In [1]:

```
#Loading Metadata
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

In [2]:

```
root_path = 'data/CORD_19_research_challenge'
```

In [3]:

```
metadata_path = f'{root_path}/Covid_19_Dataset.csv'
```

In [13]:

```
df_covid = pd.read_csv(metadata_path, dtype={
    'pubmed_id': str,
    'Microsoft Academic Paper ID': str,
    'doi': str
})
```

In [14]:

```
df_covid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 256684 entries, 0 to 256683
Data columns (total 9 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   Unnamed: 0        256684 non-null  int64 
 1   paper_id          256684 non-null  object
 2   doi               247073 non-null  object
 3   abstract          181579 non-null  object
 4   body_text         256684 non-null  object
 5   authors           254329 non-null  object
 6   title             256680 non-null  object
 7   journal           231637 non-null  object
 8   abstract_summary  256684 non-null  object
dtypes: int64(1), object(8)
memory usage: 17.6+ MB
```

In [15]:

```
#Handle Possible Duplicates
```

In [16]:

```
df_covid.drop_duplicates(['abstract', 'body_text'], inplace=True)
df_covid['abstract'].describe(include='all')
```

Out[16]:

```
count                                                181488
unique                                               180200
top       Publisher's Note Springer Nature remains neutr...
freq                                                    211
Name: abstract, dtype: object
```

In [17]:

```
df_covid['body_text'].describe(include='all')
```

Out[17]:

```
count              256393
unique             256375
top       To the Editor:
freq                    4
Name: body_text, dtype: object
```

In [399]:

```
#sample 400 to df for ease of use
df_sample = df_covid.sample(1000,random_state=50).sort_index(axis=0)
```

In [403]:

```
#del df_covid
```

In [404]:

```
df_sample.dropna(inplace=True)
df_sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 620 entries, 716 to 256426
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Unnamed: 0        620 non-null    int64 
 1   paper_id          620 non-null    object
 2   doi               620 non-null    object
 3   abstract          620 non-null    object
 4   body_text         620 non-null    object
 5   authors           620 non-null    object
 6   title             620 non-null    object
 7   journal           620 non-null    object
 8   abstract_summary  620 non-null    object
dtypes: int64(1), object(8)
memory usage: 48.4+ KB
```

In [405]:

```
from tqdm import tqdm
from langdetect import detect
from langdetect import DetectorFactory

# set seed
DetectorFactory.seed = 0

# hold label - language
languages = []

# go through each text
for ii in tqdm(range(0,len(df_sample))):
    # split by space into list, take the first x intex, join with space
    text = df_sample.iloc[ii]['body_text'].split(" ")

    lang = "en"
    try:
        if len(text) > 50:
            lang = detect(" ".join(text[:50]))
        elif len(text) > 0:
            lang = detect(" ".join(text[:len(text)]))
    # ught... beginning of the document was not in a good format
    except Exception as e:
        all_words = set(text)
        try:
            lang = detect(" ".join(all_words))
        # what!! :( let's see if we can find any text in abstract...
        except Exception as e:

            try:
                # let's try to label it through the abstract then
                lang = detect(df_sample.iloc[ii]['abstract_summary'])
            except Exception as e:
                lang = "unknown"
                pass

    # get the language
    languages.append(lang)
```

```
  0%|          | 0/620 [00:00<?, ?it/s]
  3%|▎         | 21/620 [00:00<00:02, 205.24it/s]
  7%|▋         | 42/620 [00:00<00:02, 199.90it/s]
 10%|▉         | 64/620 [00:00<00:02, 205.83it/s]
 14%|█▎        | 85/620 [00:00<00:02, 198.75it/s]
 17%|█▋        | 105/620 [00:00<00:02, 191.24it/s]
 20%|██        | 125/620 [00:00<00:02, 193.95it/s]
 24%|██▍       | 148/620 [00:00<00:02, 205.23it/s]
 27%|██▋       | 170/620 [00:00<00:02, 209.12it/s]
 31%|███       | 191/620 [00:00<00:02, 204.44it/s]
 34%|███▍      | 212/620 [00:01<00:02, 203.75it/s]
 38%|███▊      | 233/620 [00:01<00:01, 204.27it/s]
 41%|████      | 254/620 [00:01<00:01, 203.86it/s]
 45%|████▍     | 276/620 [00:01<00:01, 207.48it/s]
 48%|████▊     | 298/620 [00:01<00:01, 209.68it/s]
 51%|█████     | 319/620 [00:01<00:01, 208.46it/s]
 55%|█████▍    | 340/620 [00:01<00:01, 202.76it/s]
 58%|█████▊    | 361/620 [00:01<00:01, 204.54it/s]
 62%|██████▏   | 382/620 [00:01<00:01, 203.74it/s]
```

```
 65%|██████▌   | 404/620 [00:01<00:01, 205.68it/s]
 69%|██████▊   | 425/620 [00:02<00:00, 203.39it/s]
 72%|███████   | 446/620 [00:02<00:00, 202.32it/s]
 75%|███████▍  | 467/620 [00:02<00:00, 197.91it/s]
 79%|███████▊  | 490/620 [00:02<00:00, 205.62it/s]
 82%|████████  | 511/620 [00:02<00:00, 201.02it/s]
 86%|████████▍ | 533/620 [00:02<00:00, 205.14it/s]
 90%|████████▊ | 557/620 [00:02<00:00, 214.06it/s]
 93%|█████████ | 579/620 [00:02<00:00, 214.30it/s]
100%|██████████| 620/620 [00:03<00:00, 205.11it/s]
```

In [406]:

```
from pprint import pprint

languages_dict = {}
for lang in set(languages):
    languages_dict[lang] = languages.count(lang)

print("Total: {}\n".format(len(languages)))
pprint(languages_dict)
```

```
Total: 620

{'ca': 1, 'de': 3, 'en': 608, 'es': 5, 'fr': 3}
```

In [407]:

```
#Lets take a look at the language distribution in the dataset
```

In [408]:

```
df_sample['language'] = languages
plt.bar(range(len(languages_dict)), list(languages_dict.values()), align='center')
plt.xticks(range(len(languages_dict)), list(languages_dict.keys()))
plt.title("Distribution of Languages in Dataset")
plt.show()
```

In [409]:

```
#handell only en language paper
df_en = df_sample[df_sample['language'] == 'en']
#del df_en['Unnamed']
```

In [410]:

```
df_en['abstract_word_count'] = df_en['abstract'].apply(lambda x: len(x.strip().split
df_en['body_word_count'] = df_en['body_text'].apply(lambda x: len(x.strip().split())
df_en['body_unique_words']=df_en['body_text'].apply(lambda x:len(set(str(x).split())
#df_en.head()
```

```
<ipython-input-410-c09ae8b1b872>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_en['abstract_word_count'] = df_en['abstract'].apply(lambda x: len
(x.strip().split()))  # word count in abstract
<ipython-input-410-c09ae8b1b872>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_en['body_word_count'] = df_en['body_text'].apply(lambda x: len(x.
strip().split()))  # word count in body
<ipython-input-410-c09ae8b1b872>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_en['body_unique_words']=df_en['body_text'].apply(lambda x:len(set
(str(x).split())))  # number of unique words in body
```

In [411]:

```
df_en.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 608 entries, 716 to 256426
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Unnamed: 0           608 non-null    int64 
 1   paper_id             608 non-null    object
 2   doi                  608 non-null    object
 3   abstract             608 non-null    object
 4   body_text            608 non-null    object
 5   authors              608 non-null    object
 6   title                608 non-null    object
 7   journal              608 non-null    object
 8   abstract_summary     608 non-null    object
 9   language             608 non-null    object
 10  abstract_word_count  608 non-null    int64 
 11  body_word_count      608 non-null    int64 
 12  body_unique_words    608 non-null    int64 
dtypes: int64(4), object(9)
memory usage: 66.5+ KB
```

In [412]:

```
#df_en_sample = df_en.sample(1000,random_state=100).sort.index(axis=0)
```

In [413]:

```
#df_en.to_csv('Covid19_en.csv')
```

In [414]:

```
#Download the spacy bio parser.
#io is used to hide the messy download
```

In [415]:

```
from IPython.utils import io

#with io.capture_output() as captured:
   # !pip install https://s3-us-west-2.amazonaws.com/ai2-s2-scispacy/releases/v0.2.4
```

In [416]:

```
#NLP

import spacy
from spacy.lang.en.stop_words import STOP_WORDS
import en_core_sci_lg
```

In [417]:

```
#Stopwords
```

In [418]:

```
import string
punctuations = string.punctuation
stopwords = list(STOP_WORDS)
stopwords[:10]
```

Out[418]:

```
['which',
 'if',
 'eight',
 'us',
 'however',
 'every',
 'per',
 'third',
 'side',
 'along']
```

In [*]:

```
custom_stop_words = [
    'doi', 'preprint', 'copyright', 'peer', 'reviewed', 'org', 'https', 'et', 'al',
    'rights', 'reserved', 'permission', 'used', 'using', 'biorxiv', 'medrxiv', 'lice
    'al.', 'Elsevier', 'PMC', 'CZI', 'www'
]

for w in custom_stop_words:
    if w not in stopwords:
        stopwords.append(w)
```

In [420]:

```
# Parser
parser = en_core_sci_lg.load(disable=["tagger", "ner"])
```

In [421]:

```
parser.max_length = 5000000
```

In [422]:

```
#tqdm.pandas()
def spacy_tokenizer(sentence):
    mytokens = parser(sentence)
    mytokens = [ word.lemma_.lower().strip() if word.lemma_ != "-PRON-" else word.lo
    mytokens = [ word for word in mytokens if word not in stopwords and word not in
    mytokens = " ".join([i for i in mytokens])
    return mytokens
```

In [423]:

```
from tqdm import tqdm, tqdm_notebook
tqdm_notebook().pandas()
```

```
<ipython-input-423-d6253f1a8a87>:2: TqdmDeprecationWarning: This funct
ion will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  tqdm_notebook().pandas()
```

0/? [00:00<?, ?it/s]

In [*]:

```
df_en["processed_text"] = df_en["body_text"].progress_apply(spacy_tokenizer)
```

100% 606/608 [33:21<00:05, 2.90s/it]

In [425]:

