Import libraries

```
import subprocess
print((subprocess.check_output("lscpu", shell=True).strip()).decode())
```

```
    Architecture:                    x86_64
    CPU op-mode(s):                  32-bit, 64-bit
    Byte Order:                      Little Endian
    Address sizes:                   46 bits physical, 48 bits virtual
    CPU(s):                          2
    On-line CPU(s) list:             0,1
    Thread(s) per core:              2
    Core(s) per socket:              1
    Socket(s):                       1
    NUMA node(s):                    1
    Vendor ID:                       GenuineIntel
    CPU family:                      6
    Model:                           79
    Model name:                      Intel(R) Xeon(R) CPU @ 2.20GHz
    Stepping:                        0
    CPU MHz:                         2200.150
    BogoMIPS:                        4400.30
    Hypervisor vendor:               KVM
    Virtualization type:             full
    L1d cache:                       32 KiB
    L1i cache:                       32 KiB
    L2 cache:                        256 KiB
    L3 cache:                        55 MiB
    NUMA node0 CPU(s):               0,1
    Vulnerability Itlb multihit:     Not affected
    Vulnerability L1tf:              Mitigation; PTE Inversion
    Vulnerability Mds:               Vulnerable; SMT Host state unknown
    Vulnerability Meltdown:          Vulnerable
    Vulnerability Mmio stale data:   Vulnerable
    Vulnerability Retbleed:          Vulnerable
    Vulnerability Spec store bypass: Vulnerable
    Vulnerability Spectre v1:        Vulnerable: __user pointer sanitization and usercopy barriers only; no swapgs barriers
    Vulnerability Spectre v2:        Vulnerable, IBPB: disabled, STIBP: disabled, PBRSB-eIBRS: Not affected
    Vulnerability Srbds:             Not affected
    Vulnerability Tsx async abort:   Vulnerable
    Flags:                           fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr ss
```

```
!pip install transformers
```

```
#######################################
### -------- Load libraries ------- ###
# Load Huggingface transformers
from transformers import TFBertModel,  BertConfig, BertTokenizerFast
# Then what you need from tensorflow.keras
from tensorflow.keras.layers import Input, Dropout, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers.legacy import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.initializers import TruncatedNormal
from tensorflow.keras.losses import CategoricalCrossentropy
from tensorflow.keras.metrics import CategoricalAccuracy
from tensorflow.keras.utils import to_categorical
# And pandas for data import + sklearn because you allways need sklearn
import pandas as pd
from sklearn.model_selection import train_test_split
```

Loading the Data

```
train_raw = pd.read_csv('consumer_complaints.csv')
train_raw.head()
```

```
<ipython-input-52-c9bd59dabc04>:1: DtypeWarning: Columns (5,11) have mixed
  train_raw = pd.read_csv('consumer_complaints.csv')
```

	date_received	product	sub_product	issue	sub_issue
0	08/30/2013	Mortgage	Other mortgage	Loan modification,collection,foreclosure	Na
1	08/30/2013	Mortgage	Other mortgage	Loan servicing, payments, escrow account	Na
2	08/30/2013	Credit reporting	NaN	Incorrect information on credit report	Accoun status
3	08/30/2013	Student loan	Non-federal student loan	Repaying your loan	Repaying your loar
4	08/30/2013	Debt	Credit card	False statements or	Attemptec to collec

```
train_raw.shape
```

```
(555957, 18)
```

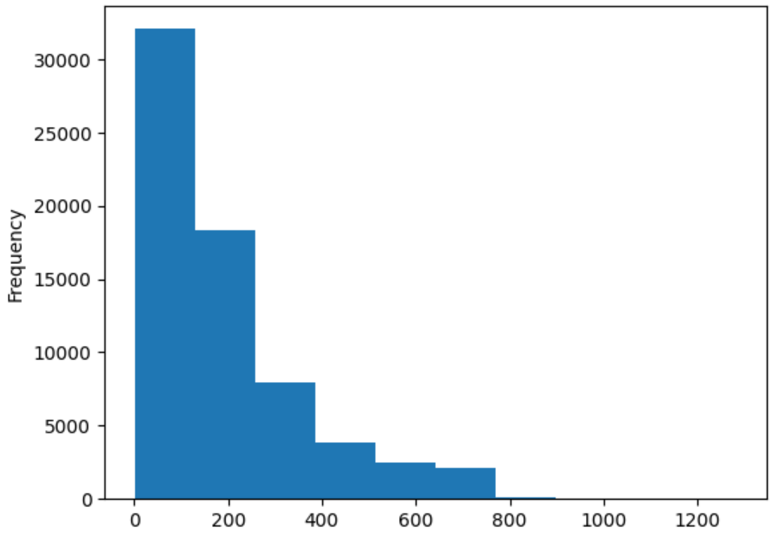
Preprocessing Data

```
train_raw = train_raw[train_raw.consumer_complaint_narrative.notnull()]
train_raw.shape
```

```
(66806, 18)
```

```
train_raw.consumer_complaint_narrative.apply(lambda x: len(x.split())).plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```



