Topic Analysis of Review Data

October 20, 2020

DESCRIPTION Help a leading mobile brand understand the voice of the customer by analyzing the reviews of their product on Amazon and the topics that customers are talking about. You will perform topic modeling on specific parts of speech. You'll finally interpret the emerging topics.

Problem Statement: A popular mobile phone brand, Lenovo has launched their budget smartphone in the Indian market. The client wants to understand the VOC (voice of the customer) on the product. This will be useful to not just evaluate the current product, but to also get some direction for developing the product pipeline. The client is particularly interested in the different aspects that customers care about. Product reviews by customers on a leading e-commerce site should provide a good view.

Domain: Amazon reviews for a leading phone brand

Analysis to be done: POS tagging, topic modeling using LDA, and topic interpretation

Dataset: 'K8 Reviews v0.2.csv'

Sentiment: The sentiment against the review (4,5 star reviews are positive, 1,2 are negative)

Reviews: The main text of the review

Steps to perform: Discover the topics in the reviews and present it to business in a consumable format. Employ techniques in syntactic processing and topic modeling.

Perform specific cleanup, POS tagging, and restricting to relevant POS tags, then, perform topic modeling using LDA. Finally, give business-friendly names to the topics and make a table for business.

Tasks:

1. Read the .csv file using Pandas. Take a look at the top few records.
2. Normalize casings for the review text and extract the text into a list for easier manipulation.
3. Tokenize the reviews using NLTKs word_tokenize function.
4. Perform parts-of-speech tagging on each sentence using the NLTK POS tagger.
5. For the topic model, we should want to include only nouns.

- Find out all the POS tags that correspond to nouns.
- Limit the data to only terms with these tags.

6. Lemmatize.

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

7. Remove stopwords and punctuation (if there are any).
8. Create a topic model using LDA on the cleaned up data with 12 topics.

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

9. Analyze the topics through the business lens.

- Determine which of the topics can be combined.

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

- What is the coherence of the model?

11. The business should 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 the business.

Environment Setup

```
In [1]: import numpy as np
        import pandas as pd
        import re
        import nltk
        import gensim
        from pprint import pprint

        # Plotting tools
        import pyLDAvis
        import pyLDAvis.gensim
        import matplotlib.pyplot as plt
        %matplotlib inline
```

1. Read the data

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

Understanding the data

```
In [3]: data.head()
```

```
Out[3]:    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...
```

```
In [4]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14675 entries, 0 to 14674
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   sentiment  14675 non-null  int64
 1   review     14675 non-null  object
dtypes: int64(1), object(1)
memory usage: 229.4+ KB
```

2. Normalizing the casing to lower

```
In [5]: data['review_cleaned'] = data['review'].apply(lambda review: review.lower())
```

```
In [6]: data.head()
```

```
Out[6]:    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...

                                              review_cleaned
        0             good but need updates and improvements
        1  worst mobile i have bought ever, battery is dr...
        2  when i will get my 10% cash back... its alrea...
        3                                               good
        4  the worst phone everthey have changed the last...
```

3. Tokenize the reviews using NLTKs word_tokenize function.

```
In [7]: reviews = data['review_cleaned']
        reviews = [nltk.word_tokenize(review) for review in reviews]
```

```
In [8]: len(reviews)
```