```
import seaborn as sns
```

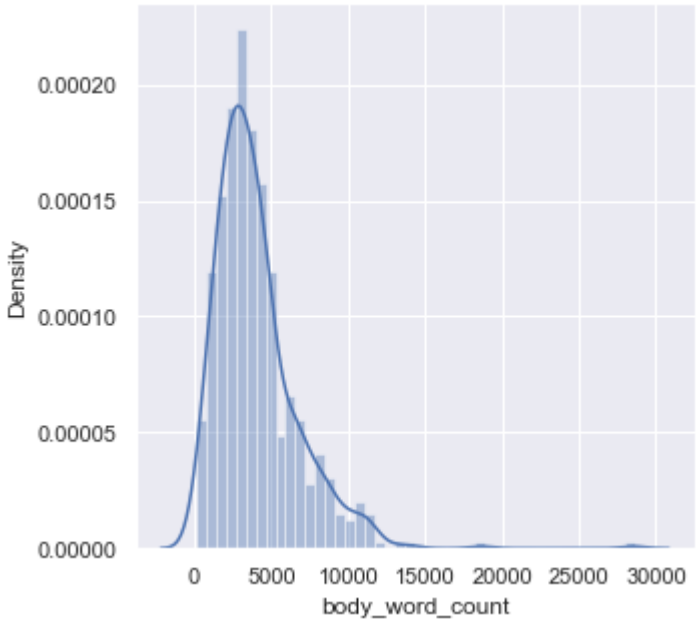
In [426]:

```
sns.distplot(df_en['body_word_count'])
df_en['body_word_count'].describe()
```

```
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/seaborn/distribu
tions.py:2557: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use eit
her `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[426]:

```
count      608.000000
mean      4040.840461
std       2726.981977
min        101.000000
25%       2231.250000
50%       3461.500000
75%       5007.500000
max      28586.000000
Name: body_word_count, dtype: float64
```



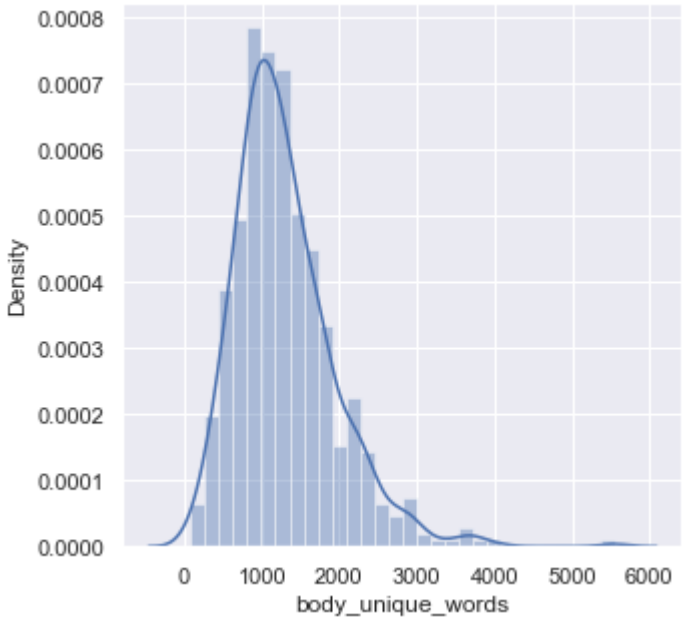
In [427]:

```
sns.distplot(df_en['body_unique_words'])
df_en['body_unique_words'].describe()
```

```
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/seaborn/distribu
tions.py:2557: FutureWarning: `distplot` is a deprecated function and
will be removed in a future version. Please adapt your code to use eit
her `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[427]:

```
count     608.000000
mean     1297.462171
std       648.648664
min        79.000000
25%       866.000000
50%      1186.500000
75%      1639.000000
max      5553.000000
Name: body_unique_words, dtype: float64
```



In [428]:

```
#Vectorization
```

In [429]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
def vectorize(text, maxx_features):

    vectorizer = TfidfVectorizer(max_features=maxx_features)
    X = vectorizer.fit_transform(text)
    return X
```

In [430]:

```
text = df_en['processed_text'].values
X = vectorize(text, 2 ** 12)
X.shape
```

Out[430]:

```
(608, 4096)
```

In [431]:

```
##PCA & Clustering
```

In [434]:

```
from sklearn.decomposition import PCA

pca = PCA(n_components=0.95, random_state=42)
X_reduced= pca.fit_transform(X.toarray())
X_reduced.shape
```

Out[434]:

```
(608, 493)
```

In [435]:

```
#X_reduced
```

In [436]:

```
from sklearn.cluster import KMeans
```

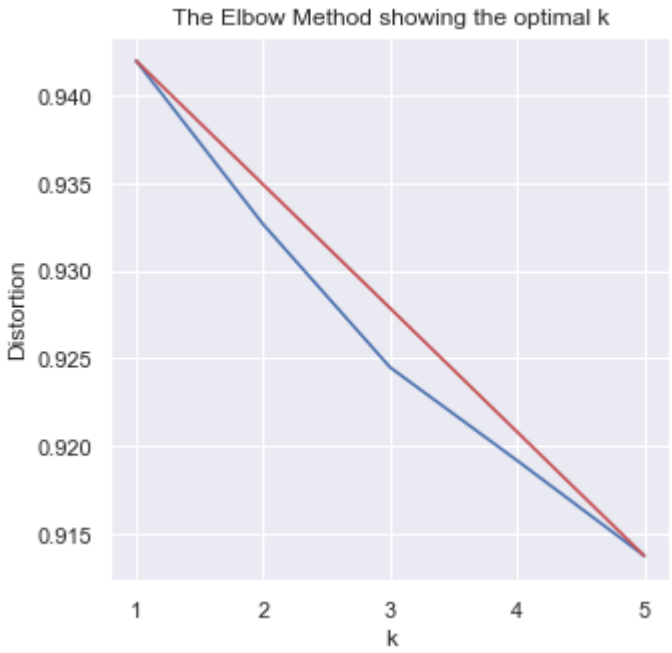
In [437]:

```
from sklearn import metrics
from scipy.spatial.distance import cdist

# run kmeans with many different k
distortions = []
K = range(1, 6)
for k in K:
    k_means = KMeans(n_clusters=k, random_state=20, n_jobs=-1).fit(X_reduced)
    k_means.fit(X_reduced)
    distortions.append(sum(np.min(cdist(X_reduced, k_means.cluster_centers_, 'euclic
    #print('Found distortion for {} clusters'.format(k))
```

```
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
```

In [438]:

```
X_line = [K[0], K[-1]]
Y_line = [distortions[0], distortions[-1]]

# Plot the elbow
plt.plot(K, distortions, 'b-')
plt.plot(X_line, Y_line, 'r')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



In this plot we can see that the better k values are between 3-4. After that, the decrease in distortion is not as significant. For simplicity, we will use k=3

Run k-means

In [570]:

```
k =3
kmeans = KMeans(n_clusters=k, random_state=20, n_jobs=-1)
y_pred = kmeans.fit_predict(X_reduced)
df_en['y'] = y_pred
```

```
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/sklearn/cluster/
_kmeans.py:792: FutureWarning: 'n_jobs' was deprecated in version 0.23
and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
<ipython-input-570-ee2881958671>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  df_en['y'] = y_pred
```

In [571]:

```
#Dimensionality Reduction with t-SNE
```

In [588]:

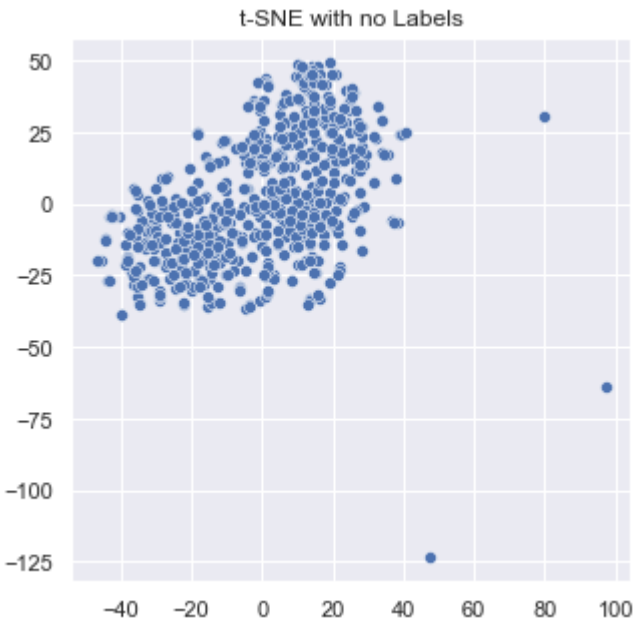
```
from sklearn.manifold import TSNE

tsne = TSNE(verbose=1, perplexity=20, random_state=20)
X_embedded = tsne.fit_transform(X.toarray())
```

```
[t-SNE] Computing 61 nearest neighbors...
[t-SNE] Indexed 608 samples in 0.002s...
[t-SNE] Computed neighbors for 608 samples in 0.029s...
[t-SNE] Computed conditional probabilities for sample 608 / 608
[t-SNE] Mean sigma: 0.291243
[t-SNE] KL divergence after 250 iterations with early exaggeration: 7
9.683456
[t-SNE] KL divergence after 1000 iterations: 1.313160
```

In [589]:

```
from matplotlib import pyplot as plt
import seaborn as sns

# sns settings
sns.set(rc={'figure.figsize':(5,5)})

# colors
palette = sns.color_palette("bright", 1)

# plot
sns.scatterplot(X_embedded[:,0], X_embedded[:,1], palette=palette)
plt.title('t-SNE with no Labels')
#plt.savefig("plots/t-sne_covid19.png")
plt.show()
```

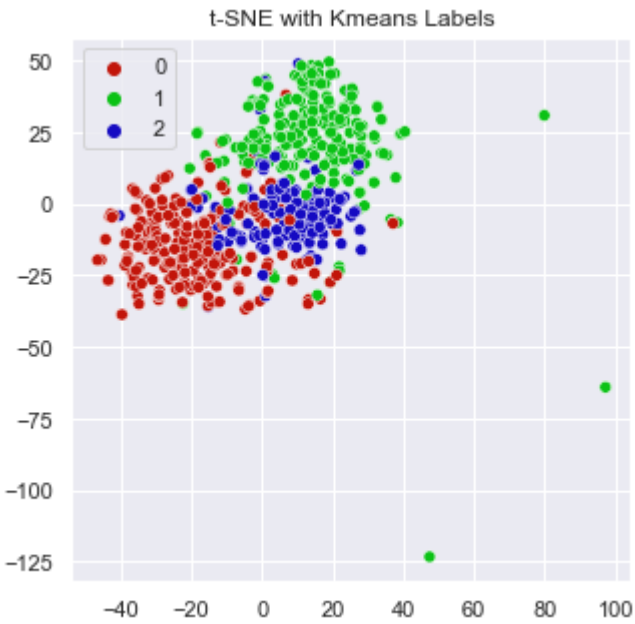
```
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorat
ors.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will r
esult in an error or misinterpretation.
  warnings.warn(
```



In [590]:

```
%matplotlib inline
from matplotlib import pyplot as plt
import seaborn as sns

# sns settings
sns.set(rc={'figure.figsize':(5, 5)})

# colors
palette = sns.hls_palette(3, l=.4, s=.9)

# plot
sns.scatterplot(X_embedded[:,0], X_embedded[:,1], hue=y_pred, legend='full', palette
plt.title('t-SNE with Kmeans Labels')
#plt.savefig("plots/improved_cluster_tsne.png")
plt.show()
```

```
/Users/rana/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorat
ors.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will r
esult in an error or misinterpretation.
  warnings.warn(
```



In [591]:

```
#Topic Modeling on Each Cluster
```

In [592]:

```
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction.text import CountVectorizer
```

In [652]:

```
vectorizers = []

for ii in range(0, 3):
    # Creating a vectorizer
    vectorizers.append(CountVectorizer(min_df=3, max_df=0.9, stop_words='english', l
```

In [653]:

```
vectorizers
```

Out[653]:

```
[CountVectorizer(max_df=0.9, min_df=3, stop_words='english',
                 token_pattern='[a-zA-Z\\-][a-zA-Z\\-]{2,}'),
 CountVectorizer(max_df=0.9, min_df=3, stop_words='english',
                 token_pattern='[a-zA-Z\\-][a-zA-Z\\-]{2,}'),
 CountVectorizer(max_df=0.9, min_df=3, stop_words='english',
                 token_pattern='[a-zA-Z\\-][a-zA-Z\\-]{2,}')]
```

In [654]:

```
vectorizers[0]
```

Out[654]:

```
CountVectorizer(max_df=0.9, min_df=3, stop_words='english',
                token_pattern='[a-zA-Z\\-][a-zA-Z\\-]{2,}')
```

In [655]:

```
#Now we will vectorize the data from each of our clusters

vectorized_data = []

for current_cluster, cvec in enumerate(vectorizers):
    try:
        vectorized_data.append(cvec.fit_transform(df.loc[df['y'] == current_cluster,
    except Exception as e:
        print("Not enough instances in cluster: " + str(current_cluster))
        vectorized_data.append(None)
```

```
Not enough instances in cluster: 0
Not enough instances in cluster: 1
Not enough instances in cluster: 2
```

In [656]:

```
vectorized_data
```

Out[656]:

```
[None, None, None]
```

In [657]:

```
len(vectorized_data)
```

Out[657]:

```
3
```

In [658]:

```
# number of topics per cluster
NUM_TOPICS_PER_CLUSTER = 3


lda_models = []

for ii in range(0, 3):
    # Latent Dirichlet Allocation Model
    lda = LatentDirichletAllocation(n_components=NUM_TOPICS_PER_CLUSTER, max_iter=10
    lda_models.append(lda)

lda_models[0]
```

Out[658]:

```
LatentDirichletAllocation(learning_method='online', n_components=3,
                          random_state=20, verbose=False)
```

In [659]:

```
lda_models
```

Out[659]:

```
[LatentDirichletAllocation(learning_method='online', n_components=3,
                           random_state=20, verbose=False),
 LatentDirichletAllocation(learning_method='online', n_components=3,
                           random_state=20, verbose=False),
 LatentDirichletAllocation(learning_method='online', n_components=3,
                           random_state=20, verbose=False)]
```

In [660]:

```
clusters_lda_data = []

for current_cluster, lda in enumerate(lda_models):
    print("Current Cluster: " + str(current_cluster))

    if vectorized_data[current_cluster] != None:
        clusters_lda_data.append((lda.fit_transform(vectorized_data[current_cluster]
```

```
Current Cluster: 0
Current Cluster: 1
Current Cluster: 2
```

In [661]:

```
clusters_lda_data
```

Out[661]:

```
[]
```

In [662]:

```
# Functions for printing keywords for each topic
def selected_topics(model, vectorizer, top_n=2):
    current_words = []
    keywords = []

    for idx, topic in enumerate(model.components_):
        words = [(vectorizer.get_feature_names()[i], topic[i]) for i in topic.argsor
        for word in words:
            if word[0] not in current_words:
                keywords.append(word)
                current_words.append(word[0])

    keywords.sort(key = lambda x: x[1])
    keywords.reverse()
    return_values = []
    for ii in keywords:
        return_values.append(ii[0])
    return return_values
```

In [663]:

```
#Append list of keywords for a single cluster to 2D list of length NUM_TOPICS_PER_CL


all_keywords = []
for current_vectorizer, lda in enumerate(lda_models):
    print("Current Cluster: " + str(current_vectorizer))

    if vectorized_data[current_vectorizer] != None:
        all_keywords.append(selected_topics(lda, vectorizers[current_vectorizer]))
```

```
Current Cluster: 0
Current Cluster: 1
Current Cluster: 2
```

In [664]:

```
all_keywords
```

Out[664]:

```
[]
```

In [665]:

```
all_keywords[0][:5]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call
 last)
<ipython-input-665-8ffee62a7ad9> in <module>
----> 1 all_keywords[0][:5]

IndexError: list index out of range
```

In [666]:

```
len(all_keywords)
```

Out[666]:

```
0
```

In [667]:

```
#Save current outputs to file
```

In [668]:

```
f=open('lib/topics.txt','w')

count = 0

for ii in all_keywords:

    if vectorized_data[count] != None:
        f.write(', '.join(ii) + "\n")
    else:
        f.write("Not enough instances to be determined. \n")
        f.write(', '.join(ii) + "\n")
    count += 1

f.close()
```

In [669]:

```
import pickle

# save the COVID-19 DataFrame, too large for github
#pickle.dump(df_en, open("plot_data/df_covid.p", "wb" ))

# save the final t-SNE
#pickle.dump(X_embedded, open("plot_data/X_embedded.p", "wb" ))

# save the labels generate with k-means(3)
#pickle.dump(y_pred, open("plot_data/y_pred.p", "wb" ))
```

In [670]:

```
#Classify
```

In [671]:

```
# function to print out classification model report
def classification_report(model_name, test, pred):
    from sklearn.metrics import precision_score, recall_score
    from sklearn.metrics import accuracy_score
    from sklearn.metrics import f1_score

    print(model_name, ":\n")
    print("Accuracy Score: ", '{:,.3f}'.format(float(accuracy_score(test, pred)) * 1
    print("     Precision: ", '{:,.3f}'.format(float(precision_score(test, pred, ave
    print("        Recall: ", '{:,.3f}'.format(float(recall_score(test, pred, averag
    print("      F1 score: ", '{:,.3f}'.format(float(f1_score(test, pred, average='m
```

In [672]:

```
#split the data into train/test sets
from sklearn.model_selection import train_test_split

# test set size of 20% of the data and the random seed 42 <3
X_train, X_test, y_train, y_test = train_test_split(X.toarray(),y_pred, test_size=0.

print("X_train size:", len(X_train))
print("X_test size:", len(X_test), "\n")
```

```
X_train size: 486
X_test size: 122
```

In [673]:

```
#RandomForest
from sklearn.ensemble import RandomForestClassifier

RF_clf = RandomForestClassifier(n_estimators=500,max_depth=3, criterion="entropy",ra
# train RF
RF_clf.fit(X_train, y_train)
# cross validation predictions
RF_pred = cross_val_predict(RF_clf, X_train, y_train, cv=3, n_jobs=-1)
# print out the classification report
classification_report("Random Forest Report (Training Set)", y_train, RF_pred)
```

```
Random Forest Report (Training Set) :

Accuracy Score:  84.156 %
     Precision:  87.846 %
        Recall:  79.457 %
      F1 score:  80.361 %
```

In [676]:

```
# cross validation predictions
RF_pred = cross_val_predict(RF_clf, X_test, y_test, cv=3, n_jobs=-1)

# print out the classification report
classification_report("Random Forest Report (Test Set", y_test, RF_pred)
```

```
Random Forest Report (Test Set :

Accuracy Score:  86.066 %
     Precision:  87.575 %
        Recall:  84.713 %
      F1 score:  85.354 %
```

In [677]:

```
RF_cv_score = cross_val_score(RF_clf, X.toarray(), y_pred, cv=10)
print("Mean cv Score - SGD: {:,.3f}".format(float(RF_cv_score.mean()) * 100), "%")
```

```
Mean cv Score - SGD: 85.194 %
```

In [678]:

```
#Stochastic Gradient Descent classifier
```

In [679]:

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import SGDClassifier

# SGD instance
sgd_clf = SGDClassifier(max_iter=10000, tol=1e-3, random_state=42, n_jobs=-1)
# train SGD
sgd_clf.fit(X_train, y_train)

# cross validation predictions
sgd_pred = cross_val_predict(sgd_clf, X_train, y_train, cv=3, n_jobs=-1)

# print out the classification report
classification_report("Stochastic Gradient Descent Report (Training Set)", y_train,
```

```
Stochastic Gradient Descent Report (Training Set) :

Accuracy Score:  93.827 %
     Precision:  93.785 %
        Recall:  93.342 %
      F1 score:  93.551 %
```

In [680]:

```
# cross validation predictions
sgd_pred = cross_val_predict(sgd_clf, X_test, y_test, cv=3, n_jobs=-1)

# print out the classification report
classification_report("Stochastic Gradient Descent Report (Test Set)", y_test, sgd_p
```

```
Stochastic Gradient Descent Report (Test Set) :

Accuracy Score:  88.525 %
     Precision:  87.937 %
        Recall:  88.133 %
      F1 score:  88.010 %
```

In [681]:

```
sgd_cv_score = cross_val_score(sgd_clf, X.toarray(), y_pred, cv=10)
print("Mean cv Score - SGD: {:,.3f}".format(float(sgd_cv_score.mean()) * 100), "%")
```

```
Mean cv Score - SGD: 93.751 %
```