```
train_raw['len_txt'] =train_raw.consumer_complaint_narrative.apply(lambda x: len(x.split()))
train_raw.describe()
```

complaint_id	len_txt

```
train_raw = train_raw[train_raw.len_txt >249]
train_raw.shape
```

```
(17142, 19)
```

```
train_raw = train_raw[['consumer_complaint_narrative', 'product']]
train_raw.reset_index(inplace=True, drop=True)
train_raw.head()
```

	consumer_complaint_narrative	product
0	In XX/XX/XXXX my wages that I earned at my job...	Mortgage
1	XXXX was submitted XX/XX/XXXX. At the time I s...	Mortgage
2	I spoke to XXXX of green tree representatives ...	Mortgage
3	i opened XXXX Bank of America credit cards 15-...	Credit card
4	I applied for a loan with XXXX XXXX and had pu...	Consumer Loan

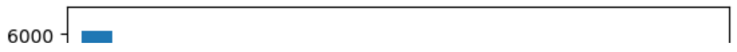
```
train_raw.at[train_raw['product'] == 'Credit reporting', 'product'] = 'Credit reporting, credit repair services, or other persona
train_raw.at[train_raw['product'] == 'Credit card', 'product'] = 'Credit card or prepaid card'
train_raw.at[train_raw['product'] == 'Prepaid card', 'product'] = 'Credit card or prepaid card'
train_raw.at[train_raw['product'] == 'Payday loan', 'product'] = 'Payday loan, title loan, or personal loan'
train_raw.at[train_raw['product'] == 'Virtual currency', 'product'] = 'Money transfer, virtual currency, or money service'
train_raw.head()
```

```
import numpy as np
for l in np.unique(train_raw['product']):
  print(l)
```

```
Bank account or service
Consumer Loan
Credit card
Credit reporting
Debt collection
Money transfers
Mortgage
Other financial service
Payday loan
Prepaid card
Student loan
```

```
train_raw['product'].value_counts().sort_values(ascending=False).plot(kind='bar')
```

```
<Axes: >
```



```
train_raw=train_raw.rename(columns = {'consumer_complaint_narrative':'text', 'product':'label'})
train_raw.head()
```

	text	label
0	In XX/XX/XXXX my wages that I earned at my job...	Mortgage
1	XXXX was submitted XX/XX/XXXX. At the time I s...	Mortgage
2	I spoke to XXXX of green tree representatives ...	Mortgage
3	i opened XXXX Bank of America credit cards 15-...	Credit card
4	I applied for a loan with XXXX XXXX and had pu...	Consumer Loan

```
from sklearn.preprocessing import LabelEncoder

LE = LabelEncoder()
train_raw['label'] = LE.fit_transform(train_raw['label'])
train_raw.head()
```

	text	label
0	In XX/XX/XXXX my wages that I earned at my job...	6
1	XXXX was submitted XX/XX/XXXX. At the time I s...	6
2	I spoke to XXXX of green tree representatives ...	6
3	i opened XXXX Bank of America credit cards 15-...	2
4	I applied for a loan with XXXX XXXX and had pu...	1

```
len(np.unique(train_raw['label']))
```

```
11
```

```
train = train_raw.copy()
```

```
train = train.reindex(np.random.permutation(train.index))
train.head()
```

	text	label
7482	On XXXX XXXX, 2015, I noticed a fraudulent cha...	0
4894	We called Capiital One Bank at XXXX at XX/XX/X...	2
9282	A little over two years ago, I had a Macy 's d...	2
11935	On XXXX/XXXX/2016 I received an email stating ...	9
2290	Dear CFPB, My name is XXXX XXXX and my husband...	0

Clean the text columns

```
import re
def clean_txt(text):
  text = re.sub("'", "",text)
  text=re.sub("(\\W)+"," ",text)
  return text
```

```
train['text']  = train.text.apply(clean_txt)
train.head()
```

	text	label
7482	On XXXX XXXX 2015 I noticed a fraudulent charg...	0

```
from sklearn.model_selection import train_test_split
train, val = train_test_split(train, test_size=0.2, random_state=35)
train.head()
```

	text	label
13143	On XXXX XXXX 2016 I used my Capitol One 360 de...	0
9978	I filed an complaint with Barclays bank of Del...	2
8679	I have been attempting to resolve a dispute wi...	0
12277	I was contacted by Conns on XX XX 2016 to make...	4
12207	Have a loan with One Main financial formerly k...	1

```
train.reset_index(drop=True, inplace=True)
train.head(2)
```

	text	label
0	On XXXX XXXX 2016 I used my Capitol One 360 de...	0
1	I filed an complaint with Barclays bank of Del...	2

```
val.reset_index(drop=True, inplace=True)
val.head(2)
```

	text	label
0	I changed homeowner insurance companies recent...	6
1	The servicer continually claims that the compl...	6

```
val.shape, train.shape
```

```
((3429, 2), (13713, 2))
```

BERT: Use pretrained model to classify product and issue with Complaint Narrative

```
data = pd.read_csv('consumer_complaints.csv')
```

```
<ipython-input-83-0fe41c053794>:1: DtypeWarning: Columns (5,11) have mixed types. Specify dtype option on import or set low_
  data = pd.read_csv('consumer_complaints.csv')
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 555957 entries, 0 to 555956
Data columns (total 18 columns):
 #   Column                        Non-Null Count   Dtype 
---  ------                        --------------   ----- 
 0   date_received                 555957 non-null  object
 1   product                       555957 non-null  object
 2   sub_product                   397635 non-null  object
 3   issue                         555957 non-null  object
 4   sub_issue                     212622 non-null  object
 5   consumer_complaint_narrative  66806 non-null   object
 6   company_public_response       85124 non-null   object
 7   company                       555957 non-null  object
 8   state                         551070 non-null  object
 9   zipcode                       551452 non-null  object
 10  tags                          77959 non-null   object
 11  consumer_consent_provided     123458 non-null  object
 12  submitted_via                 555957 non-null  object
 13  date_sent_to_company          555957 non-null  object
 14  company_response_to_consumer  555957 non-null  object
 15  timely_response               555957 non-null  object
 16  consumer_disputed?            555957 non-null  object
 17  complaint_id                  555957 non-null  int64 
```

```
dtypes: int64(1), object(17)
memory usage: 76.3+ MB
```

```
data.head()
```

	date_received	product	sub_product	issue	sub_issue	con
0	08/30/2013	Mortgage	Other mortgage	Loan modification,collection,foreclosure	NaN	
1	08/30/2013	Mortgage	Other mortgage	Loan servicing, payments, escrow account	NaN	
2	08/30/2013	Credit reporting	NaN	Incorrect information on credit report	Account status	
3	08/30/2013	Student loan	Non-federal student loan	Repaying your loan	Repaying your loan	
4	08/30/2013	Debt collection	Credit card	False statements or representation	Attempted to collect wrong amount	

```
data['product'].nunique()
```

```
11
```

```
data['issue'].nunique()
```

```
95
```

```
#######################################
### --------- Import data --------- ###
# Import data from csv

# Select required columns
data = data[['consumer_complaint_narrative', 'product', 'issue']]
# Remove a row if any of the three remaining columns are missing
data = data.dropna()
# Remove rows, where the label is present only ones (can't be split)
data = data.groupby('issue').filter(lambda x : len(x) > 1)
data = data.groupby('product').filter(lambda x : len(x) > 1)
# Set your model output as categorical and save in new label col
data['Issue_label'] = pd.Categorical(data['issue'])
data['Product_label'] = pd.Categorical(data['product'])
# Transform your output to numeric
data['issue'] = data['Issue_label'].cat.codes
data['product'] = data['Product_label'].cat.codes
# Split into train and test - stratify over Issue
data, data_test = train_test_split(data, test_size = 0.2, stratify = data[['issue']])
```

```
#######################################
### ------- Setup RoBERTa -------- ###
# Name of the BERT model to use
model_name = 'bert-base-uncased'
# Max length of tokens
max_length = 100
# Load transformers config and set output_hidden_states to False
config = BertConfig.from_pretrained(model_name)
config.output_hidden_states = False
# Load BERT tokenizer
tokenizer = BertTokenizerFast.from_pretrained(pretrained_model_name_or_path = model_name, config = config)
# Load the Transformers BERT model
transformer_model = TFBertModel.from_pretrained(model_name, config = config)
```

```
Some layers from the model checkpoint at bert-base-uncased were not used when initializing TFBertModel: ['nsp___cls', 'mlm__
- This IS expected if you are initializing TFBertModel from the checkpoint of a model trained on another task or with anothe
- This IS NOT expected if you are initializing TFBertModel from the checkpoint of a model that you expect to be exactly iden
```

```
All the layers of TFBertModel were initialized from the model checkpoint at bert-base-uncased.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFBertModel for predicti
```

```
#transformer_model.layers[0]
```

```
#######################################
### ------- Build the model ------- ###
# TF Keras documentation: https://www.tensorflow.org/api_docs/python/tf/keras/Model
# Load the MainLayer
bert = transformer_model.layers[0]
# Build your model input
input_ids = Input(shape=(max_length,), name='input_ids', dtype='int32')
attention_mask = Input(shape=(max_length,), name='attention_mask', dtype='int32')
inputs = {'input_ids': input_ids, 'attention_mask': attention_mask}
# Load the Transformers BERT model as a layer in a Keras model
bert_model = bert(inputs)[1]
dropout = Dropout(config.hidden_dropout_prob, name='pooled_output')
pooled_output = dropout(bert_model, training=False)
# Then build your model output
issue = Dense(units=len(data.Issue_label.value_counts()), kernel_initializer=TruncatedNormal(stddev=config.initializer_range), na
product = Dense(units=len(data.Product_label.value_counts()), kernel_initializer=TruncatedNormal(stddev=config.initializer_range)
outputs = {'issue': issue, 'product': product}
# And combine it all in a model object
model = Model(inputs=inputs, outputs=outputs, name='BERT_MultiLabel_MultiClass')
# Take a look at the model
model.summary()
```

```
Model: "BERT_MultiLabel_MultiClass"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to                     
==================================================================================================
 attention_mask (InputLayer)    [(None, 100)]        0           []                               
                                                                                                  
 input_ids (InputLayer)         [(None, 100)]        0           []                               
                                                                                                  
 bert (TFBertMainLayer)         TFBaseModelOutputWi  109482240   ['attention_mask[0][0]',         
                                thPoolingAndCrossAt               'input_ids[0][0]']              
                                tentions(last_hidde                                               
                                n_state=(None, 100,                                               
                                 768),                                                            
                                 pooler_output=(Non                                               
                                e, 768),                                                          
                                 past_key_values=No                                               
                                ne, hidden_states=N                                               
                                one, attentions=Non                                               
                                e, cross_attentions                                               
                                =None)                                                            
                                                                                                  
 pooled_output (Dropout)        (None, 768)          0           ['bert[0][1]']                   
                                                                                                  
 issue (Dense)                  (None, 87)           66903       ['pooled_output[0][0]']          
                                                                                                  
 product (Dense)                (None, 11)           8459        ['pooled_output[0][0]']          
                                                                                                  
==================================================================================================
Total params: 109,557,602
Trainable params: 109,557,602
Non-trainable params: 0
__________________________________________________________________________________________________
```

```
#######################################
### ------- Train the model ------- ###
# Set an optimizer
optimizer = Adam(
    learning_rate=5e-05,
    epsilon=1e-08,
    decay=0.01,
    clipnorm=1.0)
# Set loss and metrics
loss = {'issue': CategoricalCrossentropy(from_logits = True), 'product': CategoricalCrossentropy(from_logits = True)}
metric = {'issue': CategoricalAccuracy('accuracy'), 'product': CategoricalAccuracy('accuracy')}
# Compile the model
model.compile(
    optimizer = optimizer,
    loss = loss,
    metrics = metric)
# Ready output data for the model
y_issue = to_categorical(data['issue'])
```

```
y_product = to_categorical(data['product'])
# Tokenize the input (takes some time)
x = tokenizer(
    text=data['consumer_complaint_narrative'].to_list(),
    add_special_tokens=True,
    max_length=max_length,
    truncation=True,
    padding=True,
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
# Fit the model
history = model.fit(
    x={'input_ids': x['input_ids'] , 'attention_mask': x['attention_mask']},
    y={'issue': y_issue, 'product': y_product},
    validation_split=0.2,
    batch_size=4,
    epochs=1)
```

```
 3787/10689 [=========>....................] - ETA: 10:32:19 - loss: 3.4909 - issue_loss: 2.6464 - product_loss: 0.8445 - is
```

```
#######################################
### ----- Evaluate the model ------ ###
# Ready test data
# test_y_issue = to_categorical(data_test['issue'])
test_y_product = to_categorical(val['label'])
test_x = tokenizer(
    text=val['text'].to_list(),
    add_special_tokens=True,
    max_length=max_length,
    truncation=True,
    padding=True,
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
# Run evaluation
model_eval = model.evaluate(
    x={'input_ids': test_x['input_ids'],'attention_mask': test_x['attention_mask']},
    y={'product': test_y_product}
)
```

```
#model.predict('hello')
```

0s completed at 10:15 AM

Import Packages

```
# Run in python console
import nltk; nltk.download('stopwords')

# Run in terminal or command prompt
# !python3 -m spacy download en
```

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

```
!pip install pyLDAvis
```

```
    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Collecting pyLDAvis
      Downloading pyLDAvis-3.4.0-py3-none-any.whl (2.6 MB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 40.1 MB/s eta 0:00:00
    Requirement already satisfied: numexpr in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (2.8.4)
    Collecting funcy
      Downloading funcy-2.0-py2.py3-none-any.whl (30 kB)
    Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (1.2.2)
    Collecting joblib>=1.2.0
      Downloading joblib-1.2.0-py3-none-any.whl (297 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 298.0/298.0 KB 41.0 MB/s eta 0:00:00
    Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (67.6.1)
    Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (1.10.1)
    Requirement already satisfied: pandas>=1.3.4 in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (1.4.4)
    Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (3.1.2)
    Requirement already satisfied: gensim in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (4.3.1)
    Requirement already satisfied: numpy>=1.22.0 in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (1.22.4)
    Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=1.3.4->pyLDAvis
    Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=1.3.4->pyLDAvis) (2022.7.
    Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=1.0.0->pyLI
    Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.9/dist-packages (from gensim->pyLDAvis) (6.3.0)
    Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from jinja2->pyLDAvis) (2.1.2)
    Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas>=1.3.
    Installing collected packages: funcy, joblib, pyLDAvis
      Attempting uninstall: joblib
        Found existing installation: joblib 1.1.1
        Uninstalling joblib-1.1.1:
          Successfully uninstalled joblib-1.1.1
    ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is
    pandas-profiling 3.2.0 requires joblib~=1.1.0, but you have joblib 1.2.0 which is incompatible.
    Successfully installed funcy-2.0 joblib-1.2.0 pyLDAvis-3.4.0
```

```
import re
import numpy as np
import pandas as pd
from pprint import pprint

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

# spacy for lemmatization
import spacy

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

# Enable logging for gensim - optional
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.ERROR)

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

```
/usr/local/lib/python3.9/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
```

key factors to obtaining good segregation topics:

1. The quality of text processing.
2. The variety of topics the text talks about.
3. The choice of topic modeling algorithm.
4. The number of topics fed to the algorithm.
5. The algorithms tuning parameters.

Prepare Stopwords

```
# NLTK Stop words
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu', 'use'])
```

Import Newsgroups Data

```
# Import Dataset
df = pd.read_json('https://raw.githubusercontent.com/selva86/datasets/master/newsgroups.json')
print(df.target_names.unique())
df.head()
```

```
['rec.autos' 'comp.sys.mac.hardware' 'comp.graphics' 'sci.space'
 'talk.politics.guns' 'sci.med' 'comp.sys.ibm.pc.hardware'
 'comp.os.ms-windows.misc' 'rec.motorcycles' 'talk.religion.misc'
 'misc.forsale' 'alt.atheism' 'sci.electronics' 'comp.windows.x'
 'rec.sport.hockey' 'rec.sport.baseball' 'soc.religion.christian'
 'talk.politics.mideast' 'talk.politics.misc' 'sci.crypt']
```

	content	target	target_names
0	From: lerxst@wam.umd.edu (where's my thing)\nS...	7	rec.autos
1	From: guykuo@carson.u.washington.edu (Guy Kuo)...	4	comp.sys.mac.hardware
2	From: twillis@ec.ecn.purdue.edu (Thomas E Will...	4	comp.sys.mac.hardware
3	From: jgreen@amber (Joe Green)\nSubject: Re: W...	1	comp.graphics
4	From: jcm@head-cfa.harvard.edu (Jonathan McDow...	14	sci.space

Remove emails and newline characters

```
# Convert to list
data = df.content.values.tolist()

# Remove Emails
data = [re.sub('\S*@\S*\s?', '', sent) for sent in data]

# Remove new line characters
data = [re.sub('\s+', ' ', sent) for sent in data]

# Remove distracting single quotes
data = [re.sub("\'", "", sent) for sent in data]

pprint(data[:1])
```

```
['From: (wheres my thing) Subject: WHAT car is this!? Nntp-Posting-Host: '
 'rac3.wam.umd.edu Organization: University of Maryland, College Park Lines: '
 '15 I was wondering if anyone out there could enlighten me on this car I saw '
 'the other day. It was a 2-door sports car, looked to be from the late 60s/ '
 'early 70s. It was called a Bricklin. The doors were really small. In '
 'addition, the front bumper was separate from the rest of the body. This is '
 'all I know. If anyone can tellme a model name, engine specs, years of '
```

```
      'production, where this car is made, history, or whatever info you have on '
      'this funky looking car, please e-mail. Thanks, - IL ---- brought to you by '
      'your neighborhood Lerxst ---- ']
```

Tokenize words and Clean-up text

```
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))

print(data_words[:1])
```

```
[['from', 'wheres', 'my', 'thing', 'subject', 'what', 'car', 'is', 'this', 'nntp', 'posting', 'host', 'rac', 'wam', 'umd', 'e
```

Creating Bigram and Trigram Models

```
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=5, 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)

# See trigram example
print(trigram_mod[bigram_mod[data_words[0]]])
```

```
['from', 'wheres', 'my', 'thing', 'subject', 'what', 'car', 'is', 'this', 'nntp_posting_host', 'rac_wam_umd_edu', 'organizati
```

Remove Stopwords, Make Bigrams and Lemmatize

```
# Define functions for stopwords, bigrams, trigrams and lemmatization
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


# Remove Stop Words
data_words_nostops = remove_stopwords(data_words)

# Form Bigrams
data_words_bigrams = make_bigrams(data_words_nostops)

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

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

```
print(data_lemmatized[:1])
```

```
[['s', 'thing', 'car', 'nntp_poste', 'host', 'rac_wam', 'university', 'park', 'line', 'wonder', 'enlighten', 'car', 'see', '
```

Create the Dictionary and Corpus needed for Topic Modeling

```
# Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)

# Create Corpus
texts = data_lemmatized

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]

# View
print(corpus[:1])
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 5), (5, 1), (6, 2), (7, 1), (8, 1), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1), (14, 1
```

```
id2word[0]
```

```
'addition'
```

```
[[(id2word[id], freq) for id, freq in cp] for cp in corpus[:1]]
```

```
[[('addition', 1),
  ('body', 1),
  ('bring', 1),
  ('call', 1),
  ('car', 5),
  ('day', 1),
  ('door', 2),
  ('early', 1),
  ('engine', 1),
  ('enlighten', 1),
  ('funky', 1),
  ('history', 1),
  ('host', 1),
  ('info', 1),
  ('know', 1),
  ('late', 1),
  ('lerxst', 1),
  ('line', 1),
  ('look', 2),
  ('mail', 1),
  ('make', 1),
  ('model', 1),
  ('name', 1),
  ('neighborhood', 1),
  ('nntp_poste', 1),
  ('park', 1),
  ('production', 1),
  ('rac_wam', 1),
  ('really', 1),
  ('rest', 1),
  ('s', 1),
  ('see', 1),
  ('separate', 1),
  ('small', 1),
  ('spec', 1),
  ('sport', 1),
  ('thank', 1),
  ('thing', 1),
  ('university', 1),
  ('wonder', 1),
  ('year', 1)]]
```

Build the Topic Model

```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                           id2word=id2word,
                                           num_topics=20,
                                           random_state=100,
                                           update_every=1,
                                           chunksize=100,
                                           passes=10,
                                           alpha='auto',
                                           per_word_topics=True)


# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

```
      '0.065*"cost" + 0.059*"model" + 0.039*"character" + 0.036*"picture" + '
      '0.036*"format" + 0.032*"quality" + 0.032*"associate" + 0.028*"handle" + '
      '0.023*"hole" + 0.023*"gift"'),
     (6,
      '0.032*"system" + 0.028*"use" + 0.024*"program" + 0.023*"file" + '
      '0.018*"card" + 0.016*"run" + 0.014*"software" + 0.014*"bit" + '
      '0.013*"machine" + 0.013*"problem"'),
     (7,
      '0.092*"moral" + 0.056*"property" + 0.045*"serial" + 0.036*"lock" + '
      '0.022*"positively" + 0.021*"intent" + 0.018*"alarm" + 0.012*"converter" + '
      '0.011*"unnecessary" + 0.007*"provision"'),
     (8,
      '0.249*"window" + 0.057*"monitor" + 0.055*"normal" + 0.041*"do" + '
      '0.032*"font" + 0.023*"left" + 0.020*"widget" + 0.019*"please_respond" + '
      '0.017*"environment" + 0.017*"trivial"'),
     (9,
      '0.061*"child" + 0.028*"church" + 0.027*"woman" + 0.025*"armenian" + '
      '0.022*"authority" + 0.020*"community" + 0.019*"greek" + 0.017*"period" + '
      '0.017*"turk" + 0.016*"soldier"'),
     (10,
      '0.765*"ax" + 0.035*"physical" + 0.024*"graphic" + 0.014*"direct" + '
      '0.011*"convert" + 0.006*"daughter" + 0.006*"capture" + 0.005*"human_being" '
      '+ 0.004*"split" + 0.003*"accomplish"'),
     (11,
      '0.130*"line" + 0.076*"organization" + 0.074*"write" + 0.063*"article" + '
      '0.056*"nntp_poste" + 0.050*"host" + 0.029*"reply" + 0.024*"thank" + '
      '0.018*"university" + 0.013*"post"'),
     (12,
      '0.072*"plane" + 0.030*"hi" + 0.021*"subscription" + 0.020*"steve" + '
      '0.015*"divide" + 0.011*"evolve" + 0.010*"intersection" + 0.010*"rip" + '
      '0.008*"upcoming" + 0.007*"script"'),
     (13,
      '0.031*"people" + 0.028*"state" + 0.018*"gun" + 0.017*"government" + '
      '0.017*"law" + 0.016*"right" + 0.015*"kill" + 0.013*"death" + 0.011*"live" + '
      '0.011*"force"'),
     (14,
      '0.141*"drug" + 0.029*"film" + 0.026*"movie" + 0.025*"stereo" + '
      '0.024*"japanese" + 0.022*"deficit" + 0.020*"plot" + 0.014*"mad" + '
      '0.009*"harley" + 0.007*"deck"'),
     (15,
      '0.061*"box" + 0.050*"club" + 0.041*"modem" + 0.041*"status" + '
      '0.030*"primary" + 0.029*"routine" + 0.029*"spec" + 0.026*"sufficient" + '
      '0.023*"public_access" + 0.023*"automatically"'),
     (16,
      '0.152*"drive" + 0.091*"car" + 0.036*"bike" + 0.024*"engine" + 0.023*"nhl" + '
      '0.022*"ride" + 0.018*"road" + 0.017*"weight" + 0.016*"mile" + '
      '0.015*"ground"'),
     (17,
      '0.113*"patient" + 0.060*"disease" + 0.054*"scientific" + '
      '0.050*"computer_science" + 0.043*"animal" + 0.041*"health" + '
      '0.040*"treatment" + 0.037*"medical" + 0.033*"dog" + 0.030*"study"'),
     (18,
      '0.023*"get" + 0.018*"go" + 0.015*"good" + 0.015*"time" + 0.015*"know" + '
      '0.014*"make" + 0.013*"well" + 0.013*"think" + 0.012*"see" + 0.010*"take"'),
     (19,
      '0.106*"key" + 0.043*"test" + 0.032*"public" + 0.031*"encryption" + '
      '0.028*"security" + 0.028*"server" + 0.022*"clipper" + 0.021*"chip" + '
      '0.018*"secure" + 0.018*"message"')]
```

