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

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.

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

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving K8 Reviews v0.2.csv to K8 Reviews v0.2 (2).csv
```

```
import pandas as pd
```

```
reviews = pd.read_csv('K8 Reviews v0.2.csv')
```

```
reviews.head()
```

	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...

```
reviews.shape
```

```
(14675, 2)
```

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

```
reviews = reviews.review.values
```

```
reviews = [text.lower() for text in reviews]
reviews[:3]
```

```
['good but need updates and improvements',
 "worst mobile i have bought ever, battery is draining like hell, backup is only 6 to 7 hours witl
 'when i will get my 10% cash back.... its already 15 january..']
```

Task 3: Tokenize the reviews using NLTKs word_tokenize function.

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
from nltk.tokenize import word_tokenize
reviews = [word_tokenize(text) for text in reviews]
```

```
reviews[:1]
```

```
[['good', 'but', 'need', 'updates', 'and', 'improvements']]
```

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

```
from nltk.tag import pos_tag
```

```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
True
```

```
reviews = [nltk.pos_tag(review) for review in reviews]
```

```
reviews[:2]
```

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

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

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

```
def noun_pos_tags(reviews):
    reviews_postag=[]
    for review in reviews:
        for word,pos in review:
            if pos=='NN' or pos=='NNS' or pos=='NNP' or pos=='NNPS':
                reviews_postag.append(word)
            else:
                pass
    return reviews_postag
```

```
tagged_reviews = noun_pos_tags(reviews)
tagged_reviews
```

```
['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',
 'camerawaste',
 'money',
 'phone',
 'reason',
 'k8',
 'battery',
 'level',
 'problems',
```

```
 'phone',
 'hanging',
 'problems',
 'note',
 'station',
 'ahmedabad',
 'years',
 'phone',
 'lenovo',
 'lot',
 'glitches',
 'thing',
 'options',
 'wrost',
 'phone',
 'charger',
 'damage',
 'months',
 'item',
 'battery',
 'life',
 'i'
```

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

```
#removing punctuation

tagged_reviews = [word for word in tagged_reviews if word.isalpha()]
```

```
len(tagged_reviews)
```

```
83530
```

```
# removing stopwords

from nltk.corpus import stopwords
```

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
stop_words = set(stopwords.words('english'))
tagged_reviews = [w for w in tagged_reviews if not w in stop_words]
```

```
len(tagged_reviews)
```

```
80355
```

```
tagged_reviews
```

```
['updates',
 'improvements',
 'mobile',
 'battery',
 'hell',
 'backup',
 'hours',
 'uses',
 'idle',
 'lie',
 'amazon',
 'lenove',
 'battery',
 'charger',
 'hours',
 'cash',
 'phone',
 'everthey',
 'phone',
 'problem',
 'amazon',
 'phone',
 'amazon',
 'camerawaste',
 'money',
 'phone',
 'reason',
 'battery',
 'level',
 'problems',
 'phone',
 'hanging',
 'problems',
 'note',
 'station',
 'ahmedabad',
 'years',
 'phone',
 'lenovo',
 'lot',
 'glitches',
 'thing',
 'options',
 'wrost',
 'phone',
 'charger',
 'damage',
 'months',
 'item',
 'battery',
 'life',
 'battery',
 'problem',
 'motherboard',
 'problem',
 'months',
 'mobile',
 'life',
 'phone',
```

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.

```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
```

```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

```
lemmatized_wordlist = [wordnet_lemmatizer.lemmatize(word) for word in tagged_reviews]
```

```
lemmatized_wordlist
```

```
['update',
 'improvement',
 'mobile',
 'battery',
 'hell',
 'backup',
 'hour',
 'us',
 'idle',
 'lie',
 'amazon',
 'lenove',
 'battery',
 'charger',
 'hour',
 'cash',
 'phone',
 'everthey',
 'phone',
 'problem',
 'amazon',
 'phone',
 'amazon',
 'camerawaste',
 'money',
 'phone',
 'reason',
 'battery',
 'level',
 'problem',
 'phone',
 'hanging',
 'problem',
 'note',
```

```
 'station',
 'ahmedabad',
 'year',
 'phone',
 'lenovo',
 'lot',
 'glitch',
 'thing',
 'option',
 'wrost',
 'phone',
 'charger',
 'damage',
 'month',
 'item',
 'battery',
 'life',
 'battery',
 'problem',
 'motherboard',
 'problem',
 'month',
 'mobile',
 'life',
 'phone',
```

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?

```
from gensim import corpora, models
import gensim
```

```
dictionary = corpora.Dictionary([lemmatized_wordlist])
```

```
print(dictionary)
```

```
Dictionary(5965 unique tokens: ['aa', 'aab', 'aachha', 'aaguthu', 'aaj']...)
```

```
print(dictionary.token2id)
```

```
{'aa': 0, 'aab': 1, 'aachha': 2, 'aaguthu': 3, 'aaj': 4, 'aajata': 5, 'aap': 6, 'aapki': 7, 'aapk
```

```
corpus = [dictionary.doc2bow(text) for text in [lemmatized_wordlist]]
```

```
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics=12, id2word=dictionary)
```

```
print(ldamodel)
```

```
LdaModel(num_terms=5965, num_topics=12, decay=0.5, chunksize=2000)
```

```
for idx, topic in ldamodel.print_topics(-1):
    print("Topic: {} \nWords: {}".format(idx, topic ))
    print("\n")
```

```
Topic: 0
Words: 0.082*"phone" + 0.041*"battery" + 0.036*"camera" + 0.030*"product" + 0.026*"issue" + 0.021


Topic: 1
Words: 0.063*"phone" + 0.041*"camera" + 0.037*"battery" + 0.027*"product" + 0.023*"mobile" + 0.02


Topic: 2
Words: 0.055*"phone" + 0.030*"camera" + 0.021*"battery" + 0.020*"product" + 0.019*"mobile" + 0.01


Topic: 3
Words: 0.080*"phone" + 0.041*"battery" + 0.032*"camera" + 0.031*"product" + 0.016*"mobile" + 0.01


Topic: 4
Words: 0.085*"phone" + 0.032*"camera" + 0.030*"battery" + 0.023*"product" + 0.022*"quality" + 0.02


Topic: 5
Words: 0.037*"phone" + 0.022*"battery" + 0.013*"camera" + 0.011*"product" + 0.009*"mobile" + 0.008


Topic: 6
Words: 0.096*"phone" + 0.038*"battery" + 0.034*"camera" + 0.027*"product" + 0.018*"problem" + 0.01


Topic: 7
Words: 0.057*"phone" + 0.034*"camera" + 0.034*"battery" + 0.027*"product" + 0.018*"mobile" + 0.016


Topic: 8
Words: 0.064*"phone" + 0.032*"camera" + 0.027*"battery" + 0.021*"mobile" + 0.017*"issue" + 0.017*


Topic: 9
Words: 0.059*"phone" + 0.030*"battery" + 0.028*"camera" + 0.020*"product" + 0.018*"mobile" + 0.014


Topic: 10
Words: 0.084*"phone" + 0.052*"camera" + 0.032*"battery" + 0.023*"product" + 0.018*"problem" + 0.01


Topic: 11
Words: 0.075*"phone" + 0.038*"battery" + 0.029*"camera" + 0.022*"product" + 0.018*"mobile" + 0.018
```

```
print('\nPerplexity: ', ldamodel.log_perplexity(corpus))
```

```
print('\nPerplexity: ', ldamodel.log_perplexity(corpus))
```

```
Perplexity:  -6.622580098586599
```

```
#coherence

from gensim.models.coherencemodel import CoherenceModel
```

```
import numpy as np
np.seterr(divide='ignore', invalid='ignore')
```

```
{'divide': 'warn', 'invalid': 'warn', 'over': 'warn', 'under': 'ignore'}
```

```
cohmodel = CoherenceModel(model = ldamodel, texts=[lemmatized_wordlist], dictionary=dictionary, coheren
```

```
coherence = cohmodel.get_coherence()
```

```
print(coherence)
```

```
0.2387372440263221
```

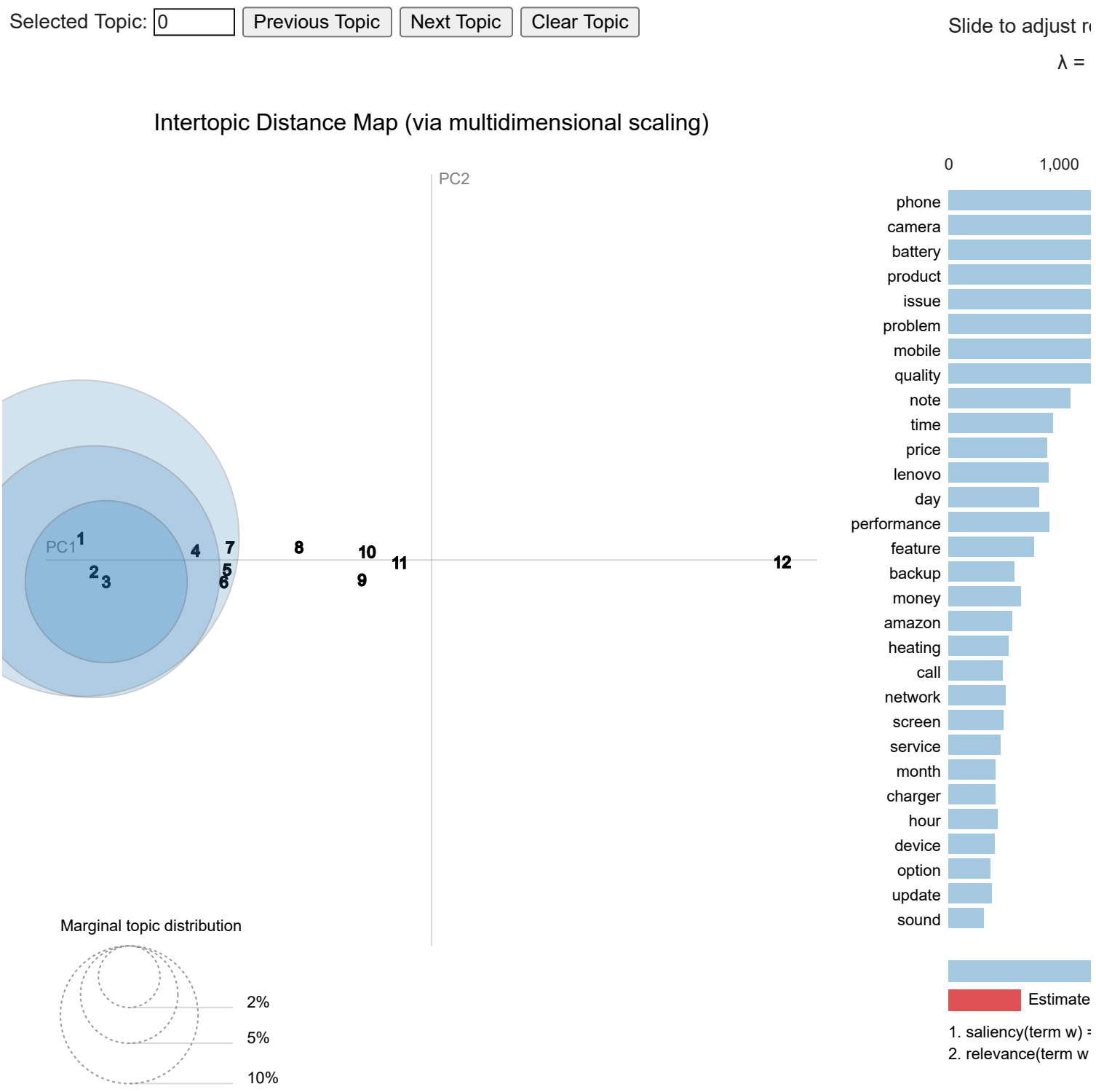
```
# !pip install pyLDAvis
```

```
# Plotting tools
import pyLDAvis
import pyLDAvis.gensim_models  # don't skip this
import matplotlib.pyplot as plt
%matplotlib inline
```

```
/usr/local/lib/python3.7/dist-packages/past/types/oldstr.py:5: DeprecationWarning: Using or impor
  from collections import Iterable
/usr/local/lib/python3.7/dist-packages/sklearn/decomposition/_lda.py:29: DeprecationWarning: `np.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0
  EPS = np.finfo(np.float).eps
```

```
# !pip install --upgrade pandas==1.2
```

```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim_models.prepare(ldamodel, corpus, dictionary)
vis
```



Task 9: Analyze the topics through the business lens.

1. Determine which of the topics can be combined.

0 - Possible Topic - BAttery issue (1)

1 - Possible Topic - Camera quality issue (2)

2 - Possible Topic - Product quality (3)

3 - Possible Topic - Servicing time (4)

4 - Possible Topic - Positive Mobile Review (5)

5 - Possible Topic - Picture quality (6)

6 - Possible Topic - Positive Review (5)

7 - Possible Topic - Review on Processor (7)

8 - Possible Topic - Positive Review (5)

9 - Possible Topic - Negative Review (8)

10 - Possible Topic - Review on Return policy (9)

11 - Possible Topic - Review on software update (10)

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

1. What is the coherence of the model?

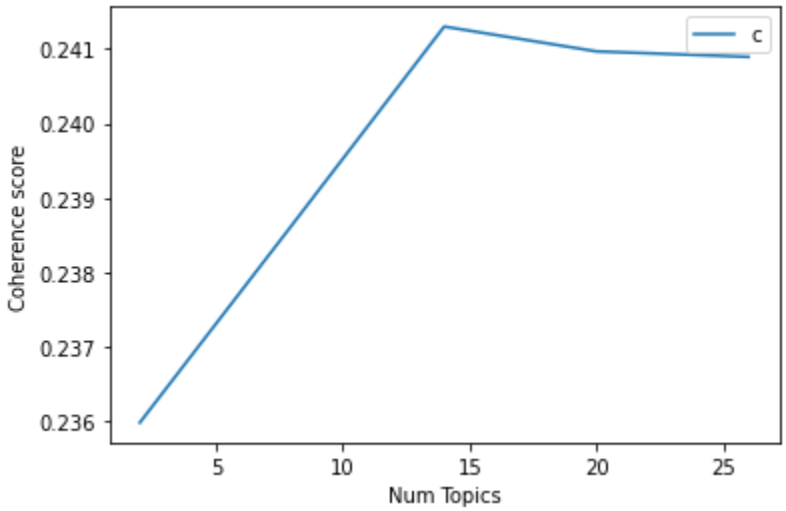
```
def compute_coherence_values(dictionary, corpus, texts, limit, start=2, step=3):
  coherence_values = []
  model_list = []
  for num_topics in range(start, limit, step):
      model = gensim.models.ldamodel.LdaModel(corpus, num_topics=num_topics, id2word=dictionary)
      model_list.append(model)
      coherencemodel = CoherenceModel(model=model, texts=[lemmatized_wordlist], dictionary=dictionary,
      coherence_values.append(coherencemodel.get_coherence())

  return model_list, coherence_values
```

```
model_list, coherence_values = compute_coherence_values(dictionary=dictionary, corpus=corpus, texts=[le
```

```
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning: Calling
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cnt in doc)
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning: Calling
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cnt in doc)
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning: Calling
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cnt in doc)
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning: Calling
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cnt in doc)
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning: Calling
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cnt in doc)
```

```
# Show graph
limit=30; start=2; step=6;
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```




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

```
Num Topics = 2  has Coherence Value of 0.236
Num Topics = 8  has Coherence Value of 0.2386
Num Topics = 14  has Coherence Value of 0.2413
Num Topics = 20  has Coherence Value of 0.241
Num Topics = 26  has Coherence Value of 0.2409
```

Coherence score decrease after 14 topic. Optimal number of topics will be 14

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.

```
#number of topic 14

ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics=14, id2word=dictionary)
for idx, topic in ldamodel.print_topics(-1):
    print("Topic: {} \nWords: {}".format(idx, topic ))
    print("\n")
```

```
Topic: 0
Words: 0.096*"phone" + 0.041*"camera" + 0.039*"battery" + 0.030*"product" + 0.018*"issue" + 0.0


Topic: 1
Words: 0.051*"phone" + 0.027*"battery" + 0.027*"camera" + 0.018*"issue" + 0.014*"problem" + 0.0


Topic: 2
Words: 0.058*"phone" + 0.039*"camera" + 0.030*"battery" + 0.030*"product" + 0.022*"problem" + 0


Topic: 3
```

```
Words: 0.061*"phone" + 0.044*"battery" + 0.033*"camera" + 0.020*"quality" + 0.020*"mobile" + 0.


Topic: 4
Words: 0.081*"phone" + 0.030*"camera" + 0.029*"battery" + 0.017*"mobile" + 0.016*"product" + 0.


Topic: 5
Words: 0.065*"phone" + 0.039*"battery" + 0.034*"camera" + 0.024*"product" + 0.021*"problem" + 0


Topic: 6
Words: 0.094*"phone" + 0.039*"camera" + 0.036*"battery" + 0.020*"product" + 0.020*"problem" + 0


Topic: 7
Words: 0.049*"phone" + 0.038*"camera" + 0.031*"battery" + 0.026*"product" + 0.021*"mobile" + 0.


Topic: 8
Words: 0.078*"phone" + 0.036*"battery" + 0.031*"camera" + 0.028*"product" + 0.021*"mobile" + 0.


Topic: 9
Words: 0.069*"phone" + 0.036*"camera" + 0.030*"battery" + 0.025*"product" + 0.022*"problem" + 0


Topic: 10
Words: 0.040*"phone" + 0.028*"camera" + 0.024*"product" + 0.023*"battery" + 0.017*"issue" + 0.0


Topic: 11
Words: 0.073*"phone" + 0.029*"battery" + 0.029*"camera" + 0.025*"product" + 0.019*"issue" + 0.0


Topic: 12
Words: 0.068*"phone" + 0.035*"battery" + 0.032*"camera" + 0.022*"mobile" + 0.022*"problem" + 0.


Topic: 13
Words: 0.066*"phone" + 0.039*"camera" + 0.037*"battery" + 0.022*"product" + 0.017*"quality" + 0


/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: DeprecationWarning: Call
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cnt in doc)
```