Out[8]: 14675

```
In [9]: pprint(reviews[:5], compact=True)
```

```
[['good', 'but', 'need', 'updates', 'and', 'improvements'],
 ['worst', 'mobile', 'i', 'have', 'bought', 'ever', ',', 'battery', 'is',
  'draining', 'like', 'hell', ',', 'backup', 'is', 'only', '6', 'to', '7',
  'hours', 'with', 'internet', 'uses', ',', 'even', 'if', 'i', 'put', 'mobile',
  'idle', 'its', 'getting', 'discharged.this', 'is', 'biggest', 'lie', 'from',
  'amazon', '&', 'lenove', 'which', 'is', 'not', 'at', 'all', 'expected', ',',
  'they', 'are', 'making', 'full', 'by', 'saying', 'that', 'battery', 'is',
  '4000mah', '&', 'booster', 'charger', 'is', 'fake', ',', 'it', 'takes', 'at',
  'least', '4', 'to', '5', 'hours', 'to', 'be', 'fully', 'charged.do', "n't",
  'know', 'how', 'lenovo', 'will', 'survive', 'by', 'making', 'full', 'of',
  'us.please', 'don', ';', 't', 'go', 'for', 'this', 'else', 'you', 'will',
  'regret', 'like', 'me', '.'],
 ['when', 'i', 'will', 'get', 'my', '10', '%', 'cash', 'back', '...', '.',
  'its', 'already', '15', 'january..'],
 ['good'],
 ['the', 'worst', 'phone', 'everthey', 'have', 'changed', 'the', 'last',
  'phone', 'but', 'the', 'problem', 'is', 'still', 'same', 'and', 'the',
  'amazon', 'is', 'not', 'returning', 'the', 'phone', '.highly',
  'disappointing', 'of', 'amazon']]
```

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

```
In [10]: %%time
         review_pos_tags = [nltk.pos_tag(doc) for doc in reviews]
         pprint(review_pos_tags[:5], compact=True)
```

```
[[('good', 'JJ'), ('but', 'CC'), ('need', 'VBP'), ('updates', 'NNS'),
  ('and', 'CC'), ('improvements', 'NNS')],
 [('worst', 'JJS'), ('mobile', 'NN'), ('i', 'NN'), ('have', 'VBP'),
  ('bought', 'VBN'), ('ever', 'RB'), (',', ','), ('battery', 'NN'),
  ('is', 'VBZ'), ('draining', 'VBG'), ('like', 'IN'), ('hell', 'NN'),
  (',', ','), ('backup', 'NN'), ('is', 'VBZ'), ('only', 'RB'), ('6', 'CD'),
  ('to', 'TO'), ('7', 'CD'), ('hours', 'NNS'), ('with', 'IN'),
  ('internet', 'JJ'), ('uses', 'NNS'), (',', ','), ('even', 'RB'), ('if', 'IN'),
  ('i', 'JJ'), ('put', 'VBP'), ('mobile', 'JJ'), ('idle', 'NN'),
  ('its', 'PRP$'), ('getting', 'VBG'), ('discharged.this', 'NN'), ('is', 'VBZ'),
  ('biggest', 'JJS'), ('lie', 'NN'), ('from', 'IN'), ('amazon', 'NN'),
  ('&', 'CC'), ('lenove', 'NN'), ('which', 'WDT'), ('is', 'VBZ'), ('not', 'RB'),
  ('at', 'IN'), ('all', 'DT'), ('expected', 'VBN'), (',', ','), ('they', 'PRP'),
  ('are', 'VBP'), ('making', 'VBG'), ('full', 'JJ'), ('by', 'IN'),
  ('saying', 'VBG'), ('that', 'DT'), ('battery', 'NN'), ('is', 'VBZ'),
  ('4000mah', 'CD'), ('&', 'CC'), ('booster', 'JJR'), ('charger', 'NN'),
  ('is', 'VBZ'), ('fake', 'JJ'), (',', ','), ('it', 'PRP'), ('takes', 'VBZ'),
```

```
 ('at', 'IN'), ('least', 'JJS'), ('4', 'CD'), ('to', 'TO'), ('5', 'CD'),
 ('hours', 'NNS'), ('to', 'TO'), ('be', 'VB'), ('fully', 'RB'),
 ('charged.do', 'VBP'), ("n't", 'RB'), ('know', 'VB'), ('how', 'WRB'),
 ('lenovo', 'JJ'), ('will', 'MD'), ('survive', 'VB'), ('by', 'IN'),
 ('making', 'VBG'), ('full', 'JJ'), ('of', 'IN'), ('us.please', 'JJ'),
 ('don', 'NN'), (';', ':'), ('t', 'CC'), ('go', 'VB'), ('for', 'IN'),
 ('this', 'DT'), ('else', 'JJ'), ('you', 'PRP'), ('will', 'MD'),
 ('regret', 'VB'), ('like', 'IN'), ('me', 'PRP'), ('.', '.')],
[('when', 'WRB'), ('i', 'NN'), ('will', 'MD'), ('get', 'VB'), ('my', 'PRP$'),
 ('10', 'CD'), ('%', 'NN'), ('cash', 'NN'), ('back', 'RB'), ('...', ':'),
 ('.', '.'), ('its', 'PRP$'), ('already', 'RB'), ('15', 'CD'),
 ('january..', 'NN')],
[('good', 'JJ')],
[('the', 'DT'), ('worst', 'JJS'), ('phone', 'NN'), ('everthey', 'NN'),
 ('have', 'VBP'), ('changed', 'VBN'), ('the', 'DT'), ('last', 'JJ'),
 ('phone', 'NN'), ('but', 'CC'), ('the', 'DT'), ('problem', 'NN'),
 ('is', 'VBZ'), ('still', 'RB'), ('same', 'JJ'), ('and', 'CC'), ('the', 'DT'),
 ('amazon', 'NN'), ('is', 'VBZ'), ('not', 'RB'), ('returning', 'VBG'),
 ('the', 'DT'), ('phone', 'NN'), ('.highly', 'RB'), ('disappointing', 'JJ'),
 ('of', 'IN'), ('amazon', 'NN')]]
Wall time: 27.7 s
```

