CORD19_report

April 29, 2020

COVID-19 Open Research Dataset Challenge (CORD-19)

Authors: Chao Zhou, Ruijin Jia, Matteo Bucalossi

Machine Learning I (DATS 6202), Spring 2020

GitHub Repository

0.0.1 Introduction

Last month, in response to the global COVID-19 pandemic, the White House and other leading research institutions, including the Allen Institute for AI, have prepared the COVID-19 Open Research Dataset (CORD-19): this dataset includes over 57,000 scholarly articles, most of them with full text, about COVID-19, SARS-CoV-2, and related coronaviruses. As literature published keeps increasing in scale, the medical community would need AI tools to quickly gain insights and directions to fight this disease in a timely manner.
Kaggle has issued a competition to develop such tools to help the medical community for this high priority scientific challenge. The public dataset represents a highly readable and clean collection of materials, and it is beeing updated constantly as more publications are released and added to the corpus.

Thanks to the minimal work on pre-processing required by this dataset, we decided to apply a Transformer model to extract sentence embeddings for machine learning task-specific "heads". Given that the pre-trained transformer by the UKPLab uses BERT, we decided to train and fine-tune the same transformer on SciBERT by Allen AI to obtain more relevant embeddings for a scientific and medical corpus.
We then used said embeddings to train different clustering algorithms for unsupervised learning: we used UAMP for dimensionality reduction, and HDBSCAN for density-based clustering while LDA for probability-based clustering. Also, we integrated a semantic search algorithm based on cosine similarity that can take user questions (we recommend to take ideas from Tasks) and provides the top 5 relevant articles from the corpus.

1 Data Preprocessing

Luckily the dataset had already been cleaned for the most part, and we only had to create an hashable corpus of text from json files. Once we had the body text and the abstact for each article, and its relevant metadata, in a dataframe, we cleaned the text columns of odd characters using regex. Here we call the preprocessing script to obtain a clean dataframe to use for our machine learning tasks.

[1]:
```
import numpy as np
import pandas as pd
import os
import glob
import json
# import scispacy
import spacy
import tqdm
import matplotlib.pyplot as plt

from scripts import preprocessing
```

[3]:
```
######################################testing##################################
# directories and paths
root_path = os.getcwd()
metadata_path = f'{root_path}/data/all_sources_metadata_2020-03-13.csv'
metadata = pd.read_csv(metadata_path)
metadata.head()
metadata.info()
# read all paths of json file(paper)
all_json = glob.glob(f'{root_path}/data/**/*.json', recursive=True)
print(len(all_json))
######################################testing##################################
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29500 entries, 0 to 29499
Data columns (total 14 columns):
sha                              17420 non-null object
source_x                         29500 non-null object
title                            29130 non-null object
doi                              26357 non-null object
pmcid                            27337 non-null object
pubmed_id                        16730 non-null float64
license                          17692 non-null object
abstract                         26553 non-null object
publish_time                     18248 non-null object
authors                          28554 non-null object
journal                          17791 non-null object
Microsoft Academic Paper ID      1134 non-null float64
WHO #Covidence                   1236 non-null object
has_full_text                    17420 non-null object
dtypes: float64(2), object(12)
memory usage: 3.2+ MB
13202
```

[3]:
```
preprocessing.nan_checker(metadata)
```

[3]:
```
                          var  proportion    dtype
0   Microsoft Academic Paper ID    0.961559  float64
1                WHO #Covidence    0.958102   object
2                     pubmed_id    0.432881  float64
3                           sha    0.409492   object
4                 has_full_text    0.409492   object
5                       license    0.400271   object
6                       journal    0.396915   object
7                  publish_time    0.381424   object
8                           doi    0.106542   object
9                      abstract    0.099898   object
10                        pmcid    0.073322   object
11                      authors    0.032068   object
12                        title    0.012542   object
```

[4]:
```
# drop the rows where abstract is missing
metadata.dropna(subset = ['abstract'],axis = 0, inplace = True)
#check dimension of new dataset
print(metadata.shape)
```

```
(26553, 14)
```

[5]:
```
# remove dupicate with same title
metadata.drop_duplicates(subset ="title", keep = False, inplace = True)
#check dimension of new dataset
print(metadata.shape)
metadata.head()
```

```
(20224, 14)
```

[5]:
```
                                         sha source_x  \
2   210a892deb1c61577f6fba58505fd65356ce6636      CZI
3   e3b40cc8e0e137c416b4a2273a4dca94ae8178cc      CZI
6   f24242580be243d5fc3f432915d86af6854bb8b7      CZI
8   e1b336d8be1a4c0ccc5a1bf41e48b3b004d3ece1      CZI
10  469ed0f00c09e2637351c9735c306f27acf3aace      CZI

                                                title  \
2   Incubation Period and Other Epidemiological Ch…
3   Characteristics of and Public Health Responses…
6   Real-time forecasts of the 2019-nCoV epidemic …
8   COVID-19 outbreak on the Diamond Princess crui…
10  First two months of the 2019 Coronavirus Disea…

                         doi pmcid   pubmed_id      license  \
2         10.3390/jcm9020538   NaN         NaN        cc-by
3         10.3390/jcm9020575   NaN  32093211.0        cc-by
```

```
6    10.1016/j.idm.2020.02.002   NaN         NaN  cc-by-nc-nd
8          10.1093/jtm/taaa030   NaN         NaN     cc-by-nc
10  10.1186/s41256-020-00137-4   NaN         NaN        cc-by

                                             abstract publish_time  \
2   The geographic spread of 2019 novel coronaviru…         2020
3   In December 2019, cases of unidentified pneumo…         2020
6   The initial cluster of severe pneumonia cases …         2020
8   Cruise ships carry a large number of people in…         2020
10  Similar to outbreaks of many other infectious …         2020

                                              authors  \
2   Linton, M. Natalie; Kobayashi, Tetsuro; Yang, …
3                     Deng, Sheng-Qun; Peng, Hong-Juan
6   Roosa, K.; Lee, Y.; Luo, R.; Kirpich, A.; Roth…
8           Rocklöv, J.; Sjödin, H.; Wilder-Smith, A.
10                             Chen, Xinguang; Yu, Bin

                             journal  Microsoft Academic Paper ID  \
2       Journal of Clinical Medicine                 3.006065e+09
3                        J Clin Med                  1.776631e+08
6       Infectious Disease Modelling                 3.006029e+09
8         Journal of Travel Medicine                 3.006304e+09
10  Global Health Research and Policy                3.006646e+09

   WHO #Covidence has_full_text
2           #1043          True
3           #1999          True
6            #865          True
8           #2926          True
10          #5595          True
```

[6]:
```
# read json files
first_row = preprocessing.FileReader(all_json[0])
print(first_row)
```

```
8f8eb4f004c2002face0723f2f58cc411954d36e: Bordetella bronchiseptica isolate KM22
has been used in experimental infections of swine as a model of clinical B.
bronchiseptica infection and to study host-to-host transmission. The draft
genome sequence of KM22 was reported in 2014. Here, we report the complete
genome sequence of KM22… 20-kb insert library preparation protocol
(https://www.pacb.com/wp-content/uploads/ Procedure-Checklist-20-kb-Template-
Preparation-Using-BluePippin-Size-Selection -System.pdf). The 20-kb library was
sequenced with a PacBio RS II platform using two single-molecule real-time
(SMRT) cells, resulting in 283,436 total reads and an average read length of
7,600 bp. Reads were subsequently assessed for quality using FastQC
(https://www.bioinformatics.babraham.ac.uk/projects/fastqc/).
```

```
Whole-genome assemb…
```

[7]:
```
# build dataframe
df_covid = preprocessing.read_directory_files(all_json, metadata)
df_covid.head()
```

