" + 0.011*"hang" + 0.011*"rate" + '
  '0.010*"whatsapp" + 0.009*"call"'),
 (1,
  '0.267*"battery" + 0.105*"problem" + 0.055*"backup" + 0.055*"heating" + '
  '0.052*"issue" + 0.037*"performance" + 0.036*"hour" + 0.032*"day" + '
  '0.030*"time" + 0.029*"life"'),
 (2,
  '0.062*"handset" + 0.051*"software" + 0.041*"box" + 0.032*"contact" + '
  '0.030*"update" + 0.026*"set" + 0.023*"star" + 0.023*"option" + 0.022*"item" '
  '+ 0.020*"purchase"'),
 (3,
  '0.080*"phone" + 0.049*"amazon" + 0.044*"service" + 0.030*"lenovo" + '
  '0.030*"day" + 0.029*"issue" + 0.027*"problem" + 0.026*"time" + '
  '0.022*"delivery" + 0.019*"experience"'),
 (4,
  '0.135*"feature" + 0.076*"camera" + 0.048*"mode" + 0.037*"video" + '
  '0.027*"android" + 0.025*"stock" + 0.023*"depth" + 0.019*"gallery" + '
  '0.018*"volta" + 0.017*"thanks"'),
 (5,
  '0.439*"product" + 0.090*"charger" + 0.018*"earphone" + 0.016*"turbo" + '
  '0.016*"buy" + 0.016*"piece" + 0.015*"awesome" + 0.012*"cable" + '
  '0.012*"work" + 0.012*"bill"'),
 (6,
  '0.091*"network" + 0.068*"call" + 0.046*"sim" + 0.036*"hai" + 0.024*"jio" + '
  '0.019*"card" + 0.017*"signal" + 0.017*"issue" + 0.016*"slot" + '
  '0.016*"voice"'),
 (7,
  '0.289*"camera" + 0.192*"quality" + 0.026*"glass" + 0.022*"clarity" + '
  '0.019*"everything" + 0.018*"performance" + 0.017*"picture" + 0.016*"front" '
  '+ 0.015*"sound" + 0.014*"mp"'),
 (8,
  '0.378*"phone" + 0.026*"device" + 0.025*"screen" + 0.021*"issue" + '
  '0.018*"time" + 0.017*"update" + 0.017*"processor" + 0.014*"budget" + '
  '0.012*"lot" + 0.012*"performance"'),
 (9,
  '0.116*"phone" + 0.097*"price" + 0.068*"camera" + 0.038*"range" + '
  '0.037*"display" + 0.029*"heat" + 0.026*"battery" + 0.020*"performance" + '
  '0.018*"bit" + 0.018*"usage"'),
 (10,
```

```
 '0.163*"note" + 0.081*"k8" + 0.063*"lenovo" + 0.052*"speaker" + '
 '0.028*"sound" + 0.023*"dolby" + 0.019*"headphone" + 0.018*"music" + '
 '0.016*"key" + 0.015*"killer"'),
(11,
 '0.176*"money" + 0.082*"waste" + 0.067*"value" + 0.057*"superb" + '
 '0.039*"expectation" + 0.030*"super" + 0.027*"worth" + 0.023*"mark" + '
 '0.012*"draining" + 0.011*"condition"')]
```

In [20]:
```python
# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=reviews_sw_removed, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Coherence Score:  0.5560767730635368
```

Task 9. Analyze the topics through the business lens.

1. Determine which of the topics can be combined.

Looking at the topics and each terms following can be combined -

Topic 5 and 8 possibly talks about 'camera features'
Topic 1, 2 and 10 closely talks about 'battery/heating related issues'
Topic 6 and 11 vaguely talks about 'mobile accessories'

Task 10. Create topic model using LDA with what you think is the optimal number of topics

- is the coherence better now?

As some of the topics can be combined, we can reduce the number of topics to 8. Created models to check 8, 7 and 6 topics to see which one is better.

In [21]:
```
# Build LDA model with 7 topics
lda_model7 = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                           id2word=id2word,
                                           num_topics=7,
                                           random_state=42,
                                           passes=16,
                                           per_word_topics=True)
```

Printing the coherence of the model

In [22]:
```
# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model7, texts=reviews_sw_removed, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('7 Topics LDA Model - Coherence Score: ', coherence_lda)
```

```
7 Topics LDA Model - Coherence Score:  0.5892003938196352
```

In [23]:
```
# Build LDA model with 6 topics
lda_model6 = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                           id2word=id2word,
                                           num_topics=6,
                                           random_state=42,
                                           passes=16,
                                           per_word_topics=True)
```

In [24]:
```
# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model6, texts=reviews_sw_removed, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('6 Topics LDA Model - Coherence Score: ', coherence_lda)
```