5. Including only nouns for building the Topic Model

```
In [11]: review_nouns = [[token for token, pos in doc if pos.startswith('NN')] for doc in review
         pprint(review_nouns[:5], compact=True)
```

```
[['updates', 'improvements'],
 ['mobile', 'i', 'battery', 'hell', 'backup', 'hours', 'uses', 'idle',
  'discharged.this', 'lie', 'amazon', 'lenove', 'battery', 'charger', 'hours',
  'don'],
 ['i', '%', 'cash', 'january..'], [],
 ['phone', 'everthey', 'phone', 'problem', 'amazon', 'phone', 'amazon']]
```

6. Lemmatize

```
In [12]: %%time
         review_lemmatized = [[nltk.stem.WordNetLemmatizer().lemmatize(token) for token in doc]
         pprint(review_lemmatized[:5],compact=True)
```

```
[['update', 'improvement'],
 ['mobile', 'i', 'battery', 'hell', 'backup', 'hour', 'us', 'idle',
  'discharged.this', 'lie', 'amazon', 'lenove', 'battery', 'charger', 'hour',
  'don'],
 ['i', '%', 'cash', 'january..'], [],
 ['phone', 'everthey', 'phone', 'problem', 'amazon', 'phone', 'amazon']]
Wall time: 3.13 s
```

7. Remove stopwords and punctuation (if there are any).

```
In [13]: %%time
         review_no_sw = [[token for token in doc if token not in nltk.corpus.stopwords.words('en
                          and token.isalpha()]
                         for doc in review_lemmatized]
         pprint(review_no_sw[:5], compact=True)
```

```
[['update', 'improvement'],
 ['mobile', 'battery', 'hell', 'backup', 'hour', 'us', 'idle', 'lie', 'amazon',
  'lenove', 'battery', 'charger', 'hour'],
 ['cash'], [],
 ['phone', 'everthey', 'phone', 'problem', 'amazon', 'phone', 'amazon']]
Wall time: 25.9 s
```

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

```
In [14]: # Build a Dictionary - association word to numeric id
         dictionary = gensim.corpora.Dictionary(review_no_sw)
         # Transform the collection of texts to a numerical form
         corpus = [dictionary.doc2bow(text) for text in review_no_sw]
```

```
In [15]: NUM_TOPICS = 12
         lda_model = gensim.models.LdaModel(corpus=corpus, num_topics=NUM_TOPICS, id2word=dictio
```

8.1 Print out the top terms for each topic.

```
In [16]: print("LDA Model:")

         for idx in range(NUM_TOPICS):
             # Print the first 10 most representative topics
             pprint("Topic #{}: {}".format(idx, lda_model.print_topic(idx, 10)), compact=True)
```

```
LDA Model:
('Topic #0: 0.070*"battery" + 0.043*"phone" + 0.042*"screen" + 0.037*"day" + '
 '0.023*"camera" + 0.019*"ram" + 0.016*"note" + 0.015*"hr" + 0.014*"time" + '
 '0.014*"app"')
('Topic #1: 0.371*"mobile" + 0.046*"glass" + 0.040*"buy" + 0.032*"phone" + '
 '0.025*"gorilla" + 0.017*"purchase" + 0.016*"star" + 0.013*"mi" + '
 '0.012*"jata" + 0.011*"rating"')
('Topic #2: 0.087*"issue" + 0.083*"phone" + 0.056*"network" + 0.025*"sim" + '
 '0.024*"note" + 0.024*"call" + 0.023*"time" + 0.023*"battery" + '
 '0.017*"problem" + 0.015*"jio"')
('Topic #3: 0.210*"product" + 0.095*"phone" + 0.053*"service" + 0.042*"lenovo" '
 '+ 0.025*"amazon" + 0.018*"customer" + 0.017*"time" + 0.016*"please" + '
 '0.015*"center" + 0.015*"day"')
('Topic #4: 0.081*"battery" + 0.046*"device" + 0.039*"hour" + 0.028*"charging" '
 '+ 0.025*"drain" + 0.025*"camera" + 0.025*"day" + 0.024*"amazon" + '
```

```
 '0.023*"speaker" + 0.022*"problem"')
('Topic #5: 0.144*"camera" + 0.050*"phone" + 0.027*"mode" + 0.025*"hai" + '
 '0.019*"quality" + 0.017*"depth" + 0.017*"battery" + 0.017*"charger" + '
 '0.015*"photo" + 0.015*"feature"')
('Topic #6: 0.167*"battery" + 0.088*"phone" + 0.071*"backup" + 0.062*"heating" '
 '+ 0.060*"heat" + 0.058*"problem" + 0.048*"issue" + 0.046*"time" + '
 '0.028*"delivery" + 0.017*"thanks"')
('Topic #7: 0.074*"phone" + 0.062*"superb" + 0.029*"work" + 0.025*"earphone" + '
 '0.024*"smartphone" + 0.023*"user" + 0.022*"awesome" + 0.020*"contact" + '
 '0.016*"option" + 0.016*"feature"')
('Topic #8: 0.128*"camera" + 0.123*"quality" + 0.062*"phone" + 0.040*"display" '
 '+ 0.034*"battery" + 0.028*"h" + 0.023*"problem" + 0.018*"budget" + '
 '0.016*"performance" + 0.016*"picture"')
('Topic #9: 0.176*"price" + 0.121*"phone" + 0.066*"range" + 0.051*"feature" + '
 '0.020*"excellent" + 0.019*"set" + 0.019*"camera" + 0.017*"system" + '
 '0.015*"function" + 0.014*"option"')
('Topic #10: 0.113*"money" + 0.079*"note" + 0.052*"waste" + 0.048*"phone" + '
 '0.047*"value" + 0.024*"product" + 0.019*"dolby" + 0.017*"speed" + '
 '0.016*"atmos" + 0.015*"everything"')
('Topic #11: 0.293*"phone" + 0.054*"problem" + 0.046*"performance" + '
 '0.034*"battery" + 0.030*"camera" + 0.026*"month" + 0.018*"day" + '
 '0.012*"look" + 0.012*"bit" + 0.011*"super"')
```

Visualising the topic model as per problem statement

In [17]:
```python
pyLDAvis.enable_notebook()

vis = pyLDAvis.gensim.prepare(lda_model, corpus, dictionary)
vis
```

Out[17]:
```
PreparedData(topic_coordinates=              x         y  topics  cluster       Freq
topic
2      0.062669 -0.085116       1        1  11.862483
0      0.096116  0.026158       2        1  11.853811
5      0.053151  0.190229       3        1  10.869544
3     -0.013538 -0.166455       4        1  10.680003
11     0.041287  0.002238       5        1   9.842621
8      0.082603  0.140830       6        1   9.113233
4      0.133813 -0.090946       7        1   8.402766
6      0.157541 -0.061287       8        1   7.171015
10    -0.082527 -0.065386       9        1   6.002136
9     -0.137876  0.105236      10        1   5.432816
7     -0.113252  0.065543      11        1   4.400358
1     -0.279987 -0.061043      12        1   4.369219, topic_info=        Term
11     mobile  1666.000000  1666.000000  Default  30.0000  30.0000
47    product  2176.000000  2176.000000  Default  29.0000  29.0000
63      price   940.000000   940.000000  Default  28.0000  28.0000
```

```
4    battery  3159.000000  3159.000000  Default  27.0000  27.0000
15     phone  6989.000000  6989.000000  Default  26.0000  26.0000
..       ...          ...          ...      ...      ...      ...
55    camera    32.057770  3218.902832  Topic12  -4.6954  -1.4787
143     cost    15.872128    85.740356  Topic12  -5.3984   1.4438
47   product    19.431063  2176.538330  Topic12  -5.1961  -1.5880
48     range    16.938084   365.143921  Topic12  -5.3334   0.0599
85       day    16.587637   940.727966  Topic12  -5.3543  -0.9074

[741 rows x 6 columns], token_table=      Topic      Freq       Term
term
1328      5  0.955870     access
332       2  0.969151  accessory
332      10  0.024850  accessory
912       2  0.097246    account
912       7  0.875212    account
...     ...       ...        ...
26        9  0.047411       year
902       8  0.883449        yes
1024      3  0.965699  yesterday
1024      5  0.029264  yesterday
1770      3  0.963135       zoom

[1904 rows x 3 columns], R=30, lambda_step=0.01, plot_opts={'xlab': 'PC1', 'ylab': 'PC2
```

8.2 coherence of the model with the c_v metric

```
In [18]: # Compute Coherence Score
         coherence_model_lda = gensim.models.CoherenceModel(model=lda_model, texts=review_no_sw,
                                                             dictionary=dictionary, coherence='c_

         coherence_lda = coherence_model_lda.get_coherence()
         print('Coherence Score: ', coherence_lda)
```

```
Coherence Score:  0.49850343048731377
```

9. Analyze the topics through the business lens - Determine which of the topics can be combined Now since the problem at hand, demands identifying the different aspects of the budget smartphone launched in the market, it is imperative that the tokens (words in the customer reviews) with length less than 3 characters may be removed. That way topic model focuses on the variety of aspects like 'camera', 'battery'. Also removing tokens for length less than three would clean the dataset from low significant tokens.

```
In [19]: #custom_stop_words = ['problem', 'issue']

         review_no_custom_sw = [[token for token in doc if len(token) > 3] for doc in review_no_

         pprint(review_no_custom_sw[:5], compact=True)
```

```
[['update', 'improvement'],
 ['mobile', 'battery', 'hell', 'backup', 'hour', 'idle', 'amazon', 'lenove',
  'battery', 'charger', 'hour'],
 ['cash'], [],
 ['phone', 'everthey', 'phone', 'problem', 'amazon', 'phone', 'amazon']]
```

Building up the topic model after removing tokens of length < 3

In [20]:
```
%%time
NUM_TOPICS = 12
# Build a Dictionary - association word to numeric id
dictionary = gensim.corpora.Dictionary(review_no_custom_sw)

# Transform the collection of texts to a numerical form
corpus = [dictionary.doc2bow(text) for text in review_no_custom_sw]

#Creating the LDA model
lda_model = gensim.models.LdaModel(corpus=corpus, num_topics=NUM_TOPICS, id2word=dictio

#Calculating the cohenrence score
coherence_model_lda = gensim.models.CoherenceModel(model=lda_model, texts=review_no_cus
                                                    dictionary=dictionary, coherence='c_

#Collecting the score in tuple
coherence_lda = coherence_model_lda.get_coherence()

print('Coherence Score after removing custom stop words: ', coherence_lda)
```

```
Coherence Score after removing custom stop words:  0.5031360323267438
Wall time: 12 s
```

There is only a marginal change in the cv score, hence we need to search the parameter space for number of topics for the optimal cv score.

10. Create topic model using LDA with the optimal number of topics

Gensim Model for finding the optimal number of topics

In [21]:
```
%%time
NUM_TOPICS = range(3,21)
coherence_lda_scores = []
best_coherence_score = 0
best_lda_model = None

for topic in NUM_TOPICS:
    #Creating the LDA model
```