[7]:
```
                                   paper_id  \
0  8f8eb4f004c2002face0723f2f58cc411954d36e
1  63f7049d200896290b38b38711113054f7ea1b50
2  4df45b8404d9de0b376a8ae3c282a517df36fe51
3  3c3572ba243d61e7631725669c8f88347fdbd5bc
4  4cb9c6ef889605b3149ab8b59c8258074067ba04

                                            abstract  \
0  Bordetella bronchiseptica isolate KM22 has bee…
1
2  The influenza A nucleoprotein (NP) is an attra…
3  The prevalence of feline herpesvirus-1 (FHV-1)…
4  The epidemic of severe acute respiratory syndr…

                                           body_text  \
0  20-kb insert library preparation protocol (htt…
1  I nfectious diseases have been an ever-present…
2  The transmission of a pathogenic avian H5N1 vi…
3  Feline herpesvirus type 1 (FHV-1) is the most …
4  E merging diseases are frequently zoonoses cau…

                                             authors  \
0  ['Nicholson, Tracy L.', 'Bayles, Darrell O.', …
1  ['Fauci, Anthony S.', 'Touchette, Nancy A.', '…
2  ['Cheung, Ying-Kit', 'Cheng, Samuel Chak-Sum',…
3           ['Kang, Byeong-Teck', 'Park, Hee-Myung']
4  ['Dominguez, Samuel R.', 'O’Shea, Thomas J.', …

                                               title                 journal
0  Complete Genome Sequence of Bordetella bronchi…  Microbiol Resour Announc
1  Emerging Infectious Diseases: a 10-Year Perspe…         Emerg Infect Dis
2  Two novel HLA-A*0201 T-cell epitopes in avian …                  Vet Res
3  Prevalence of feline herpesvirus 1, feline cal…                J Vet Sci
4  Detection of Group 1 Coronaviruses in Bats in …         Emerg Infect Dis
```

[8]:
```
# clean abstract and body_text
cleaned_abstract = []
for item in df_covid['abstract']:
    item = preprocessing.clean_text(item)
    cleaned_abstract.append(item)
df_covid['abstract'] = cleaned_abstract
```

```
#clean body_text
cleaned_body = []
for item in df_covid['body_text']:
    item = preprocessing.clean_text(item)
    cleaned_body.append(item)
df_covid['body_text'] = cleaned_body
```

[9]:
```
pd.DataFrame([[df_covid.shape[0], df_covid.shape[1]]], columns=['# rows', '#␣
↪columns'])
```

[9]:
```
   # rows  # columns
0    9657          6
```

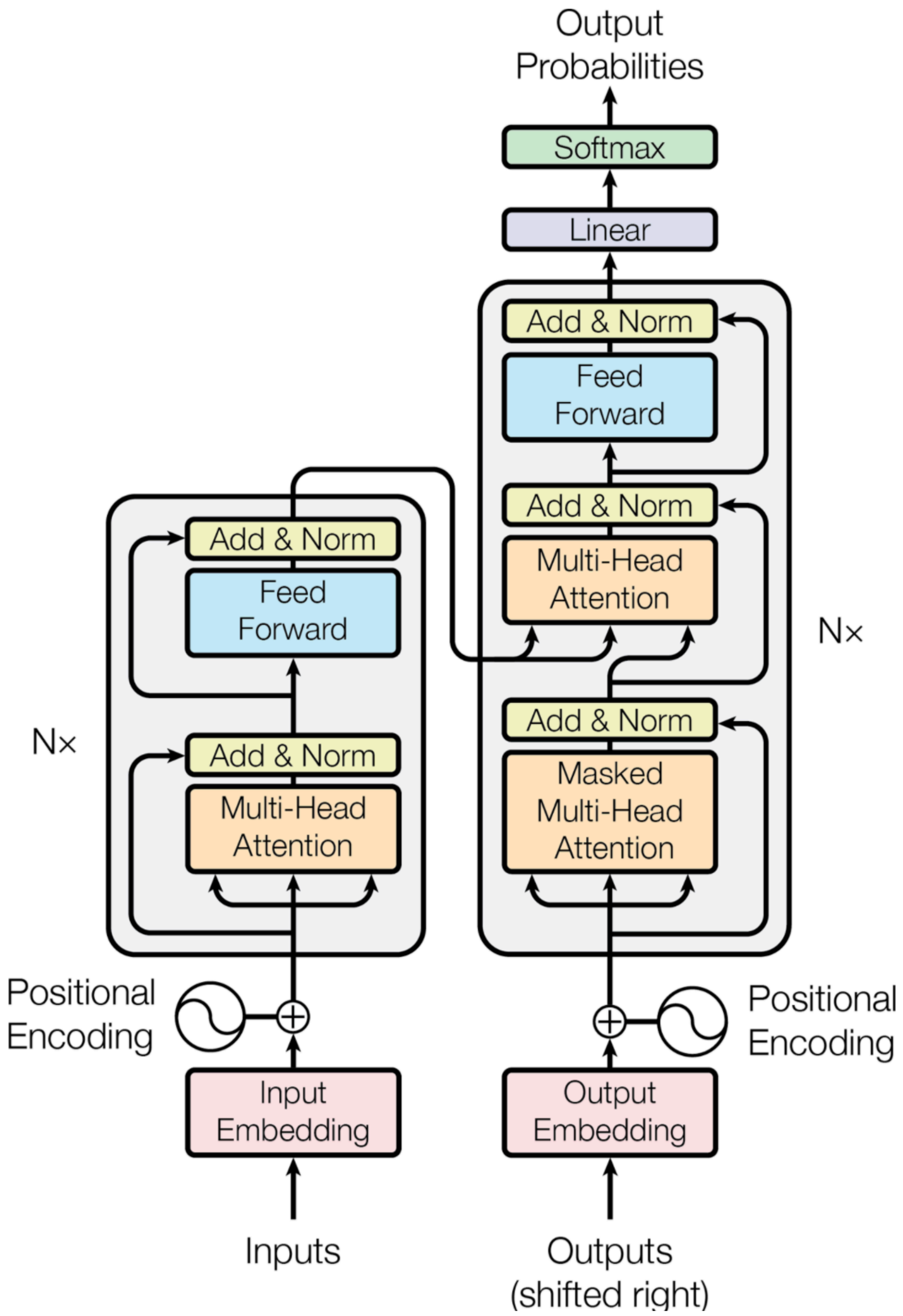
[10]:
```
# save dataframe
df_covid.to_pickle(f'{root_path}/data/sample_preprocessed_dataframe.pkl')
```

2 Train Sentence Transformer on sciBERT

2.0.1 Transformers

When it comes to natural language processing, the most recent developments have seen attention mechanisms prevailing versus more traditiona RNN models. These models use an architecture called Transformer, much faster and easier to parallelize than other networks. This is where these models revolutionize the field: an attention mechanism looks at an input sequence and decides at each step which other components of the sequence are important1 - meaning it can replicate the way we actually process text, i.e. not only focusing on single words but also considering what's around it to make sense of the language.

Transformers are architectures for transforming a sequence into another by using Encoder and Decoder; yet, they only imply attention mechanisms without any Recurrent Networks (previously the go-to models for many NLP tasks). Here's an image2 to illustrate such Transformer architecture, with the Encoder part on the left and the Decoder on the right.



We won't bore you with the details of the model and its mathematical aspects, but we can point out two main characteristics of Transformers: - the Multi-Head Attention layers treat each word's relationship with every other word in the same sentence, basically paying attention to more words than just one when processing sequences. These layers will apply to every (input/target, depending on encoder/decoder) sentence the following equation:

$$\text{Attention}\,(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\intercal}}{\sqrt{d_{keys}}}\right)\mathbf{V}$$

- the positional encodings of words are dense vectors (some extra word embeddings) representing the position of a word within the sentencem and are added to each word's embeddings.

