```
import nltk
import pandas as pd
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('tagsets')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package tagsets to /root/nltk_data...
[nltk_data]   Unzipping help/tagsets.zip.
True
```

Reading the data

```
data = pd.read_csv("/content/K8 Reviews v0.2.csv")
data.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...

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

```
normalized_reviews = [sent.lower() for sent in data.review.values]
normalized_reviews[:5]
```

```
['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.don'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']
```

Tokenize the reviews using NLTKs word_tokenize function.

```
tokenized_reviews = [nltk.word_tokenize(review) for review in normalized_reviews]
tokenized_reviews[:5]
```

```
  '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']]
```

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

```
pos_tagged_reviews = [nltk.pos_tag(review) for review in tokenized_reviews]
pos_tagged_reviews[5]
```

```
[('only', 'RB'),
 ('i', 'JJ'),
 ("'m", 'VBP'),
 ('telling', 'VBG'),
 ('do', 'VBP'),
 ("n't", 'RB'),
 ('buyi', 'VB'),
 ("'m", 'VBP'),
 ('totally', 'RB'),
 ('disappointedpoor', 'JJ'),
 ('batterypoor', 'JJ'),
 ('camerawaste', 'NN'),
 ('of', 'IN'),
 ('money', 'NN')]
```

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

```
nltk.help.upenn_tagset()
```

```
$: dollar
    $ -$ --$ A$ C$ HK$ M$ NZ$ S$ U.S.$ US$
'': closing quotation mark
    ' ''
(: opening parenthesis
    ( [ {
): closing parenthesis
    ) ] }
,: comma
    ,
--: dash
    --
.: sentence terminator
    . ! ?
:: colon or ellipsis
    : ; ...
CC: conjunction, coordinating
    & 'n and both but either et for less minus neither nor or plus so
    therefore times v. versus vs. whether yet
CD: numeral, cardinal
    mid-1890 nine-thirty forty-two one-tenth ten million 0.5 one forty-
    seven 1987 twenty '79 zero two 78-degrees eighty-four IX '60s .025
    fifteen 271,124 dozen quintillion DM2,000 ...
DT: determiner
    all an another any both del each either every half la many much nary
    neither no some such that the them these this those
EX: existential there
    there
FW: foreign word
    gemeinschaft hund ich jeux habeas Haementeria Herr K'ang-si vous
    lutihaw alai je jour objets salutaris fille quibusdam pas trop Monte
    terram fiche oui corporis ...
IN: preposition or conjunction, subordinating
    astride among uppon whether out inside pro despite on by throughout
    below within for towards near behind atop around if like until below
    next into if beside ...
JJ: adjective or numeral, ordinal
    third ill-mannered pre-war regrettable oiled calamitous first separable
    ectoplasmic battery-powered participatory fourth still-to-be-named
    multilingual multi-disciplinary ...
```

```
JJR: adjective, comparative
    bleaker braver breezier briefer brighter brisker broader bumper busier
    calmer cheaper choosier cleaner clearer closer colder commoner costlier
    cozier creamier crunchier cuter ...
JJS: adjective, superlative
    calmest cheapest choicest classiest cleanest clearest closest commonest
    corniest costliest crassest creepiest crudest cutest darkest deadliest
    dearest deepest densest dinkiest ...
LS: list item marker
    A A. B B. C C. D E F First G H I J K One SP-44001 SP-44002 SP-44005
    SP-44007 Second Third Three Two * a b c d first five four one six three
    two
MD: modal auxiliary
    can cannot could couldn't dare may might must need ought shall should
    shouldn't will would
NN: noun, common, singular or mass
    common-carrier cabbage knuckle-duster Casino afghan shed thermostat
    investment slide humour falloff slick wind hyena override subhumanity
```

Loooking at pos tagset we have found that there are 4 tags related to nouns - NN, NNP, NNPS, NNS

```
import re
noun_reviews = []
for word in pos_tagged_reviews:
    noun_reviews.append([token for token in word if re.search("NN.*", token[1])])

noun_reviews[:5]
```

```
[[('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'), ('..', 'NN')],
 [],
 [('phone', 'NN'),
  ('everthey', 'NN'),
  ('phone', 'NN'),
  ('problem', 'NN'),
  ('amazon', 'NN'),
  ('phone', 'NN'),
  ('amazon', 'NN')]]
```

Lemmatize

```
# Initialize lemmatizer
lemmatizer = nltk.WordNetLemmatizer()
```

```
lemmatized_reviews = []
for sent in noun_reviews:
    lemmatized_reviews.append([lemmatizer.lemmatize(word[0]) for word in sent])
```

```
print(noun_reviews[68])
print(lemmatized_reviews[68])
```

```
[('camera', 'NN'), ('mo', 'NN'), ('issue', 'NN'), ('system', 'NN'), ('system', 'NN'), ('updates', 'NNS'
['camera', 'mo', 'issue', 'system', 'system', 'update']
```

Remove stopwords and punctuations if there are any

```
from nltk.corpus import stopwords
nltk.download('stopwords')
from string import punctuation
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
stop_words = stopwords.words('english') + list(punctuation) + [".."] + ["..."]
```

```
cleaned_reviews = []
for words in lemmatized_reviews:
    cleaned_reviews.append([word for word in words if word not in stop_words])

print(lemmatized_reviews[24])
print(cleaned_reviews[24])
```

```
['product', 'camera', 'o', 'battery', 'phone', 'product', '..']
['product', 'camera', 'battery', 'phone', 'product']
```

Topic modelling using LDA

```
import gensim
import gensim.corpora as corpora
from gensim.models import ldamodel
from gensim.models import CoherenceModel
```

```
id2word = corpora.Dictionary(cleaned_reviews)
corpus = [id2word.doc2bow(word) for word in cleaned_reviews]
```

```
# Building the LDA model
lda_model = ldamodel.LdaModel(corpus=corpus, num_topics=12, id2word=id2word, random_state=42, passes=10, per_
```

```
# coherence of the model with c_v metric
coherence_model = CoherenceModel(model=lda_model, texts=cleaned_reviews, dictionary=id2word, coherence='c_v')
print("Model score:", coherence_model.get_coherence())
```

```
Model score: 0.48926964040228144
```

```
lda_model.show_topics()
```

```
[(5,
  '0.093*"heat" + 0.070*"....." + 0.052*"processor" + 0.038*"everything" + 0.038*"budget" +
0.031*"...." + 0.030*"core" + 0.025*"display" + 0.017*"cell" + 0.016*"hr"'),
 (9,
  '0.158*"camera" + 0.136*"battery" + 0.064*"quality" + 0.061*"phone" + 0.045*"performance" +
0.029*"backup" + 0.019*"issue" + 0.017*"life" + 0.017*"day" + 0.015*"mode"'),
 (1,
  '0.151*"money" + 0.128*"...." + 0.071*"waste" + 0.056*"value" + 0.046*"glass" + 0.038*"speaker" +
0.024*"gorilla" + 0.022*"set" + 0.022*"ok" + 0.020*"piece"'),
 (2,
  '0.216*"note" + 0.113*"k8" + 0.090*"lenovo" + 0.030*"sound" + 0.023*"dolby" + 0.020*"killer" +
0.018*"gallery" + 0.018*"system" + 0.018*"atmos" + 0.018*"excellent"'),
 (0,
  '0.138*"mobile" + 0.040*"call" + 0.036*"screen" + 0.031*"feature" + 0.030*"option" + 0.020*"music" +
0.017*"software" + 0.016*"app" + 0.015*"video" + 0.015*"card"'),
 (6,
  '0.126*"range" + 0.075*"price" + 0.046*"work" + 0.038*"mobile" + 0.038*"specification" +
0.035*"super" + 0.034*"......" + 0.030*"bit" + 0.026*"cam" + 0.025*"k"'),
 (3,
  '0.078*"phone" + 0.040*"day" + 0.038*"amazon" + 0.035*"service" + 0.034*"issue" + 0.027*"time" +
0.027*"lenovo" + 0.026*"battery" + 0.024*"month" + 0.023*"device"'),
 (4,
  '0.280*"product" + 0.176*"problem" + 0.080*"network" + 0.075*"issue" + 0.066*"heating" + 0.021*"jio"
+ 0.021*"sim" + 0.019*"volta" + 0.010*"connection" + 0.009*"signal"'),
 (7,
  '0.118*"charger" + 0.059*"hai" + 0.056*"handset" + 0.037*"box" + 0.029*"turbo" + 0.027*"charge" +
0.021*"plz" + 0.016*"hi" + 0.016*"cable" + 0.013*"bhi"'),
 (8,
  '0.242*"price" + 0.065*"superb" + 0.046*"buy" + 0.045*"headphone" + 0.039*"thanks" + 0.036*"worth" +
0.034*"feature" + 0.029*"smartphone" + 0.026*"expectation" + 0.017*"offer"')]
```

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

9 topics seem to be optimal number of topics with more coherence score

```
# Building the LDA model
lda_model2 = ldamodel.LdaModel(corpus=corpus, num_topics=9, id2word=id2word, random_state=42, passes=10, per_
```

```
# coherence of the model with c_v metric
coherence_model2 = CoherenceModel(model=lda_model2, texts=cleaned_reviews, dictionary=id2word, coherence='c_v
print("Model score:", coherence_model2.get_coherence())
```

```
Model score: 0.5659949842769635
```

```
lda_model2.show_topics()
```

```
[(0,
  '0.119*"mobile" + 0.037*"screen" + 0.034*"call" + 0.024*"option" + 0.022*"feature" + 0.020*"video" +
0.016*"app" + 0.015*"light" + 0.014*"cast" + 0.013*"music"'),
 (1,
  '0.120*"money" + 0.056*"waste" + 0.045*"value" + 0.044*"performance" + 0.041*"speaker" +
0.037*"glass" + 0.027*"mobile" + 0.023*"piece" + 0.019*"gorilla" + 0.017*"budget"'),
 (2,
  '0.119*"note" + 0.068*"k8" + 0.055*"feature" + 0.050*"lenovo" + 0.029*"phone" + 0.021*"sound" +
0.021*"h" + 0.019*"dolby" + 0.014*"quality" + 0.012*"gb"'),
 (3,
  '0.212*"phone" + 0.032*"issue" + 0.031*"battery" + 0.031*"day" + 0.028*"time" + 0.021*"month" +
0.020*"problem" + 0.019*"lenovo" + 0.017*"service" + 0.013*"update"'),
 (4,
  '0.215*"product" + 0.101*"problem" + 0.053*"network" + 0.042*"heating" + 0.041*"issue" +
0.024*"amazon" + 0.023*"sim" + 0.017*"return" + 0.014*"jio" + 0.013*"volta"'),
 (5,
  '0.152*"camera" + 0.112*"battery" + 0.071*"phone" + 0.059*"quality" + 0.031*"performance" +
0.028*"backup" + 0.015*"processor" + 0.014*"mode" + 0.013*"life" + 0.012*"issue"'),
 (6,
  '0.207*"price" + 0.084*"phone" + 0.081*"range" + 0.050*"device" + 0.032*"mobile" + 0.023*"super" +
0.019*"set" + 0.012*"gud" + 0.008*"concern" + 0.008*"complain"'),
 (7,
  '0.082*"charger" + 0.041*"hai" + 0.039*"handset" + 0.020*"turbo" + 0.017*"ho" + 0.017*"box" +
0.014*"plz" + 0.014*"item" + 0.011*"charge" + 0.011*"hi"'),
 (8,
  '0.088*"...." + 0.064*"....." + 0.060*"delivery" + 0.046*"superb" + 0.032*"headphone" +
0.028*"thanks" + 0.027*"smartphone" + 0.027*"earphone" + 0.024*"......" + 0.021*"ok"')]
```

Interpret the topics

```
topics = lda_model2.show_topics(formatted=False)
topics_words = [(topic[0], [word[0] for word in topic[1]]) for topic in topics]

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

```
0 ['mobile', 'screen', 'call', 'option', 'feature', 'video', 'app', 'light', 'cast', 'music']
1 ['money', 'waste', 'value', 'performance', 'speaker', 'glass', 'mobile', 'piece', 'gorilla', 'budget'
2 ['note', 'k8', 'feature', 'lenovo', 'phone', 'sound', 'h', 'dolby', 'quality', 'gb']
3 ['phone', 'issue', 'battery', 'day', 'time', 'month', 'problem', 'lenovo', 'service', 'update']
4 ['product', 'problem', 'network', 'heating', 'issue', 'amazon', 'sim', 'return', 'jio', 'volta']
5 ['camera', 'battery', 'phone', 'quality', 'performance', 'backup', 'processor', 'mode', 'life', 'issu
6 ['price', 'phone', 'range', 'device', 'mobile', 'super', 'set', 'gud', 'concern', 'complain']
7 ['charger', 'hai', 'handset', 'turbo', 'ho', 'box', 'plz', 'item', 'charge', 'hi']
8 ['....', '.....', 'delivery', 'superb', 'headphone', 'thanks', 'smartphone', 'earphone', '......', 'o
```

✓ 0s completed at 11:45 AM