LDA : Thanks to machinelearningplus.com for their educational guide on LDA
https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/

```
!pip install pyLDAvis # install then restart runtime
```

```
!pip install -U spacy
```

```
!pip install --upgrade gensim==3.8
```

```
#install Mallet
!wget http://mallet.cs.umass.edu/dist/mallet-2.0.8.zip
!unzip mallet-2.0.8.zip
```

```
#to be run only once
import os       #importing os to set environment variable
def install_java():
  !apt-get install -y openjdk-8-jdk-headless -qq > /dev/null      #install openjdk
  os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"     #set environment variat
  !java -version       #check java version
install_java()
```

```
!python -m spacy download en_core_web_sm
```

```
## Main libraries
import pandas as pd
import numpy as np

# Preprocessing
import gensim
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
import re
from gensim.models.wrappers import LdaMallet

# to upload local files to Google cloud #in case used
from google.colab import files

# Lemmatization
import spacy

import gensim.corpora as corpora

# Plotting tools
import pyLDAvis
```

```
#import pyLDAvis.gensim as gensimvis # with older versions of pyLDAvis

import matplotlib.pyplot as plt
%matplotlib inline
```

Load extracted submissions and comments

```
# use this as many Comments files you have, just change the dataframe number and file name
cdf= pd.read_csv('/content/drive/MyDrive/TMData/comments_ve2y.csv', on_bad_lines='skip')
cdf1= pd.read_csv('/content/drive/MyDrive/TMData/comments_ca1y.csv', on_bad_lines='skip')
```

```
cdf_ve = cdf[cdf['created_utc']> 1606780799].copy() # selecting comments after 1/12/2020 [a
print(cdf_ve.shape)
```

```
# Combine all Comments dataframes in one dataframe
cdf_master = cdf_ve.append([cdf1], ignore_index= True)  #ignore_index= True
```

```
# use this as many Submissions files you have, just change the dataframe number and file na
sdf= pd.read_csv('/content/drive/MyDrive/TMData/submissions_ve1y.csv', on_bad_lines='skip')
sdf1= pd.read_csv('/content/drive/MyDrive/TMData/submissions_ca1y.csv', on_bad_lines='skip
```

```
# Combine all Submissions dataframes in one dataframe
sdf_master = sdf.append([sdf1], ignore_index = True)
```

```
#check number of submissions and comments
print('Total number of submissions' + " " + str(len(sdf_master)))
print('Total number of comments' + " " + str(len(cdf_master)))
```

Deal with removed/deleted submissions and comments

```
#noticed that some "removed" submissions have some text beside the word "removed", so this
sdf_master['selftext'] = sdf_master['selftext'].replace(r"\[removed\].*", value='[removed]
```

```
#Count the number of deleted or removed submissions/comments from the corpus
print(len(sdf_master[(sdf_master['selftext'] == '[removed]') | (sdf_master['selftext'] ==
print(len(cdf_master[(cdf_master['body'] == '[removed]') | (cdf_master['body'] == '[delete
```

```
#Count the number of deleted or removed submissions which has no comments from the corpus
print(len((sdf_master[(((sdf_master['selftext'] == '[removed]') | (sdf_master['selftext'] =
#Count the number of deleted or removed comments from the corpus
print(len(cdf_master[(cdf_master['body'] == '[removed]') | (cdf_master['body'] == '[deletec
```

```
# Remove deleted and removed comments/submissions
sdf_master_clean = sdf_master.loc[(((sdf_master['selftext'] == '[removed]') | (sdf_master[
print('Total number of submissions = ', len(sdf_master_clean))

cdf_master_clean = cdf_master.loc[(cdf_master['body'] != '[removed]') & (cdf_master['body']
print('Total number of comments = ', len(cdf_master_clean))
#for .copy, check https://www.dataquest.io/blog/settingwithcopywarning/ (I used .loc, but i
```

Create documents: each submission and its comments are considered as one document

```
#Just add a space before a comment's body to enhnce readability in next steps
cdf_master_clean['body'] =  ' ' + cdf_master_clean['body']
```

```
#Group comments related to the same submission
cdf_Grouped = cdf_master_clean.groupby('submission_id', as_index=False)['body'].sum()
```

```
#check the total no. of unique submissions, and shape of the grouped comments dataframe
cdf_Grouped.shape
```

```
#check the total no. of submissions in the submissions dataframe
# number of submission could be higher due to submissions which have no comments
sdf_master_clean.shape
```

```
#Combine submissions with their comments to create documents
final_df = sdf_master_clean.merge(cdf_Grouped,  how= 'left',left_on='id', right_on='submiss
```

```
#Check shape and number of documents in the final dataframe
final_df.shape
```

```
#Combine the text of submission and its comments in a single field for future text processi
# Note that both submission's title and text are considered becuase there are a lot of subm
final_df['posts_texts'] = final_df["title"].astype(str) + ' ' + final_df["selftext"].astype
```

```
# Seperate the text for documents
posts_texts= final_df['posts_texts'].copy()
```

