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

import gensim
import gensim.corpora as corpora
from gensim.utils import  simple_preprocess
from gensim.models import CoherenceModel

import spacy

import pyLDAvis
import pyLDAvis.gensim_models
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

In [ ]:
```
# import dataset
df = pd.read_csv('cleaned_data.csv')
df.head()
```

Out[ ]:

	Consumer complaint narrative	Issue	Product
0	My minimum payment for my Fortiva Retail Credi...	Problem when making payments	Credit card or prepaid card
1	I HAD MY IDENTITY STOLEN AND SOMEONE CREATED F...	Problem with a credit reporting company's inve...	Credit reporting, credit repair services, or o...
2	I'm sending this complaint to inform AGAIN the...	Incorrect information on your report	Credit reporting, credit repair services, or o...
3	IDENTITY THEFT XX/XX/XXXX XXXX XXXX XXXX, XXXX...	Incorrect information on your report	Credit reporting, credit repair services, or o...
4	I have filed numerous complaints with the 3 cr...	Problem with a credit reporting company's inve...	Credit reporting, credit repair services, or o...

In [ ]:
```
# print product type
print(df['Product'].nunique())
print(df['Product'].unique())

# print issue
print(df['Issue'].nunique())
print(df['Issue'].unique())
```

```
18
['Credit card or prepaid card'
 'Credit reporting, credit repair services, or other personal consumer reports'
 'Debt collection' 'Vehicle loan or lease' 'Checking or savings account'
 'Mortgage' 'Money transfer, virtual currency, or money service'
 'Payday loan, title loan, or personal loan' 'Student loan'
 'Credit reporting' 'Credit card' 'Consumer Loan'
 'Bank account or service' 'Money transfers' 'Payday loan' 'Prepaid card'
 'Other financial service' 'Virtual currency']
157
['Problem when making payments'
 "Problem with a credit reporting company's investigation into an existing problem"
 'Incorrect information on your report' 'Written notification about debt'
 'Struggling to pay your loan' 'Managing an account'
 'Problems at the end of the loan or lease'
 'Problem with a purchase shown on your statement'
 'Improper use of your report' 'Attempts to collect debt not owed'
 'Trouble during payment process' 'Opening an account'
 'Problem with a lender or other company charging your account'
 'Money was not available when promised' 'Struggling to pay mortgage'
 'Other transaction problem' 'Communication tactics' 'Closing an account'
 'Managing the loan or lease'
 'Applying for a mortgage or refinancing an existing mortgage'
 'Threatened to contact someone or share information improperly'
 'False statements or representation' 'Trouble using your card'
 'Struggling to pay your bill'
 'Problem with the payoff process at the end of the loan'
 'Took or threatened to take negative or legal action'
 'Lost or stolen check' 'Getting a credit card'
 'Other features, terms, or problems' 'Closing your account'
 'Fees or interest' 'Fraud or scam' 'Unexpected or other fees'
 'Dealing with your lender or servicer'
 'Unable to get your credit report or credit score'
 'Credit monitoring or identity theft protection services'
 'Problem with fraud alerts or security freezes'
 'Confusing or misleading advertising or marketing'
 'Problem caused by your funds being low'
 'Advertising and marketing, including promotional offers'
 'Problem with a purchase or transfer' 'Getting a loan or lease'
 'Getting a loan' 'Problem getting a card or closing an account'
 'Closing on a mortgage' "Charged fees or interest you didn't expect"
 "Loan payment wasn't credited to your account" 'Getting a line of credit'
 'Unauthorized transactions or other transaction problem'
 'Problem with additional add-on products or services'
 'Struggling to repay your loan'
 'Identity theft protection or other monitoring services'
 'Confusing or missing disclosures'
 'Managing, opening, or closing your mobile wallet account'
 'Getting the loan' "Can't stop withdrawals from your bank account"
 "Problem with a company's investigation into an existing issue"
 "Can't contact lender or servicer" 'Problem with customer service'
 'Trouble using the card' 'Other service problem'
```