Compute Model Perplexity and Coherence Score

```
# 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_lemmatized, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

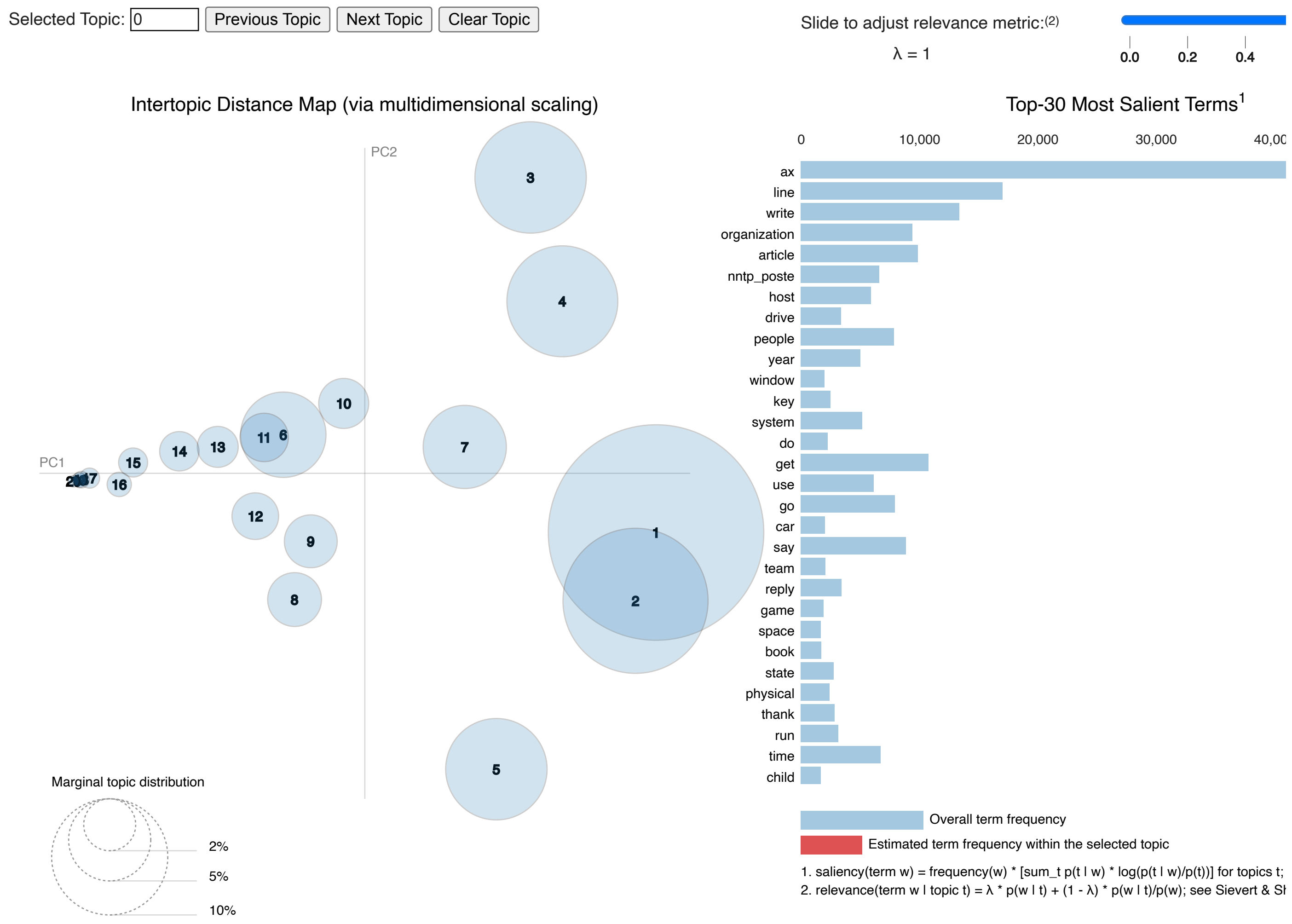
```
    Perplexity:  -13.32461333694394

    Coherence Score:  0.483541481988623
```

Viz the topic-keywords

```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```

```
    /usr/local/lib/python3.9/dist-packages/pyLDAvis/_prepare.py:243: FutureWarning: In a future version of pandas all arguments c
      default_term_info = default_term_info.sort_values(
```



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t;
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Sh

✓ 12s completed at 3:28 PM