In [89]:

```
#Plotting the data
```

In [137]:

```
! mkdir lib
! ls
```

```
mkdir: lib: File exists
Applications                                __results___files
BPM_Prework                                 budget_idx.csv
CORD-19-research-challenge                  cleaning.ipynb
COVID-19 Literature Clustering-2.ipynb      data
COVID-19 Literature Clustering-3.ipynb      data.csv
COVID-19 Literature Clustering.ipynb        data.zip
CRIME_DATA.db                               data_2019.csv
Covid_19_Dataset.csv                        en_core_sci_lg-0.2.4
Covid_19_en.csv                             en_core_sci_lg-0.2.4.tar.gz
Desktop                                     en_core_sci_lg-0.4.0.tar.gz
Documents                                   final_project_work.ipynb
Downloads                                   game_project_EDA&modeling.ipyn
b
Feature Engineering_moves.ipynb             get_mta
Google Drive                                get_mta.ipynb
Library                                     htmlpair.ipynb
MTA.db                                      imdb_movies.csv
MTA_19.db                                   lib
Movies                                      metadata.csv
Multiclass ROC.png                          model_movies-Copy1.ipynb
Music                                       model_movies.ipynb
NBM_Classification_Gamma                    mon
NBM_EDA_Gamma                               move_project-2.ipynb
NBM_Regression_Gamma                        move_project-Copy1.ipynb
NYPD_Complaint_Data_Historic-2.csv          move_project.ipynb
Pictures                                    movies.csv
Public                                      movies_f.csv
RANAOR.py                                   mta pair.ipynb
ROC.png                                     mta_data.db
Untitled.ipynb                              mta_pair2.ipynb
Untitled1.ipynb                             mta_project.ipynb
Untitled10.ipynb                            nltk_data
Untitled11.ipynb                            opt
Untitled12.ipynb                            pair_classfication_matrics.ipy
nb
Untitled13.ipynb                            piar_2.ipynb
Untitled14.ipynb                            pire3 mat.ipynb
Untitled15.ipynb                            project_4.ipynb
Untitled2.ipynb                             project_work.ipynb
Untitled3.ipynb                             sal.ipynb
Untitled4.ipynb                             scikit_learn_data
Untitled5.ipynb                             scraping.ipynb
Untitled6.ipynb                             station_location.db
Untitled7.ipynb                             station_location.sqbpro
Untitled8.ipynb                             tds_scraping.ipynb
Untitled9.ipynb                             titanic
Video_Games_Sales_as_at_22_Dec_2016.csv     titanic .ipynb
```

In [138]:

```
! wget /Users/rana/lib/lib/plot_text.py
! wget /Users/rana/lib/lib/call_backs.py
! mv plot_text.py lib/.
! mv call_backs.py lib/.
! ls lib
```

```
zsh:1: command not found: wget
zsh:1: command not found: wget
mv: rename plot_text.py to lib/./plot_text.py: No such file or directo
ry
mv: rename call_backs.py to lib/./call_backs.py: No such file or direc
tory
topics.txt
```

In [ ]:

```

```

In [90]:

```
#pip install bokeh
```

In [91]:

```
# required libraries for plot

import bokeh
from bokeh.models import ColumnDataSource, HoverTool, LinearColorMapper, CustomJS, S
from bokeh.palettes import Category20
from bokeh.transform import linear_cmap, transform
from bokeh.io import output_file, show, output_notebook
from bokeh.plotting import figure
from bokeh.models import RadioButtonGroup, TextInput, Div, Paragraph
from bokeh.layouts import column, widgetbox, row, layout
from bokeh.layouts import column
```

In [92]:

```
from lib.plot_text import header, description, description2, cite, description_searc
from lib.call_backs import input_callback, selected_code
```

```
---------------------------------------------------------------------------
-----
ModuleNotFoundError                       Traceback (most recent call
 last)
<ipython-input-92-9c0382ed7af7> in <module>
----> 1 from lib.plot_text import header, description, description2, c
ite, description_search, description_slider, notes, dataset_descriptio
n, toolbox_header
      2 from lib.call_backs import input_callback, selected_code

ModuleNotFoundError: No module named 'lib.plot_text'
```

In [93]:

```
#Load the Keywords per Cluster
```

In [94]:

```
import os

topic_path = os.path.join(os.getcwd(), 'lib', 'topics.txt')
with open(topic_path) as f:
    topics = f.readlines()
```

In [95]:

```
#Setup
```

In [98]:

```
# show on notebook
output_notebook()
# target labels
y_labels = y_pred

# data sources
source = ColumnDataSource(data=dict(
    x= X_embedded[:,0],
    y= X_embedded[:,1],
    x_backup = X_embedded[:,0],
    y_backup = X_embedded[:,1],
    desc= y_labels,
    titles= df_en['title'],
    authors = df_en['authors'],
    journal = df_en['journal'],
    abstract = df_en['abstract_summary'],
    labels = ["C-" + str(x) for x in y_labels],
    links = df_en['doi']
    ))

# hover over information
hover = HoverTool(tooltips=[
    ("Title", "@titles{safe}"),
    ("Author(s)", "@authors{safe}"),
    ("Journal", "@journal"),
    ("Abstract", "@abstract{safe}"),
    ("Link", "@links")
],
point_policy="follow_mouse")

# map colors
initial_palette = Category20[20]
random.Random(20).shuffle(initial_palette)

mapper = linear_cmap(field_name='desc',
                     palette=Category20[20],
                     low=min(y_labels) ,high=max(y_labels))

# prepare the figure
plot = figure(plot_width=1200, plot_height=850,
           tools=[hover, 'pan', 'wheel_zoom', 'box_zoom', 'reset', 'save', 'tap'],
           title="Clustering of the COVID-19 Literature with t-SNE and K-Means",
           toolbar_location="above")

# plot settings
plot.scatter('x', 'y', size=5,
          source=source,
          fill_color=mapper,
          line_alpha=0.3,
          line_width=1.1,
          line_color="black",
          legend = 'labels')
plot.legend.background_fill_alpha = 0.6
```