2.0.2 BERT & SciBERT

A 2018 paper[3] published by various Google researchers brought to life a nowadays state-of-the-art application of a Transformer-based architectures for self-supervised pretraining on large corpus, BERT (*Bidirectional Encoder Representations from Transformers*). BERT is a method of pre-training language representations, so that we can train a general-purpose model on an immense corpus and then use said model for downstream NLP tasks - its bidirectional characteristic allows to represent each word within its context (i.e. other words in the sentence, both on the left and right of represented word).

If the original BERT trained a large (12-layer to 24-layer) Transformer on a large corpuse of Wikipedia and BookCorpus, more recent BERT-alike models have trained the same architecture on specific corpuses for domain-relevant tasks. For instance, in 2019 Allen AI released SciBERT,[4] a BERT model trained on huge corpus of scientific papers (82% of which from biomedical domain) from semanticscholar.org, which significantly improves BERT performance on downstream NLP tasks specific to scientific problems. We believe that using SciBERT for this project will yield much better results than the generic original BERT model given the specificity of our data.

2.0.3 Sentence-BERT

BERT-alike models described above have set a new bar for sentence-pair regressions tasks, but unfortunately they need both sentences to be fed in the Transformer, causing such a computational overhead that makes unsupervised tasks virtually impossible.
Thus, in 2019 researchers at UKPLab released SBERT,[5] a modification of pre-trained BERT to derive semantic sentence embeddings easily comparable for similarity and clustering. SBERT fine-tunes BERT-alike models with a siamese or triplet network structure to obtain such embeddings. Such a revolutionary paper proposed a model that still maintains BERT-level accuracy, while scaling down time complexity from 65 hours to 5 seconds for specific unsupervised NLP tasks, including similarity comparison, semantic search and clustering. (ah! exactly what we are trying to do here!)
Here's an illustration of the SBERT architecture as described in the paper (the structure would not change if the objective function was different, as it may be for different tasks):

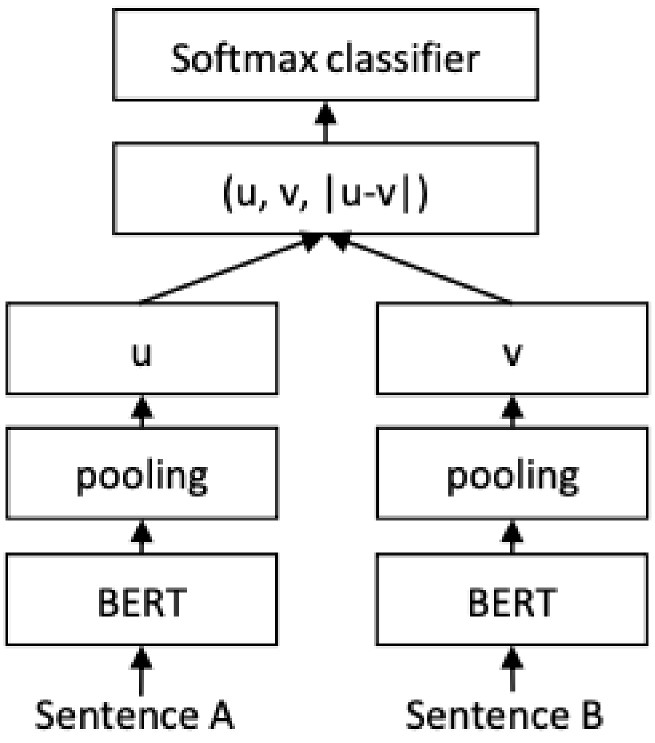


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

3 Sentence embeddings for analysis

Now, we intend to perform some clustering as well as semantic search on the CORD-19 dataset. To do so, we need SBERT sentence embeddings so that we can accomplish these tasks in reasonable time. We decided that pre-trained embeddings on BERT would have not provide the SOTA outcome we were looking for, thus we fine-tuned our sentence embedding method on SciBERT to get science-specific sentence embeddings.
The SBERT library provides the code to tune any BERT-like model (in our case, SciBERT) on the Natural Language Inference (NLI) data, published by Allen AI. The following script trained SciBERT on the NLI dataset using a Softmax Classifier as training loss and the STS (Semantic Text Similarity) dataset as benchmark for evaluation, and provided us a file-tuned sentence embedder to derive embeddings for our clustering and search tasks on the CORD-19 dataset.

```
# select one Transformer
model_name = 'allenai/scibert_scivocab_uncased'

# Use sciBERT model for mapping tokens to embeddings
word_embedding_model = models.BERT(model_name)
```

```
# Apply mean pooling to get one fixed sized sentence vector
pooling_model = models.Pooling(word_embedding_model.get_word_embedding_dimension(),
                               pooling_mode_mean_tokens=True,
                               pooling_mode_cls_token=False,
                               pooling_mode_max_tokens=False)

model = SentenceTransformer(modules=[word_embedding_model, pooling_model])

# Convert the dataset to a DataLoader ready for training
train_data = SentencesDataset(nli_reader.get_examples('train.gz'), model=model)
train_dataloader = DataLoader(train_data, shuffle=True, batch_size=batch_size)
train_loss = losses.SoftmaxLoss(model=model, sentence_embedding_dimension=model.get_sentence_en

dev_data = SentencesDataset(examples=sts_reader.get_examples('sts-dev.csv'), model=model)
dev_dataloader = DataLoader(dev_data, shuffle=False, batch_size=batch_size)
evaluator = EmbeddingSimilarityEvaluator(dev_dataloader)

num_epochs = 2
warmup_steps = math.ceil(len(train_dataloader) * num_epochs / batch_size * 0.1) #10% of train
# Train the model
model.fit(train_objectives=[(train_dataloader, train_loss)],
          evaluator=evaluator,
          epochs=num_epochs,
          evaluation_steps=1000,
          warmup_steps=warmup_steps,
          output_path=model_save_path
          )
```

The training of the sentence transformer took eventually around 5 hours when performed on Kaggle notebook with GPU.

At this point, we can simply use our fine-tuned model (perfect for our biomedical dataset) to encode our corpus. This script provided the embeddings we need, both for the abstract and separately for the full text of each article (the latter process also took around 4-6 hours on GPU to complete). We will then access these embeddings in the dataframe for downstream tasks below.

```
## import embeddings
import pandas as pd
from sentence_transformers import SentenceTransformer
from scripts import embeddings

# download pre-trained model
model = SentenceTransformer('model1') # model we fine-tuned and saved

# import dataframe without embeedings yet
df_covid = pd.read_pickle(f'{root_path}/data/sample_preprocessed_dataframe.pkl')

# add abstract embeddings to dataframe
df_covid['abs_embeddings'] = embeddings.sent_embeddings(df_covid['abstract'], model)
```

```
# add full text embeddings to dataframe
df_covid['body_embeddings'] = embeddings.sent_embeddings(df_covid['body_text'], model)

# save dataframe
df_covid.to_pickle('./data/compelete_dataframe.pkl')
```

4 Clustering

4.1 UMAP

Uniform Manifold Approximation and Projection (UMAP)6 is an algorithm for dimension reduction based on manifold learning techniques and ideas from topological data analysis. It provides a very general framework for approaching manifold learning and dimension reduction, but can also provide specific concrete realizations and can preserve more of the global structure with superior run time performance.

