Natural Language Processing

Review Project Analysis

Report and Source Code

Import the dataset, **"K8 Reviews v0.2.csv"** from the location it saved in your computer in Jupyter Notebook and other basic libraries required for the analysis and building recommendation model.

Tasks:

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

Solution: Importing Libraries:

Input code: import pandas as pd

Reading and saving the file in dataframe variable:

Input Code:

df = pd.read_csv("K8 Reviews v0.2.csv")

df

Output: The output has 14675 rows and 2 columns displaying the sentiment and reviews with 0 value of sentiment as negative and 1 value to be positive review as shown below:

	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...
5	0	Only I'm telling don't buyI'm totally disappoi...
6	1	Phone is awesome. But while charging, it heats...
7	0	The battery level has worn down
8	0	It's over hitting problems...and phone hanging...
9	0	A lot of glitches dont buy this thing better g...
10	0	Wrost
11	1	Good phone but charger not working / damage wi...
12	0	Don't purchase this item, It is so much of hea...

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

   Input Code:

   df["review"] = df["review"].str.lower()

   df["review"]

   Output: Contains a list of reviews with 14675 rows of only review text. A glimpse is shown in the image below:

```
: 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...
  5       only i'm telling don't buyi'm totally disappoi...
  6       phone is awesome. but while charging, it heats...
  7                          the battery level has worn down
  8       it's over hitting problems...and phone hanging...
  9       a lot of glitches dont buy this thing better g...
  10                                                  wrost
  11      good phone but charger not working / damage wi...
  12      don't purchase this item, it is so much of hea...
  13      i have faced the battery problem and motherboa...
  14      very good phone slim good battry backup good s...
  15                                headset is not available
  16      every time automatic on and off so kindly sugg...
  17      best product according to their prize range an...
  18      battery draining very rapidly i don't know why...
  19                                        good smartphone
  20                                                   good
  21      galery problem and there is not atmos speakern...
  22      excellent camera , excellent speed.excellent f...
  23                                     it ok good product
  24      it is not a very good product camera are very ...
  25      does not have many options like cast screen, w...
```

   Extracting to list:

   Input code: reviewData = df["review"].tolist()

3. Tokenize the reviews using NLTKs word_tokenize function.

   Input Code:

   from nltk import word_tokenize, pos_tag

   tokenizedReviews = [word_tokenize(x) for x in reviewData]

   tokenizedReviews

   Output: A glimpse of is shown in the image below:

```
[['good', 'but', 'need', 'updates', 'and', 'improvements'],
 ['worst',
  'mobile',
  'i',
  'have',
  'bought',
  'ever',
  ',',
  'battery',
  'is',
  'draining',
  'like',
  'hell',
  ',',
  'backup',
  'is',
  'only'
```

4. Perform parts-of-speech tagging on each sentence using the NLTK POS tagger.
   Input Code:
   posReviews = [pos_tag(x) for x in tokenizedReviews]
   posReviews
   Output: A glimpse of using the POS tagger is shown in the image below:

```
[[('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'),
```

5. For the topic model, we want to include only nouns.
   a. Find out all the POS tags that correspond to nouns.
   b. Limit the data to only terms with these tags.

Solution: Tags corresponding to the nouns are NN, NNS, NNP and NNPS

Input Code:

```
nounPosReviews = []
for posReview in posReviews:
    temp = []
    for x in posReview:
        if x[-1][0] == 'N':
          temp.append(x)
          nounPosReviews.append(temp)
nounPosReviews
```

Output: Limiting the data to these tags, glimpse is shown in the image below:

```
[[('updates', 'NNS'), ('improvements', 'NNS')],
 [('mobile', 'NN'),
  ('i', 'NN'),
  ('battery', 'NN'),
  ('hell', 'NN'),
  ('backup', 'NN'),
  ('hours', 'NNS'),
  ('uses', 'NNS'),
  ('idle', 'NN'),
  ('discharged.this', 'NN'),
  ('lie', 'NN'),
  ('amazon', 'NN'),
  ('lenove', 'NN'),
  ('battery', 'NN'),
  ('charger', 'NN'),
  ('hours', 'NNS'),
  ('don', 'NN')],
 [('i', 'NN'), ('%', 'NN'), ('cash', 'NN'), ('january..', 'NN')],
 [],
```

6. Lemmatize.
    a. Different forms of the terms need to be treated as one.
    b. No need to provide POS tag to lemmatizer for now.