(https://bokeh.org/) BokehJS 2.3.2 successfully loaded.

```
NameError                                 Traceback (most recent call
 last)
<ipython-input-98-add44087c183> in <module>
     31 # map colors
     32 initial_palette = Category20[20]
---> 33 random.Random(20).shuffle(initial_palette)
     34
     35 mapper = linear_cmap(field_name='desc',

NameError: name 'random' is not defined
```

In [99]:

```
#Widgets
```

In [101]:

```
# Keywords
text_banner = Paragraph(text= 'Keywords: Slide to specific cluster to see the keywor
input_callback_1 = input_callback(plot, source, text_banner, topics)

# currently selected article
div_curr = Div(text="""Click on a plot to see the link to the article.""",height=150
callback_selected = CustomJS(args=dict(source=source, current_selection=div_curr), c
taptool = plot.select(type=TapTool)
taptool.callback = callback_selected

# WIDGETS
slider = Slider(start=0, end=20, value=20, step=1, title="Cluster #", callback=input
keyword = TextInput(title="Search:", callback=input_callback_1)

# pass call back arguments
input_callback_1.args["text"] = keyword
input_callback_1.args["slider"] = slider
# column(,,widgetbox(keyword),,widgetbox(slider),, notes, cite, cite2, cite3), plot
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
<ipython-input-101-b8b3f3225b1c> in <module>
      1 # Keywords
      2 text_banner = Paragraph(text= 'Keywords: Slide to specific clu
ster to see the keywords.', height=25)
----> 3 input_callback_1 = input_callback(plot, source, text_banner, t
opics)
      4
      5 # currently selected article

NameError: name 'input_callback' is not defined
```

In [102]:

```
#Style
```

In [103]:

```
# STYLE
header.sizing_mode = "stretch_width"
header.style={'color': '#2e484c', 'font-family': 'Julius Sans One, sans-serif;'}
header.margin=5

description.style ={'font-family': 'Helvetica Neue, Helvetica, Arial, sans-serif;',
description.sizing_mode = "stretch_width"
description.margin = 5

description2.sizing_mode = "stretch_width"
description2.style ={'font-family': 'Helvetica Neue, Helvetica, Arial, sans-serif;',
description2.margin=10

description_slider.style ={'font-family': 'Helvetica Neue, Helvetica, Arial, sans-se
description_slider.sizing_mode = "stretch_width"

description_search.style ={'font-family': 'Helvetica Neue, Helvetica, Arial, sans-se
description_search.sizing_mode = "stretch_width"
description_search.margin = 5

slider.sizing_mode = "stretch_width"
slider.margin=15

keyword.sizing_mode = "scale_both"
keyword.margin=15

div_curr.style={'color': '#BF0A30', 'font-family': 'Helvetica Neue, Helvetica, Arial
div_curr.sizing_mode = "scale_both"
div_curr.margin = 20

text_banner.style={'color': '#0269A4', 'font-family': 'Helvetica Neue, Helvetica, Ar
text_banner.sizing_mode = "scale_both"
text_banner.margin = 20

plot.sizing_mode = "scale_both"
plot.margin = 5

dataset_description.sizing_mode = "stretch_width"
dataset_description.style ={'font-family': 'Helvetica Neue, Helvetica, Arial, sans-s
dataset_description.margin=10

notes.sizing_mode = "stretch_width"
notes.style ={'font-family': 'Helvetica Neue, Helvetica, Arial, sans-serif;', 'font-
notes.margin=10

cite.sizing_mode = "stretch_width"
cite.style ={'font-family': 'Helvetica Neue, Helvetica, Arial, sans-serif;', 'font-s
cite.margin=10

r = row(div_curr,text_banner)
r.sizing_mode = "stretch_width"
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
<ipython-input-103-b71ffbeb745e> in <module>
      1 # STYLE
```

```
----> 2 header.sizing_mode = "stretch_width"
      3 header.style={'color': '#2e484c', 'font-family': 'Julius Sans
 One, sans-serif;'}
      4 header.margin=5
      5

NameError: name 'header' is not defined
```

In [104]:

```
#SHOW
```

In [105]:

```
# LAYOUT OF THE PAGE
l = layout([
    [header],
    [description],
    [description_slider, description_search],
    [slider, keyword],
    [text_banner],
    [div_curr],
    [plot],
    [description2, dataset_description, notes, cite],
])
l.sizing_mode = "scale_both"


# show
output_file('plots/t-sne_covid-19_interactive.html')
show(l)
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
<ipython-input-105-b7ee4806ee32> in <module>
      1 # LAYOUT OF THE PAGE
      2 l = layout([
----> 3     [header],
      4     [description],
      5     [description_slider, description_search],

NameError: name 'header' is not defined
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```