```
6 Topics LDA Model - Coherence Score:  0.5564983968659106
```

In [25]:
```
# Build LDA model with 8 topics
lda_model8 = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=8,
                                            random_state=42,
                                            passes=16,
                                            per_word_topics=True)
```

In [26]:
```
# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model8, texts=reviews_sw_removed, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('8 Topics LDA Model - Coherence Score: ', coherence_lda)
```

```
8 Topics LDA Model - Coherence Score:  0.5516555771076241
```

LDA model using 7 topics is giving better Coherence Score

Task 11. The business should be able to interpret the topics.

1. Name each of the identified topics.
2. Create a table with the topic name and the top 10 terms in each to present to the business.

In [27]:
```
x = lda_model7.show_topics(formatted=False)
topics_words = [(tp[0], [wd[0] for wd in tp[1]]) for tp in x]
topics_words_only = [([wd[0] for wd in tp[1]]) for tp in x]
```

In [28]:

```
for topic,words in topics_words:
    print(str(topic)+ "::"+ str(words))
print()
```

```
0::['battery', 'mobile', 'backup', 'hour', 'problem', 'issue', 'life', 'camera', 'heat', 'day']
1::['quality', 'camera', 'speaker', 'sound', 'display', 'music', 'everything', 'dolby', 'clarity', 'sensor']
2::['product', 'money', 'value', 'glass', 'superb', 'box', 'screen', 'earphone', 'gorilla', 'delivery']
3::['phone', 'problem', 'issue', 'lenovo', 'product', 'amazon', 'day', 'service', 'month', 'time']
4::['camera', 'phone', 'price', 'quality', 'feature', 'performance', 'range', 'mode', 'device', 'processor']
5::['note', 'k8', 'call', 'option', 'charger', 'screen', 'feature', 'lenovo', 'cast', 'waste']
6::['phone', 'network', 'sim', 'hai', 'h', 'jio', 'budget', 'call', 'volta', 'card']
```

In [29]:

```
type(topics_words)
```

Out[29]:

```
list
```

Topic Names from terms present

Topic1 = battery related issues

'battery', 'mobile', 'backup', 'hour', 'problem', 'issue', 'life', 'camera', 'heat', 'day'

Topic2 = overall general phone features

'quality', 'camera', 'speaker', 'sound', 'display', 'music', 'everything', 'dolby', 'clarity', 'sensor'

Topic3 = phone performance

'product', 'money', 'value', 'glass', 'superb', 'box', 'screen', 'earphone', 'gorilla', 'delivery'

Topic4 = amazon

'phone', 'problem', 'issue', 'lenovo', 'product', 'amazon', 'day', 'service', 'month', 'time'

Topic5 = camera quality

'camera', 'phone', 'price', 'quality', 'feature', 'performance', 'range', 'mode', 'device', 'processor'

Topic6 = pricing

'note', 'k8', 'call', 'option', 'charger', 'screen', 'feature', 'lenovo', 'cast', 'waste'

Topic7 = Network Service Provider Related Issues

'phone', 'network', 'sim', 'hai', 'h', 'jio', 'budget', 'call', 'volta', 'card'

In [30]:
```
topic_names=['battery related issues', 'overall general phone features', 'phone performance', 'amazon', 'camera qualit
y', 'pricing', 'Network Service Provider Related Issue']
topic_names
```

Out[30]:
```
['battery related issues',
 'overall general phone features',
 'phone performance',
 'amazon',
 'camera quality',
 'pricing',
 'Network Service Provider Related Issue']
```

In [31]:
```
# create a dataframe to store the topic name and words in table format
topic_list_with_names = pd.DataFrame(list(zip(topic_names, topics_words_only)), columns =['TopicName', 'TopicWords'])
```

In [32]:
```
topic_list_with_names
```

Out[32]:

	TopicName	TopicWords
0	battery related issues	[battery, mobile, backup, hour, problem, issue...
1	overall general phone features	[quality, camera, speaker, sound, display, mus...
2	phone performance	[product, money, value, glass, superb, box, sc...
3	amazon	[phone, problem, issue, lenovo, product, amazo...
4	camera quality	[camera, phone, price, quality, feature, perfo...
5	pricing	[note, k8, call, option, charger, screen, feat...
6	Network Service Provider Related Issue	[phone, network, sim, hai, h, jio, budget, cal...

In [ ]: