Natural Language Processing (NLP)

Course-End Project - Solution



**Topic Analysis of Review Data**

**Objective:** 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

**Content:**

Dataset: ‘K8 Reviews v0.2.csv’

Columns:

**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:**

Your goal will be to discover the topics in the reviews and present it to business in a consumable format. As a part of this solution development, you will employ techniques in syntactic processing and topic modeling learnt.

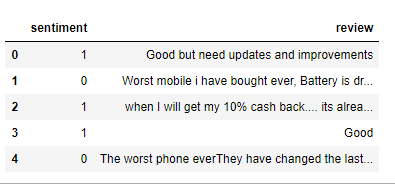
You’ll need to perform specific cleanup, POS tagging, and restricting to relevant POS tags, then, perform topic modeling using LDA. Finally, you will be interpreting and naming the topics with business-friendly names and make a table for business.

**Tasks:**

1. Using Pandas, read in the .csv file. Take a look at the top few records.

reviews0 = pd.read\_csv("K8 Reviews v0.2.csv")

reviews0.head()



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

reviews\_lower = [sent.lower() for sent in reviews0.review.values]

reviews\_lower[0]

****

1. Tokenize the reviews using NLTKs word\_tokenize function.

reviews\_token = [word\_tokenize(sent) for sent in reviews\_lower]

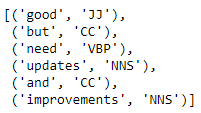
reviews\_token[0]



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

For a single sentence:

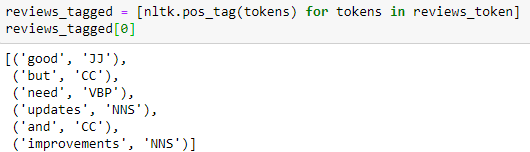
nltk.pos\_tag(reviews\_token[0])



Now do this for all the sentences using list comprehension:

reviews\_tagged = [nltk.pos\_tag(tokens) for tokens in reviews\_token]

reviews\_tagged[0]

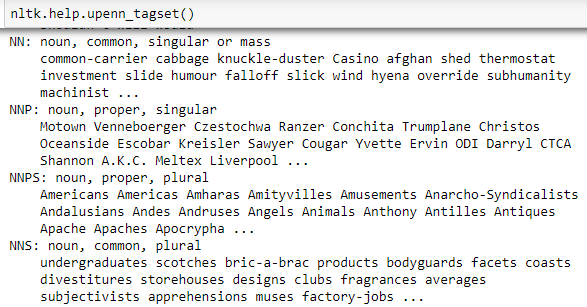
****

Looks like it worked!

1. For the topic model, we should want to include only nouns.
   1. First, find out all the POS tags that correspond to nouns.
   2. Limit the data to only terms with these tags.

First, finding out the different POS tags using the NLTK help utility.

nltk.help.upenn\_tagset()



The tags we are interested in are NN, NNP, NNS, and NNPS, all tags that begin with ‘NN’.

The output of a POS tagging is a tuple.

tagged\_tuple = nltk.pos\_tag(['great'])

tagged\_tuple[0]



So you’ll need to extract the second element of each tuple.

reviews\_noun=[]

List comprehension for the extraction.

for sent in reviews\_tagged:

reviews\_noun.append([token for token in sent if re.search("NN.\*", token[1])])

reviews\_noun[0]



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

Using the Wordnet Lemmatizer for this task.

Instantiating the lemmatizer.

lemm = WordNetLemmatizer()

Creating an empty list for the result.

reviews\_lemm=[]

List comprehension to lemmatize all the reviews.

for sent in reviews\_noun:

reviews\_lemm.append([lemm.lemmatize(word[0]) for word in sent])

reviews\_lemm[0]



1. Remove stop words and punctuation (if there are any).

from string import punctuation

from nltk.corpus import stopwords

We’ll take stop words from NLTK and punctuation from string.

stop\_nltk = stopwords.words("english")

Final stop word list will be a combination of these two.

stop\_updated = stop\_nltk + list(punctuation) + ["..."] + [".."]

Creating an empty list for the result.

reviews\_sw\_removed=[]

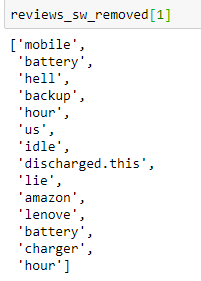
Looping over the reviews:

for sent in reviews\_lemm:

reviews\_sw\_removed.append([term for term in sent if term not in stop\_updated])

Examining one of them:

reviews\_sw\_removed[1]



1. 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?

Importing the required modules:

import gensim

import gensim.corpora as corpora

from gensim.models import CoherenceModel

from gensim.models import ldamodel

Using Gensim’s corpora utility, getting term to index mapping for each term:

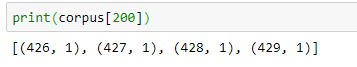
id2word = corpora.Dictionary(reviews\_sw\_removed)

texts = reviews\_sw\_removed

Applying id2word on our reviews data to get the indices, instead of words for each review.

corpus = [id2word.doc2bow(text) for text in texts]

print(corpus[200])



Building the topic model using LDA, with 12 topics.

lda\_model = gensim.models.ldamodel.LdaModel(corpus=corpus,

id2word=id2word,

num\_topics=12,

random\_state=42,

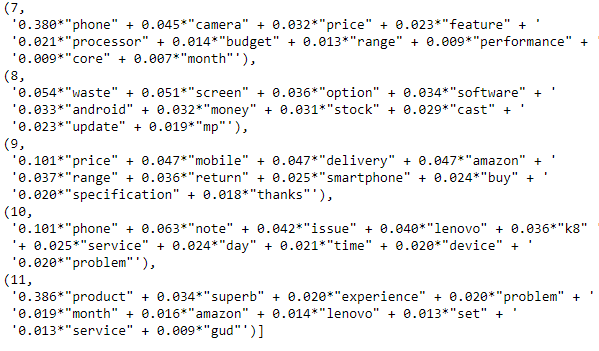
passes=10,

per\_word\_topics=True)

Printing out the top terms for the topics.

pprint(lda\_model.print\_topics())





Calculating the coherence of the model using the c\_v metric.

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)



1. 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 2 and 5 possibly talks about 'pricing'
* Topic 4, 6 and 10 closely talks about 'battery related issues'
* Topic 3 and 11 vaguely talks about 'performance'

1. Create topic model using LDA with what you think is the optimal number of topics
   1. What is the coherence of the model?

8 topics seems to be the right number of topics from the data.

We’ll create a topic model with 8 topics.

# Build LDA model:

lda\_model8 = gensim.models.ldamodel.LdaModel(corpus=corpus,

id2word=id2word,

num\_topics=8,

random\_state=42,

passes=10,

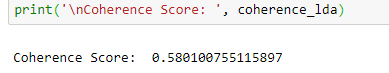
per\_word\_topics=True)

Printing the coherence of the model:

coherence\_model\_lda = CoherenceModel(model=lda\_model8, texts=reviews\_sw\_removed, dictionary=id2word, coherence='c\_v')

coherence\_lda = coherence\_model\_lda.get\_coherence()

print('\nCoherence Score: ', coherence\_lda)



The coherence is now 0.58 which is a significant increase from 0.53 previously.

1. 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.

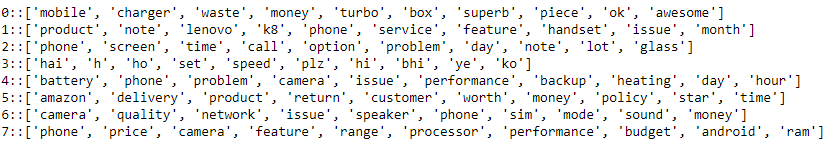
x = lda\_model8.show\_topics(formatted=False)

topics\_words = [(tp[0], [wd[0] for wd in tp[1]]) for tp in x]

for topic,words in topics\_words:

print(str(topic)+ "::"+ str(words))

print()



Giving business names and interpretation to these topics.

|  |  |
| --- | --- |
| **Topic** | **Business Name** |
| Topic1 | Product Accessories |
| Topic2 | Amazon |
| Topic3 | Pricing |
| Topic4 | Phone Performance |
| Topic5 | Battery Related Issues |
| Topic6 | Camera Quality |
| Topic7 | Sound Features |
| Topic8 | Overall General Phone Features |