Input Code:

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

lemmatizedNounPosReviews = [[(lemmatizer.lemmatize(x[0]),x[1]) for x in i] for i in nounPosReviews]

lemmatizedNounPosReviews

Output: A glimpse of using the lemmatizer is shown below:

```
[[('update', 'NNS'), ('improvement', 'NNS')],
 [('mobile', 'NN'),
  ('i', 'NN'),
  ('battery', 'NN'),
  ('hell', 'NN'),
  ('backup', 'NN'),
  ('hour', 'NNS'),
  ('us', 'NNS'),
  ('idle', 'NN'),
  ('discharged.this', 'NN'),
  ('lie', 'NN'),
  ('amazon', 'NN'),
  ('lenove', 'NN'),
  ('battery', 'NN'),
  ('charger', 'NN'),
  ('hour', 'NNS'),
  ('don', 'NN')],
 [('i', 'NN'), ('%', 'NN'), ('cash', 'NN'), ('january..', 'NN')],
 [],
```

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

Input Code:

from nltk.corpus import stopwords

stopwordList = stopwords.words('english')

cleanedNounPosReviews = [[(x[0],x[1]) for x in i if x[0] not in stopwordList] for i in lemmatizedNounPosReviews]

cleanedNounPosReviews

Output: A glimpse is shown in the image below:

```
[[('update', 'NNS'), ('improvement', 'NNS')],
 [('mobile', 'NN'),
  ('battery', 'NN'),
  ('hell', 'NN'),
  ('backup', 'NN'),
  ('hour', 'NNS'),
  ('us', 'NNS'),
  ('idle', 'NN'),
  ('discharged.this', 'NN'),
  ('lie', 'NN'),
  ('amazon', 'NN'),
  ('lenove', 'NN'),
  ('battery', 'NN'),
  ('charger', 'NN'),
```

8. Create a topic model using LDA on the cleaned-up data with 12 topics.
    a. Print out the top terms for each topic.
    b. What is the coherence of the model with the c_v metric?

Input Code:

import gensim

import gensim.corpora as corpora

Lda = gensim.models.ldamodel.LdaModel

Drop tags:

lemmatizedReviews = [[i[0] for i in x] for x in cleanedNounPosReviews]

lemmatizedReviews

Output:

```
[['update', 'improvement'],
 ['mobile',
  'battery',
  'hell',
  'backup',
  'hour',
  'us',
  'idle',
  'discharged.this',
  'lie',
  'amazon',
  'lenove',
  'battery',
  'charger',
```

Input Code:

id2word = corpora.Dictionary(lemmatizedReviews)

texts = lemmatizedReviews

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

lda_model = Lda(corpus=corpus, id2word=id2word, num_topics=12)

lda_model.top_topics(corpus)

Output: Printing out terms for each topic:

```
[([(0.08434804, 'issue'),
   (0.051711515, 'phone'),
   (0.03998527, 'network'),
   (0.039461244, 'battery'),
   (0.021033444, 'hour'),
   (0.020168453, 'sim'),
   (0.019357868, 'time'),
   (0.01911929, 'note'),
   (0.018062659, '%'),
   (0.016989574, 'charge'),
   (0.016362302, 'call'),
   (0.0130618755, 'option'),
   (0.012135512, 'data'),
```

Input Code:

topTerms = [max(x[0], key=lambda l:l[0]) for x in lda_model.top_topics(corpus)]

topTerms

Output: Printing out top terms:

```
[(0.08434804, 'issue'),
 (0.13116528, 'phone'),
 (0.16186793, 'battery'),
 (0.055123806, 'camera'),
 (0.2119095, 'camera'),
 (0.094949976, 'note'),
 (0.28875387, 'phone'),
 (0.07933463, 'phone'),
 (0.22034127, 'mobile'),
 (0.31723484, 'product'),
 (0.17752673, 'price'),
 (0.05279111, 'screen')]
```

Input Code:

from gensim.models.coherencemodel import CoherenceModel

coherence_model_lda = CoherenceModel(model=lda_model, texts=texts, dictionary=id2word, coherence='c_v')

coherence_lda = coherence_model_lda.get_coherence()

print('\nCoherence Score: ', coherence_lda)

Output: The coherence score is 0.5165950021693836

9. Analyse the topics through the business lens. Determine which of the topics can be combined.

   Solution: From top terms 1st, 2nd, 9th and 12th 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?

    Input Code:

    lda_model_optimal = Lda(corpus=corpus, id2word=id2word, num_topics=9)

    coherence_model_lda_optimal = CoherenceModel(model=lda_model_optimal, texts=texts, dictionary=id2word, coherence='c_v')

    coherence_lda_optimal = coherence_model_lda_optimal.get_coherence()

    print('\nCoherence Score: ', coherence_lda_optimal)

    Output: The coherence score is 0.5083485963025702

11. The business should be able to interpret the topics.
    a. Name each of the identified topics.
    b. Create a table with the topic name and the top 10 terms in each to present to the business.

Input Code:

topTermsOptimal = [max(x[0], key=lambda l:l[0]) for x in lda_model_optimal.top_topics(corpus)]

topTermsOptimal

Output: The identified topics are:

```
[(0.15009189, 'phone'),
 (0.11674714, 'battery'),
 (0.093986794, 'problem'),
 (0.12803072, 'phone'),
 (0.057888333, 'battery'),
 (0.17438711, 'camera'),
 (0.19824827, 'mobile'),
 (0.06061089, 'camera'),
 (0.27671534, 'product')]
```

Input Code:

lda_model_optimal.top_topics(corpus)

Output: A glimpse of the optimal LDA model is shown below:

```
[([(0.15009189, 'phone'),
   (0.055172645, 'note'),
   (0.02833446, 'k8'),
   (0.022223797, 'lenovo'),
   (0.02181811, 'call'),
   (0.020232657, 'feature'),
   (0.015543854, 'speaker'),
   (0.012190103, 'screen'),
   (0.011354237, 'option'),
   (0.010253569, 'camera'),
   (0.009346517, 'issue'),
   (0.008455988, 'app'),
   (0.00835687, 'music'),
   (0.008317331, 'sound'),
   (0.008008566, 'day'),
   (0.007522663, 'time'),
   (0.0071287374, 'software'),
   (0.0069468925, 'dolby'),
   (0.0065248418, 'apps'),
   (0.0062497235, 'problem')],
```

Input Code:

```
topicMap = dict()
for i in range(len(lda_model_optimal.top_topics(corpus))):
    entry = lda_model_optimal.top_topics(corpus)[i]
    temp = sorted(entry[0], key = lambda x:x[0], reverse=True)
    temp2 = [t[1] for t in temp]
    if temp2[0] not in topicMap:
        topicMap[temp2[0]] = temp2[1:11]
    else:
        topicMap[temp2[0]+str(i)] = temp2[1:11]
pd.DataFrame.from_dict(topicMap)
```

Output: Table representing the business: Topics with top 10 terms.

	phone	battery	problem	phone3	battery4	camera	mobile	camera7	product
0	note	phone	phone	issue	performance	quality	price	quality	hai
1	k8	backup	service	time	problem	phone	phone	..	ho
2	lenovo	camera	charger	network	network	battery	product	photo	cam
3	call	day	money	battery	camera	h	range	picture	piece
4	feature	issue	amazon	screen	issue	feature	feature	display	lenovo
5	speaker	life	heating	everything	handset	money	camera	product	bhi
6	screen	%	day	heat	work	waste	superb	sound	ko
7	option	device	product	budget	product	mode	performance	set	ye
8	camera	hour	battery	sim	hr	performance	buy	super	tha
9	issue	update	delivery	lot	speed	depth	box	excellent	yesterday