# 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->pyLD
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>=1.3.4
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', 'c
```

## 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)
```

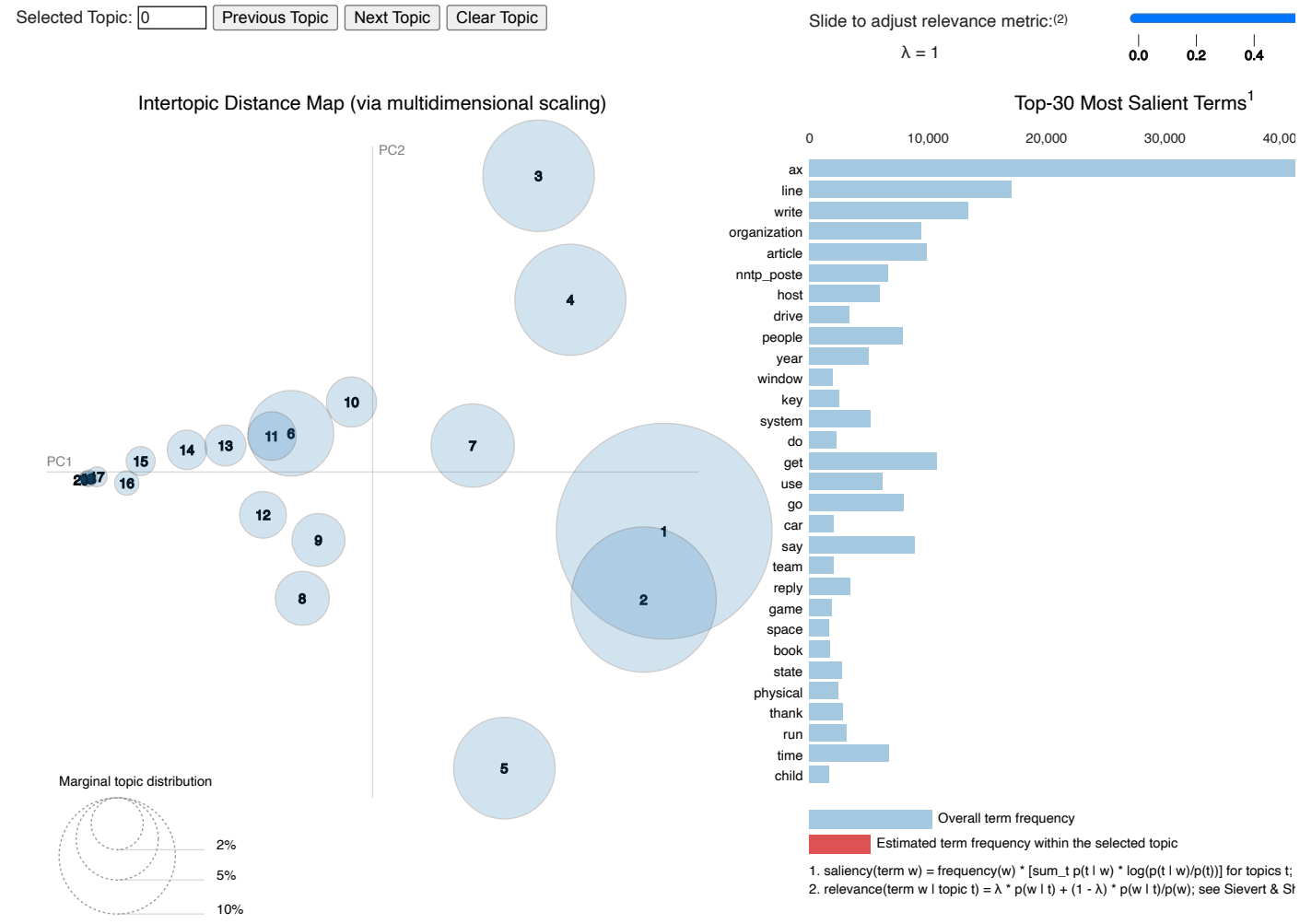
```
Perplexity:  -13.32461333694394

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(
```



✓ 12s completed at 3:28 PM