[13]:
```
import umap.umap_ as umap
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

[14]:
```
#open pickle file to extract vextorization
df=pd.read_pickle('data/compelete_dataframe.pkl')
```

[15]:
```
reducer = umap.UMAP(n_neighbors = 5)
```

[16]:
```
#for abstract
numpy_array=df['abs_embeddings'][0]
for i in range(1,len(df['abs_embeddings'])):
  numpy_array=np.row_stack((numpy_array,df['abs_embeddings'][i]))
```

[17]:
```
#for body text
numpy_array2=df['body_embeddings'][0]
for i in range(1,len(df['body_embeddings'])):
  numpy_array2=np.row_stack((numpy_array2,df['body_embeddings'][i]))
```

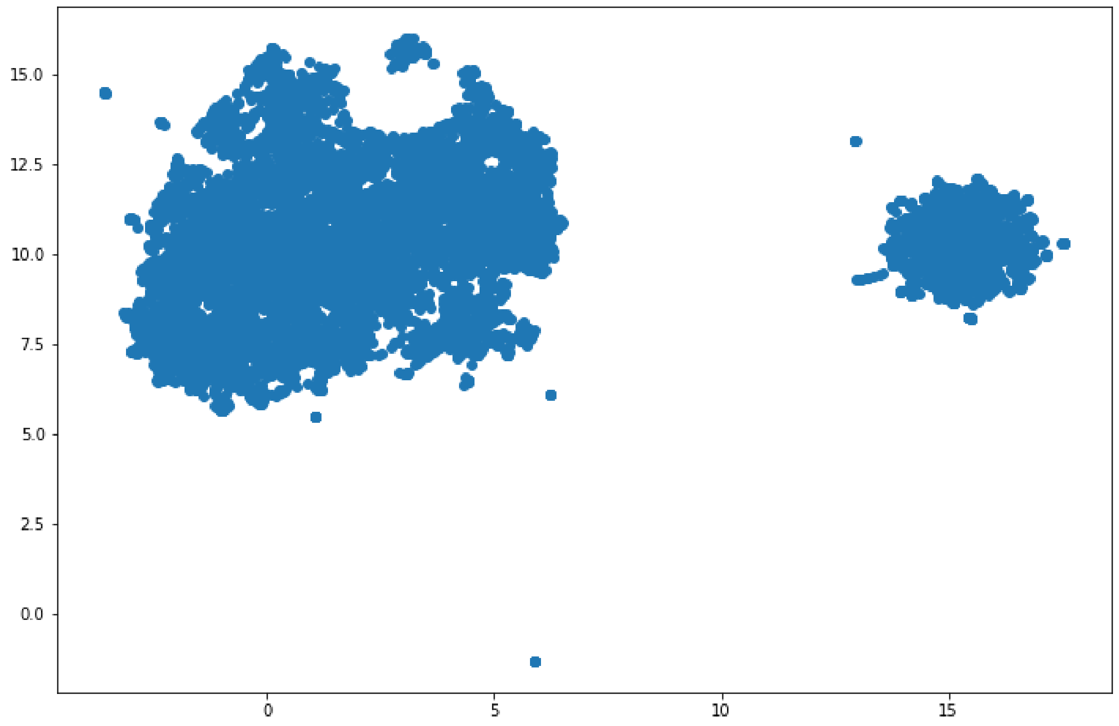
4.1.1 Visualization

Abstract

[18]:
```
# reduce to two dimentions and plot result
clusterable_embedding = reducer.fit_transform(np.asmatrix(numpy_array))
plt.figure(figsize=(12,8))
plt.scatter(clusterable_embedding[:,0],clusterable_embedding[:,1])
clusterable_embedding.shape
print(clusterable_embedding)
```

```
/usr/local/lib/python3.7/site-packages/numba/np/ufunc/parallel.py:355:
NumbaWarning: The TBB threading layer requires TBB version 2019.5 or later
i.e., TBB_INTERFACE_VERSION >= 11005. Found TBB_INTERFACE_VERSION = 11000. The
TBB threading layer is disabled.
  warnings.warn(problem)
```

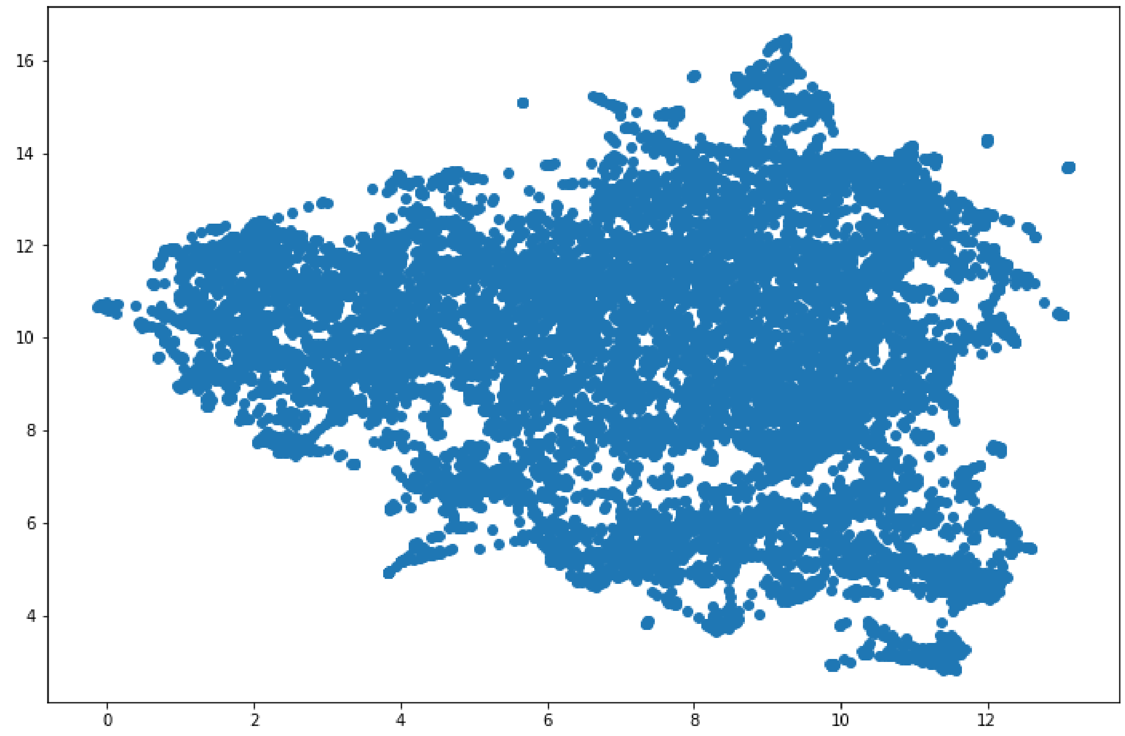
```
[[ 1.3151064 14.963993 ]
 [16.357409  10.505647 ]
 [-1.4696     7.2187424]
 …
 [ 3.8887513  7.5552874]
 [-1.7892604 10.335872 ]
 [-0.76065   12.922169 ]]
```



Body Text

```
[19]: clusterable_embedding2 = reducer.fit_transform(np.asmatrix(numpy_array2))
plt.figure(figsize=(12,8))
plt.scatter(clusterable_embedding2[:,0],clusterable_embedding2[:,1])
clusterable_embedding2.shape
print(clusterable_embedding2)
```