Documents Preprocessing

```
#'nan', newline charachter removal, and treatment for VW ID car versions
posts_texts = posts_texts.replace(r'\\n',' ', regex=True) # several newlines charachers wer
posts_texts = posts_texts.replace(r'\b([Nn][Aa][Nn])\b',' ', regex=True) # 'nan' happended
posts_texts = posts_texts.replace(r'([Hh][Aa]){2,}',' ', regex=True) #  remove hahaha relat
posts_texts = posts_texts.replace(r'([lL][Oo][lL]){2,}',' ', regex=True) # remove lollol re
posts_texts = posts_texts.replace(r'(ID4|ID.4)','VWIDIV', regex=True) # to handle Volkswage
posts_texts = posts_texts.replace(r'(ID3|ID.3)','VWIDIII', regex=True) # to handle Volkswag
posts_texts = posts_texts.replace(r'(ID2|ID.2)','VWIDII', regex=True) # to handle Volkswage
posts_texts = posts_texts.replace(r'(ID6|ID.6)','VWIDVI', regex=True) # to handle Volkswage


#check if we have empty documents
len(posts_texts[posts_texts == '']) # answer should be 0 if no empty documents
#posts_texts[0:22]


#removing links
removed_links = posts_texts.replace(r"[\[]?[hH][tT][tT][Pp]\S+", value='', regex=True).copy
#note: added [] to the regex, because there was some cells with only the URL put inside [],



# Reseting index
removed_links.reset_index(drop= True, inplace=True)


#Preparing stopwords based on Gensim set "STOPWORDS"
my_stop_words = STOPWORDS.union(set(['jpg', 'png', 'yes','isn','aren', 'removed','deleted',
my_stop_words = my_stop_words.difference(set(['system','computer','bill','re']))
print(my_stop_words)



#lower case, tokenize posts including removing punctuations, remove very short words(len=1)
# and remove Stop Words
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc).encode('utf-8'), deacc=True) if wo

data_words_nostops = remove_stopwords(removed_links)


# 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'])


def lemmatization(texts, allowed_postags=['NOUN','VERB']): #['NOUN', 'ADJ', 'VERB', 'ADV']
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent)) # I understand this is to convert sent from list to str,
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
        #only allowed post tags are kept and enter lemma
    return texts_out


# Do lemmatization keeping only noun, vb
data_lemmatized = lemmatization(data_words_nostops, allowed_postags=['NOUN','VERB']) #['NOU


# 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:])
#type(corpus)


print('Number of unique tokens: %d' % len(id2word)) #Dictionary
print('Number of documents: %d' % len(corpus))


#Set the path to the Mallet binary
import os
os.environ['MALLET_HOME'] = '/content/mallet-2.0.8'
mallet_path = '/content/mallet-2.0.8/bin/mallet' # you should NOT need to change this


import logging
logging.basicConfig(filename='gensim.log',
                    format="%(asctime)s:%(levelname)s:%(message)s",
                    level=logging.INFO)


ldamallet = gensim.models.wrappers.LdaMallet(mallet_path, corpus=corpus, num_topics=39, id2



from pprint import pprint #pretty print


#Coherence c_v

from gensim.models.coherencemodel import CoherenceModel

# Compute Coherence Score
coherence_model_ldamallet = CoherenceModel(model=ldamallet, texts=data_lemmatized, dictiona
coherence_ldamallet = coherence_model_ldamallet.get_coherence()
```

```
print('\nCoherence Score: ', coherence_ldamallet)
```

```
Topics= ldamallet.show_topics(num_topics=39, num_words=20)       # (formatted=False)
#pprint(ldamallet.show_topics(formatted=False))
```

```
import csv
with open('TopicsGrouped39_2000it.csv','w') as f:
     write = csv.writer(f)
     write.writerows(Topics)
```

```
6#Convert the Mallet model to Gensim format.
gensimmodelMall = gensim.models.wrappers.ldamallet.malletmodel2ldamodel(ldamallet) #ldamal
#https://github.com/polsci/colab-gensim-mallet/blob/master/topic-modeling-with-colab-gensi
```

```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = gensimvis.prepare(gensimmodelMall, corpus, id2word, sort_topics=False) #https://githu
vis
```

```
pyLDAvis.save_html(vis, 'lda39G.html')
```

```
all_topics = gensimmodelMall.get_document_topics(corpus, minimum_probability=0.0)
all_topics_csr = gensim.matutils.corpus2csc(all_topics)
all_topics_numpy = all_topics_csr.T.toarray()
all_topics_df = pd.DataFrame(all_topics_numpy)

#https://stackoverflow.com/questions/46574720/python-gensim-lda-add-the-topic-to-the-docume
#https://radimrehurek.com/gensim/matutils.html#gensim.matutils.corpus2csc
```

```
import seaborn as sns
plt.figure(figsize=(50,50))
svm = sns.heatmap(all_topics_df.corr(method= 'spearman'), annot=True)
figure = svm.get_figure()
figure.savefig('svm_heatmapgrpspear.png', dpi=400)
```