'Wrong amount charged or received'
"Was approved for a loan, but didn't receive the money"
'Lost or stolen money order' 'Excessive fees'
'Money was taken from your bank account on the wrong day or for the wrong amount'
"Received a loan you didn't apply for" 'Settlement process and costs'
'Advertising' 'Loan servicing, payments, escrow account'
"Cont'd attempts collect debt not owed"
'Incorrect information on credit report'
'Loan modification,collection,foreclosure'
'Identity theft / Fraud / Embezzlement' 'Problem adding money'
'Balance transfer' 'Disclosure verification of debt'
'Credit card protection / Debt protection' 'Taking out the loan or lease'
'Advertising and marketing' 'Problems when you are unable to pay'
'Dealing with my lender or servicer' 'Rewards' 'Billing disputes'
'Taking/threatening an illegal action'
'Improper contact or sharing of info'
"Credit reporting company's investigation"
'Vehicle was repossessed or sold the vehicle'
'Closing/Cancelling account' 'Account opening, closing, or management'
'Application, originator, mortgage broker'
'Unable to get credit report/credit score' 'Late fee'
'Improper use of my credit report' 'Using a debit or ATM card'
'Credit decision / Underwriting' 'Shopping for a loan or lease'
'Making/receiving payments, sending money' 'Delinquent account'
"Can't repay my loan" 'Problems caused by my funds being low'
'Billing statement' "Can't contact lender"
'Unsolicited issuance of credit card' 'Deposits and withdrawals'
'APR or interest rate' 'Credit limit changed' 'Other fee' 'Arbitration'
'Bankruptcy' 'Other' "Can't stop charges to bank account"
"Charged fees or interest I didn't expect" 'Other service issues'
'Credit monitoring or identity protection' 'Payment to acct not credited'
'Privacy' 'Fees' 'Incorrect/missing disclosures or info'
'Transaction issue' 'Vehicle was damaged or destroyed the vehicle'
"Received a loan I didn't apply for" 'Other transaction issues'
'Customer service / Customer relations'
'Managing, opening, or closing account' 'Account terms and changes'
'Application processing delay' 'Problem with cash advance'
'Credit determination' 'Charged bank acct wrong day or amt'
'Advertising, marketing or disclosures' 'Credit line increase/decrease'
'Unauthorized transactions/trans. issues' 'Payoff process'
'Cash advance fee' 'Forbearance / Workout plans'
'Customer service/Customer relations' 'Sale of account'
'Convenience checks' 'Applied for loan/did not receive money'
'Adding money' 'Lender repossessed or sold the vehicle'
"Was approved for a loan, but didn't receive money" 'Cash advance'
'Problem with overdraft' 'Managing the line of credit' 'Disclosures'
'Incorrect exchange rate' 'Overdraft, savings or rewards features'
'Balance transfer fee' 'Unexpected/Other fees'
'Overdraft, savings, or rewards features' 'Overlimit fee'
'Lender damaged or destroyed vehicle' 'Problem with an overdraft'
'Property was sold' 'Property was damaged or destroyed property']

We expect to have 18 topics

In [ ]:
```
df.info(memory_usage="deep")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400000 entries, 0 to 399999
Data columns (total 3 columns):
 #   Column                        Non-Null Count   Dtype 
---  ------                        --------------   ----- 
 0   Consumer complaint narrative  400000 non-null  object
 1   Issue                         400000 non-null  object
 2   Product                       400000 non-null  object
dtypes: object(3)
memory usage: 515.9 MB
```

In [ ]:
```
data = df['Consumer complaint narrative'].values.tolist()

# Remove senssored words (XX, XXXX, ...)
data = [re.sub('X{2,}', '', word) for word in data]

# Remove newline and spaces characters
data = [re.sub('[\s\n]+', ' ', word) for word in data]

# Remove distracting slash character
data = [re.sub('\/', '', word) for word in data]
```

In [ ]:
```
import random
pprint(data[10000])
# pprint(data[random.randint(0,400000)])
```

```
('This is my 9th attempt to dispute the information in my credit file which '
 'resulted from an alleged identity theft. Again, Im sending over an identity '
 'theft report # . Pursuant of FCRA section 605B ( a ) ( 1 ) ( 2 ) Credit '
 'Bureaus are required to remove any information in my file that resulted from '
 'an alleged identity theft, not later than 4 business days after the date of '
 'receipt with ( 1 ) appropriate proof of the identity of the consumer ; ( 2 ) '
 'a copy of an identity theft report. Please block following items listed. , '
 'CA ; , CA , CA , CA Balance Owed : {$7300.00} Balance Owed : {$7.00}, US SM '
 'BUS ADMIN Inquiry Date  Inquiry  Inquiry Date  Inquiry Date  Inquiry Date  '
 'Inquiry Date  Inquiry Date  Inquiry Date  Inquiry Date  Inquiry Date ')
```

In [ ]:
```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc=True removes punctuations

data_words = list(sent_to_words(data))
```

In [ ]:
```
print(data_words[90])
```

```
['recently', 'reviewed', 'copy', 'of', 'my', 'credit', 'report', 'was', 'shocked', 'to', 'learn', 'that', 'have', 'been', 'victim', 'of', 'identity',
'theft', 'found', 'names', 'on', 'my', 'credit', 'report', 'that', 'are', 'not', 'me', 'this', 'has', 'been', 'hurting', 'me', 'as', 'am', 'in', 'th
e', 'process', 'of', 'purchasing', 'new', 'home', 'for', 'my', 'family', 'please', 'remove', 'these', 'fraudulent', 'names', 'from', 'my', 'credit',
'report', 'as', 'soon', 'as', 'possible']
```

In [ ]:
```
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to C:\Users\chuk
[nltk_data]     bert\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[ ]:
```
True
```

In [ ]:
```
stop_words = stopwords.words('english')
print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselve
s', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'ha
s', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'f
or', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few',
'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'do
n', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doe
sn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "need
n't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

In [ ]:
```
stop_words.extend(['th'])
```

In [ ]:
```
```

In [ ]:
```
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=4, threshold=100) # higher threshold fewer phrases.
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)
```

In [ ]:
```
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts]

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]
```

```
# def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
#     """https://spacy.io/api/annotation"""
#     texts_out = []
#     for sent in texts:
#         doc = nlp(" ".join(sent))
#         texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
#     return texts_out

# for doc in nlp.pipe(docs, batch_size=32, n_process=3, disable=["parser", "ner"]):
#     print([tok.lemma_ for tok in doc])

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    for doc in nlp.pipe(texts, batch_size=1000, n_process=-1, disable=["parser", "ner"]):
        lemmas = [token.lemma_ for token in doc if token.pos_ in allowed_postags]
        yield lemmas

def list2string(texts):
    for doc in texts:
        result = ' '.join(doc)
        yield result
```

In [ ]:
```
# Remove Stop Words
data_words = remove_stopwords(data_words)
```

In [ ]:
```
# Form Bigrams
data_words = make_bigrams(data_words)
```

In [ ]:
```
# Form Trigrams
data_words = make_trigrams(data_words)
```

In [ ]:
```
# Initialize spacy 'en' model, keeping only tagger component (for efficiency)
# python3 -m spacy download en
nlp = spacy.load('en_core_web_sm')
```

In [ ]:
```
# Do lemmatization keeping only noun, adj, vb, adv
# data_words = lemmatization(data_words, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])

data_words = list(lemmatization(list2string(data_words), allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']))
```

In [ ]:
```
print(data_words[1])
```

```
['identity', 'steal', 'create', 'fraudelunet', 'account', 'name', 'm', 'try', 'buy', 'house', 'notoice', 'bad', 'thing', 'credit', 'get', 'car', 'loa
n', 'buy', 'house', 'depend', 'collection', 'never', 'hear', 'payment', 'due', 'time', 'person', 'ruin', 'life', 'police', 'report', 'idenity', 'thef
t']
```

In [ ]:
```
# Create Dictionary
id2word = corpora.Dictionary(data_words)
```

In [ ]:
```
# Term Document Frequency
corpus = [id2word.doc2bow(doc) for doc in data_words]
```

In [ ]:
```
# Human readable format of corpus (term-frequency)
[[(id2word[id], freq) for id, freq in cp] for cp in corpus[:1]]
```

Out[ ]:
```
[[('account', 1),
  ('bill', 1),
  ('card', 2),
  ('complaint', 1),
  ('corporate', 1),
  ('correct', 1),
  ('credit', 2),
  ('due', 1),
  ('dupe', 1),
  ('file', 1),
  ('fortiva', 1),
  ('fortiva_retail', 2),
  ('furniture', 1),
  ('go', 2),
  ('however', 1),
  ('minimum', 8),
  ('month', 4),
  ('monthly', 5),
  ('notcould', 1),
  ('office', 1),
  ('open', 1),
  ('pay', 3),
  ('payment', 8),
  ('prior', 1),
  ('purchase', 2),
  ('reduce', 3),
  ('representative', 1),
  ('sale', 1),
  ('say', 2),
  ('statement', 2),
  ('tell', 1),
  ('time', 2)]]
```

In [ ]:
```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=18,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

In [ ]:
```
# Print the Keyword in the 18 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

```
[(0,
  '0.081*"call" + 0.054*"tell" + 0.051*"get" + 0.046*"say" + 0.033*"go" + '
  '0.030*"ask" + 0.027*"try" + 0.027*"time" + 0.025*"never" + 0.025*"back"'),
 (1,
  '0.442*"account" + 0.065*"open" + 0.062*"balance" + 0.057*"date" + '
  '0.032*"close" + 0.024*"last" + 0.017*"status" + 0.015*"signature" + '
  '0.011*"activity" + 0.011*"mine"'),
 (2,
  '0.190*"report" + 0.185*"credit" + 0.055*"information" + 0.048*"remove" + '
  '0.033*"inquiry" + 0.029*"dispute" + 0.026*"inaccurate" + 0.023*"reporting" '
  '+ 0.018*"item" + 0.018*"delete"'),
 (3,
  '0.062*"law" + 0.050*"provide" + 0.032*"require" + 0.032*"verify" + '
  '0.028*"violation" + 0.027*"proof" + 0.026*"act" + 0.026*"legal" + '
  '0.026*"fair" + 0.025*"action"'),
 (4,
  '0.103*"identity" + 0.086*"theft" + 0.070*"balance" + 0.044*"block" + '
  '0.044*"item" + 0.043*"fraudulent" + 0.036*"victim" + 0.032*"file" + '
  '0.031*"information" + 0.028*"talk"'),
 (5,
  '0.135*"bank" + 0.113*"check" + 0.069*"money" + 0.053*"fund" + '
  '0.046*"transaction" + 0.038*"deposit" + 0.038*"transfer" + 0.026*"day" + '
  '0.021*"cash" + 0.018*"debit"'),
 (6,
  '0.126*"loan" + 0.055*"mortgage" + 0.033*"home" + 0.019*"bankruptcy" + '
  '0.018*"year" + 0.015*"sell" + 0.015*"property" + 0.014*"apply" + '
  '0.013*"student" + 0.013*"modification"'),
 (7,
  '0.093*"send" + 0.075*"letter" + 0.067*"request" + 0.055*"receive" + '
  '0.039*"dispute" + 0.039*"complaint" + 0.035*"file" + 0.032*"mail" + '
  '0.028*"state" + 0.027*"document"'),
 (8,
  '0.115*"pay" + 0.101*"charge" + 0.080*"fee" + 0.060*"amount" + '
  '0.056*"interest" + 0.040*"month" + 0.038*"car" + 0.034*"insurance" + '
  '0.031*"balance" + 0.030*"rate"'),
 (9,
  '0.234*"card" + 0.214*"credit" + 0.068*"score" + 0.050*"charge" + '
  '0.041*"fraud" + 0.039*"use" + 0.025*"purchase" + 0.024*"apply" + '
  '0.017*"alert" + 0.017*"limit"'),
 (10,
  '0.225*"debt" + 0.126*"collection" + 0.070*"owe" + 0.038*"company" + '
  '0.035*"collect" + 0.031*"agency" + 0.029*"validation" + 0.022*"collector" + '
  '0.022*"original" + 0.019*"creditor"'),
 (11,
  '0.256*"number" + 0.180*"name" + 0.133*"address" + 0.050*"security" + '
  '0.048*"information" + 0.037*"social" + 0.030*"phone" + 0.021*"personal" + '
  '0.019*"use" + 0.018*"paper"'),
 (12,
  '0.148*"consumer" + 0.072*"information" + 0.025*"code" + 0.021*"agency" + '
  '0.021*"consent" + 0.018*"violation" + 0.018*"past" + 0.016*"prove" + '
  '0.015*"usc" + 0.015*"authorization"'),
 (13,
```

```
 '0.026*"contact" + 0.024*"receive" + 0.021*"make" + 0.019*"issue" + '
 '0.018*"time" + 0.018*"service" + 0.016*"email" + 0.015*"speak" + '
 '0.013*"provide" + 0.012*"state"'),
(14,
 '0.137*"equifax" + 0.122*"chase" + 0.088*"hard" + 0.073*"acct" + '
 '0.060*"datum" + 0.045*"ignore" + 0.039*"unknown" + 0.037*"breach" + '
 '0.030*"hand" + 0.029*"day"'),
(15,
 '0.378*"payment" + 0.119*"late" + 0.076*"make" + 0.044*"due" + 0.043*"pay" + '
 '0.035*"day" + 0.017*"statement" + 0.017*"time" + 0.016*"history" + '
 '0.016*"plan"'),
(16,
 '0.091*"section" + 0.078*"reporting" + 0.075*"right" + 0.064*"state" + '
 '0.063*"consumer" + 0.056*"account" + 0.050*"violate" + 0.045*"agency" + '
 '0.040*"fair" + 0.039*"privacy"'),
(17,
 '0.114*"party" + 0.099*"financial" + 0.077*"third" + 0.054*"requirement" + '
 '0.044*"title" + 0.041*"public" + 0.037*"institution" + 0.037*"employer" + '
 '0.034*"disclose" + 0.030*"disclosure"')]
```

In [ ]:
```
# Compute Perplexity
print('\nPerplexity: ', lda_model.log_perplexity(corpus))  # a measure of how good the model is. lower the better.

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=data_words, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -9.411563903473326
```

In [ ]:
```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim_models.prepare(lda_model, corpus, id2word)
vis
```

Out[ ]:

Selected Topic: 2 Previous Topic Next Topic Clear Topic

Slide to adjust relevance metric:(2)

λ = 1

0.0 0.2 0.4 0.6