```
[[ 4.8522654  7.29898  ]
 [10.4329405 11.176286 ]
 [11.461031   2.9171674]
 …
 [ 6.1377482  6.2594786]
 [ 7.7757587  5.3794594]
 [ 7.9246597 11.538631 ]]
```



4.2 HDBSCAN

We are trying to run vectorization and separate the literature by two ways, the first one is HDBSCAN. HDBSCAN is a clustering algorithm developed by Campello, Moulavi, and Sander.7 It extends DBSCAN (a classic density-based spatial algorithm) by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters.

[20]:
```
import hdbscan
import numpy as np
import seaborn as sns
import pandas as pd
```

[21]:
```
#Abstract
clusterer = hdbscan.HDBSCAN(min_cluster_size=2, gen_min_span_tree=True)
clusterer=clusterer.fit(clusterable_embedding)
```

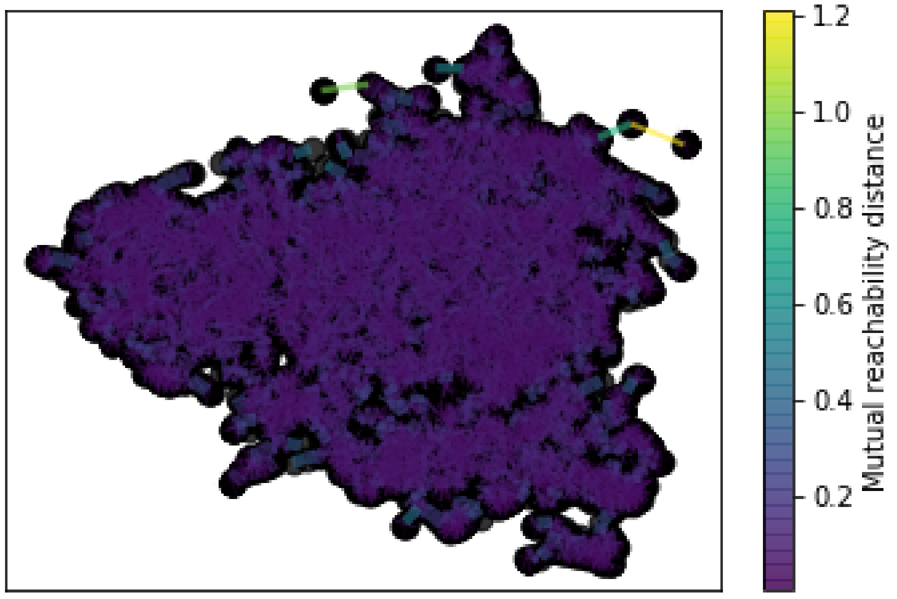
```
[22]: #Body Text
      clusterer2=clusterer.fit(clusterable_embedding2)
```

4.2.1 Visualization

Abstract

```
[23]: #Build the minimum spanning tree
      clusterer.minimum_spanning_tree_.plot(edge_cmap='viridis',
                                            edge_alpha=0.6,
                                            node_size=80,
                                            edge_linewidth=2)
```

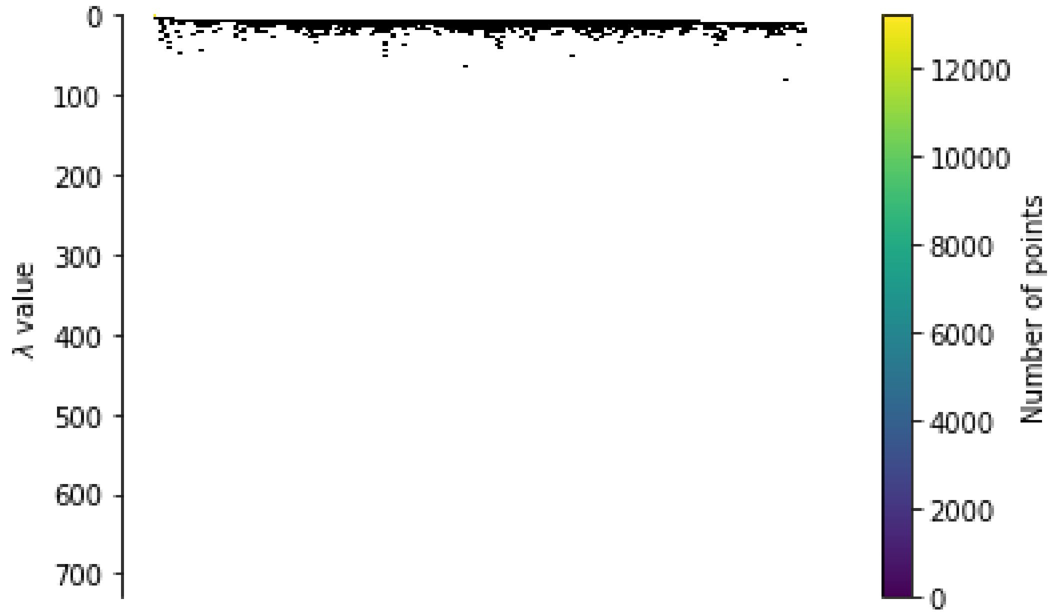
```
[23]: <matplotlib.axes._subplots.AxesSubplot at 0x187927690>
```



```
[64]: #Build the cluster hierarchy
      clusterer.single_linkage_tree_.plot(cmap='viridis', colorbar=True)
```

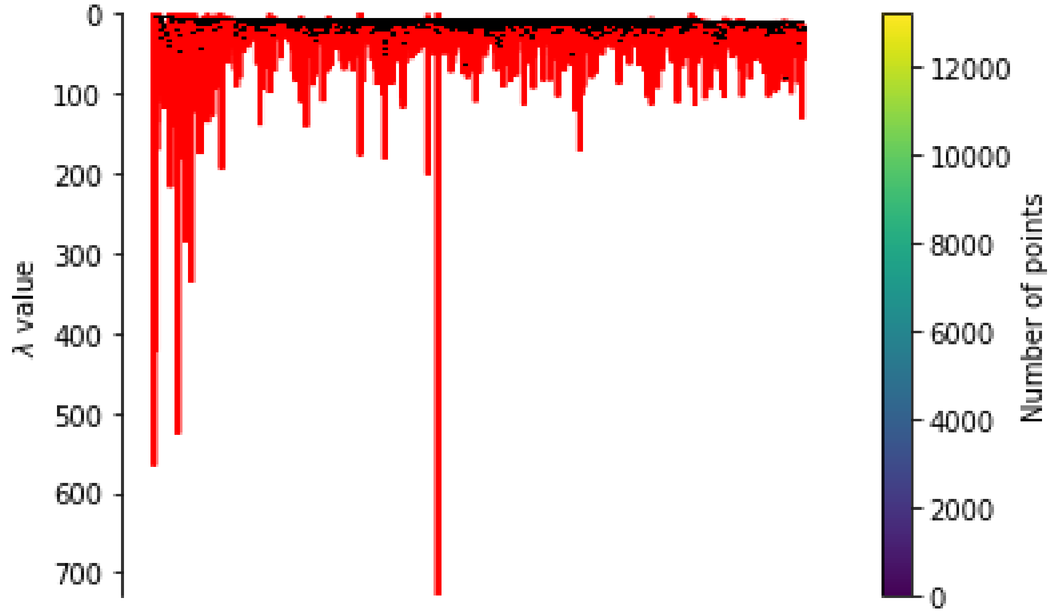
```
[25]: #Condense the cluster tree
      clusterer.condensed_tree_.plot()
```

```
[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1e1f44790>
```



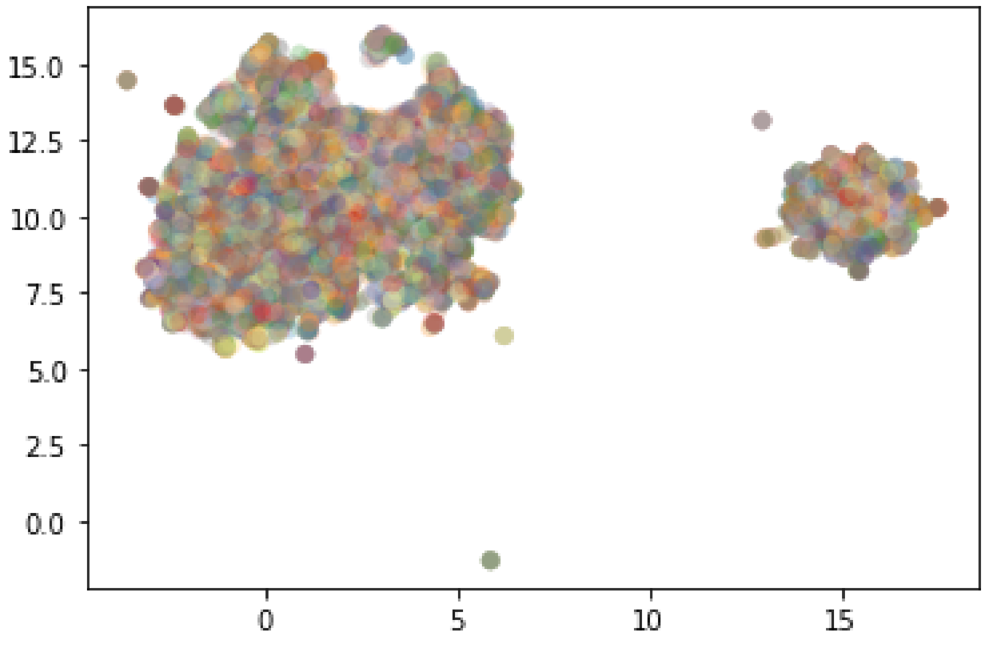
```
[26]: #Extract the clusters
      clusterer.condensed_tree_.plot(select_clusters=True, selection_palette=sns.
       ↪color_palette())
```

```
[26]: <matplotlib.axes._subplots.AxesSubplot at 0x181937990>
```



[27]:
```
#clusterer = hdbscan.HDBSCAN(min_cluster_size=10, prediction_data=True).
↪fit(clusterable_embedding)
color_palette = sns.color_palette('Paired',max(clusterer.labels_))
cluster_colors = [color_palette[x] if x >= 0 and x<max(clusterer.labels_)
                  else (0.5, 0.5, 0.5)
                  for x in clusterer.labels_]
cluster_member_colors = [sns.desaturate(x, p) for x, p in
                         zip(cluster_colors, clusterer.probabilities_)]
plt.scatter(*clusterable_embedding.T, s=50, linewidth=0,␣
↪c=cluster_member_colors, alpha=0.25)
```

[27]:
```
<matplotlib.collections.PathCollection at 0x23aa19990>
```

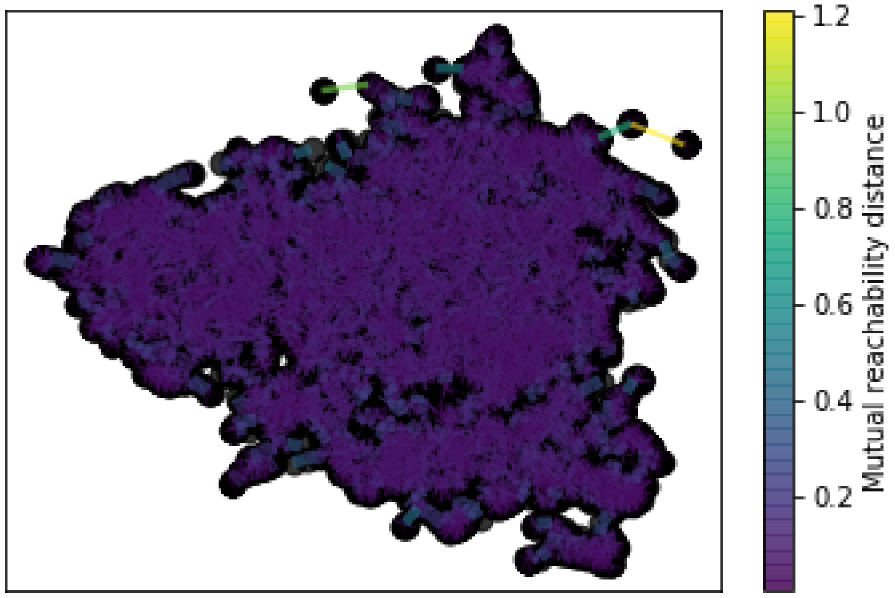


We got more than 1000 clustering groups totally, so HDBSCAN may not be the best way to cluster, and then we will try K-Means.

Body Text

[28]:
```
#Build the minimum spanning tree
clusterer2.minimum_spanning_tree_.plot(edge_cmap='viridis',
                                       edge_alpha=0.6,
                                       node_size=80,
                                       edge_linewidth=2)
```

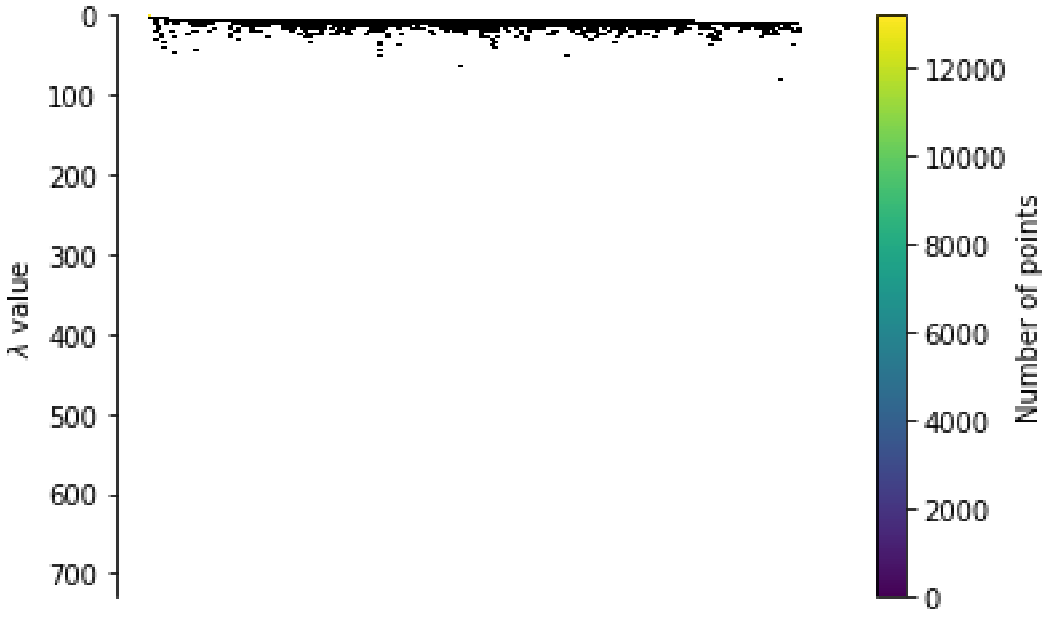
[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e2337dd0>
```



```
[ ]: #Build the cluster hierarchy
     clusterer2.single_linkage_tree_.plot(cmap='viridis', colorbar=True)
```

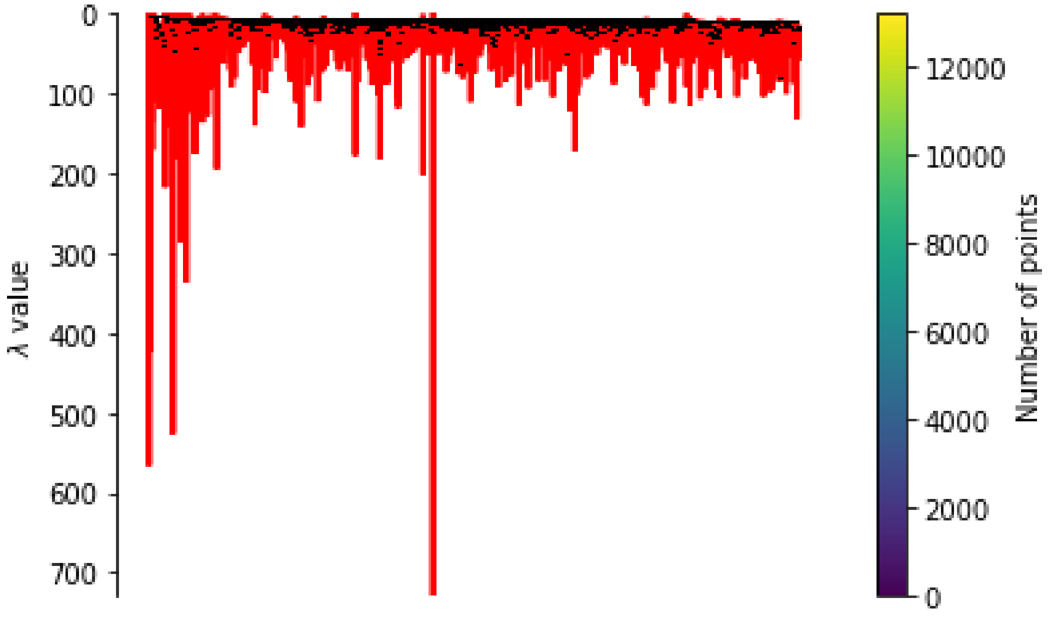
```
[29]: #Condense the cluster tree
      clusterer2.condensed_tree_.plot()
```

```
[29]: <matplotlib.axes._subplots.AxesSubplot at 0x2401f0f10>
```



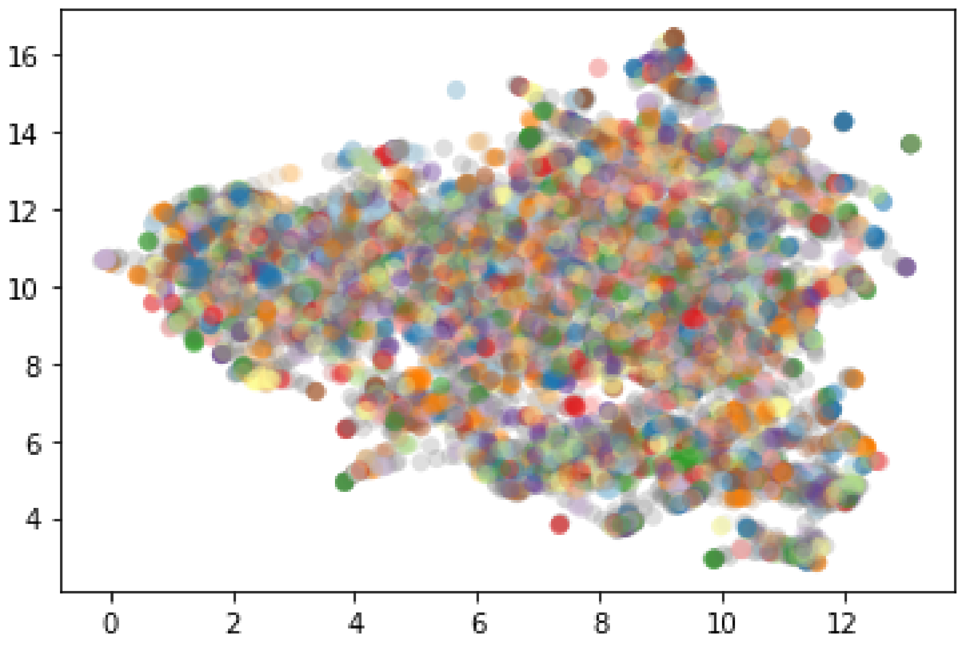
```
[30]: #Extract the clusters
      clusterer2.condensed_tree_.plot(select_clusters=True, selection_palette=sns.
       ↪color_palette())
```

```
[30]: <matplotlib.axes._subplots.AxesSubplot at 0x246403ed0>
```



```
[31]: #clusterer = hdbscan.HDBSCAN(min_cluster_size=10, prediction_data=True).
       ↪fit(clusterable_embedding)
      color_palette = sns.color_palette('Paired',max(clusterer2.labels_))
      cluster_colors = [color_palette[x] if x >= 0 and x<max(clusterer2.labels_)
                        else (0.5, 0.5, 0.5)
                        for x in clusterer.labels_]
      cluster_member_colors2 = [sns.desaturate(x, p) for x, p in
                                zip(cluster_colors, clusterer2.probabilities_)]
      plt.scatter(*clusterable_embedding2.T, s=50, linewidth=0,␣
       ↪c=cluster_member_colors2, alpha=0.25)
```

```
[31]: <matplotlib.collections.PathCollection at 0x254244b10>
```



4.3 K-means

Then we will see what k-means clustering to be like and what makes it different from HDBSCAN. First step is finding best k-value. Distortion computes the sum of squared distances from each point to its assigned center. When distortion is plotted against k there will be a k value after which decreases in distortion are minimal. This is the desired number of clusters.

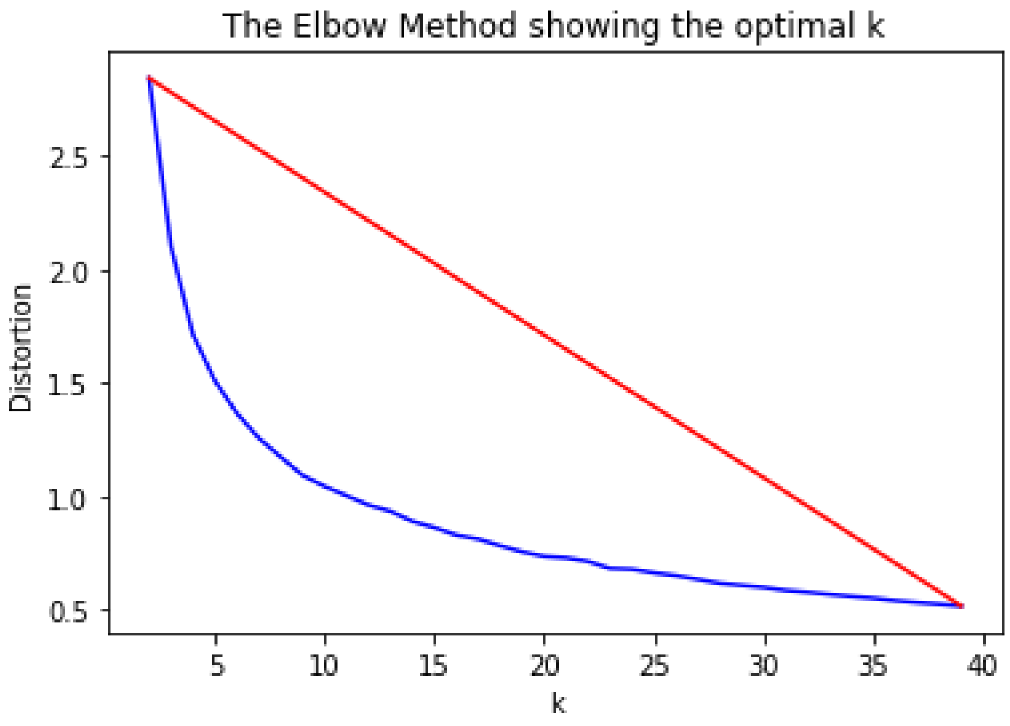
[32]:
```
from sklearn.cluster import KMeans
```

Abstract

[34]:
```
from scipy.spatial.distance import cdist
distortions = []
K = range(2, 40)
for k in K:
    k_means = KMeans(n_clusters=k, random_state=42).fit(clusterable_embedding)
    k_means.fit(clusterable_embedding)
    distortions.append(sum(np.min(cdist(clusterable_embedding, k_means.
 ↪cluster_centers_, 'euclidean'), axis=1)) / pd.DataFrame(df).shape[0])
```

[35]:
```
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()
```



The best k-value is about 9 to 12, so we determined the best one is 10

[36]:
```
kmeans = KMeans(n_clusters=10, random_state=0).
 ↪fit_predict(clusterable_embedding)
kmeans
```

[36]:
```
array([8, 1, 3, …, 9, 5, 8], dtype=int32)
```

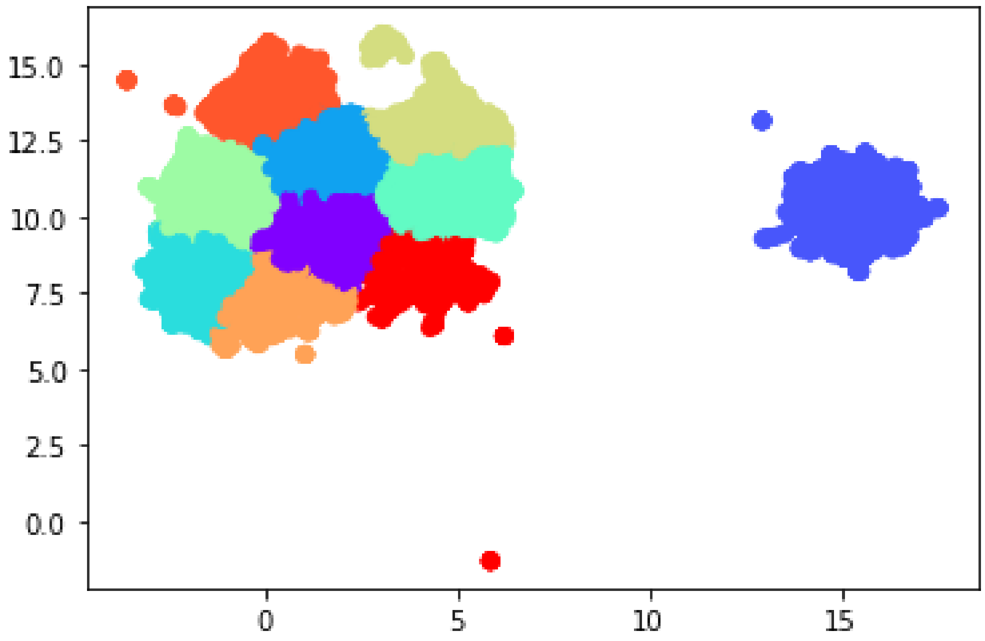
[37]:
```
kmeans_clustering_model=KMeans(n_clusters=10, random_state=0)
kmeans_clustering_model.fit(clusterable_embedding)
kmeans_label=kmeans_clustering_model.labels_
```

[38]:
```
df['cluster_']=kmeans_label
```

Now we can compare two ways of clustering. Usually K-Means works well for "round" or spherical, and when most dense in the center of the sphere not contaminated by noise/outliers. Our dataset does not centered with arbitrary shapes and too many noises, therefore K-Means works well here.

```
[39]: plt.scatter(clusterable_embedding[:,0],clusterable_embedding[:,1], c=kmeans,␣
       ↪cmap='rainbow')
```

```
[39]: <matplotlib.collections.PathCollection at 0x254578190>
```

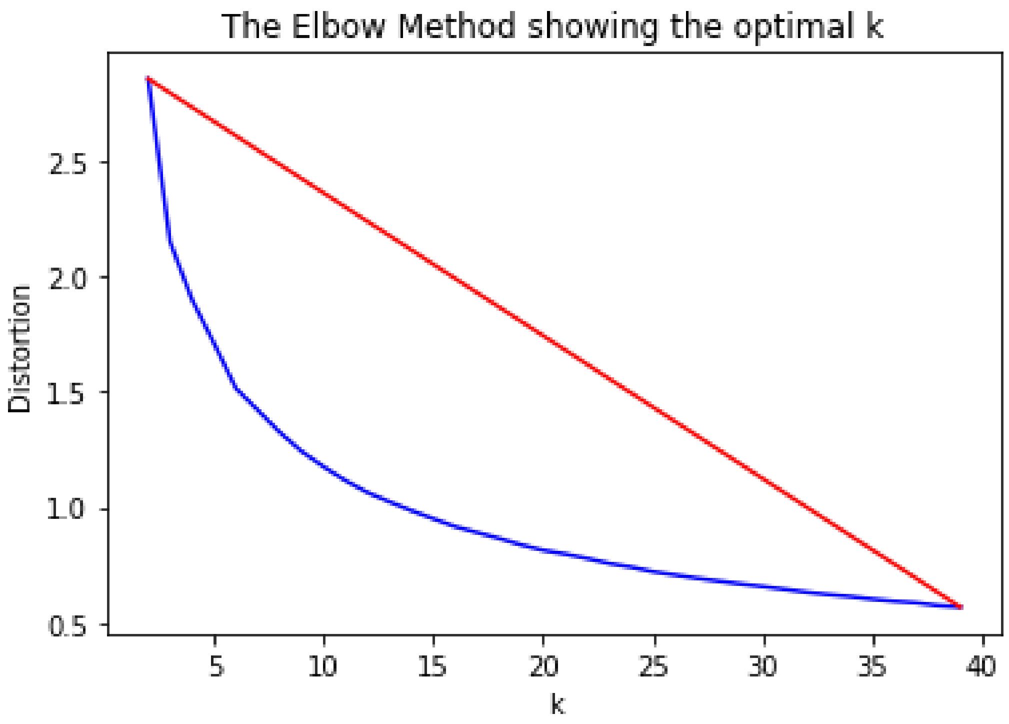


Body Text

```
[41]: from scipy.spatial.distance import cdist
      distortions2 = []
      K2 = range(2, 40)
      for k in K:
          k_means2 = KMeans(n_clusters=k, random_state=42).fit(clusterable_embedding2)
          k_means2.fit(clusterable_embedding2)
          distortions2.append(sum(np.min(cdist(clusterable_embedding2, k_means2.
       ↪cluster_centers_, 'euclidean'), axis=1)) / pd.DataFrame(df).shape[0])
```

```
[42]: X_line2 = [K2[0], K2[-1]]
      Y_line2 = [distortions2[0], distortions2[-1]]

      # Plot the elbow
      plt.plot(K2, distortions2, 'b-')
      plt.plot(X_line2, Y_line2, 'r')
      plt.xlabel('k')
      plt.ylabel('Distortion')
      plt.title('The Elbow Method showing the optimal k')
      plt.show()
```



```
[43]: kmeans2 = KMeans(n_clusters=10, random_state=0).
       ↪fit_predict(clusterable_embedding2)
      kmeans2
```

```
[43]: array([0, 8, 4, …, 0, 7, 5], dtype=int32)
```

```
[44]: kmeans_clustering_model2=KMeans(n_clusters=10, random_state=0)
      kmeans_clustering_model2.fit(clusterable_embedding2)
      kmeans_label2=kmeans_clustering_model2.labels_
```

```
[45]: df['textcluster_']=kmeans_label2
```

4.4 Topic Modeling

After finding the best clustering model, we would like to try to label each topic. Most frequent methods of topic modeling, including Latent Dirichlet Allocation.9 , a generative statistical model that uses unobserved groups to explain why some parts of a corpus are similar (and non-negative matrix factorization!), do not work with the embeddings method we chose as such models perform **supervised learning** on labeled data (while our embeddings were on the contrary created specially for unsupervised tasks).
Indeed, LDA model posits that each document is a mixture of a small number of topics, where each topic is described by a distribution of words and each word's presence in the document can be attributed to one of these topics. If someone would like to explore these model, they would

re-preprocess text using TFIDFVectorizer or CountVectorizer to then find the topics from each model.

5 Semantic search

5.0.1 Cosine similarity

An amazing use of the embeddings we obtained with our fine-tuned Sentence Transformer model is to query the corpus and find the most similar embeddings to the query's embeddings. This can be simply done by calculating cosine similarity among embeddings, and then select the least-distant ones as the most relevant to the query.
Here we propose a semantic search system where the user can input a question about COVID-19 and they will get the top 5 most relevant articles from the corpus. We suggest to interrogate our system with questions taken from the Kaggle's suggested tasks.

[4]:
```
## call searches and get table out
from scripts import searches
import pandas as pd
from sentence_transformers import SentenceTransformer
import os

# import model
embedder = SentenceTransformer('model1') #this or the model we trained and saved

# load corpus
df_covid = pd.read_pickle(os.getcwd()+'/data/compelete_dataframe.pkl')
```

[6]:
```
# asking the user
query = input('What would you like to know from CORD-19? ')
print('\nUse abstracts:')
new_searches.sem_search(query, embedder, df_covid['abstract'].to_list(),␣
 ↪df_covid['abs_embeddings'].to_list(),df_covid)
```

```
What would you like to know from CORD-19?  coronavirus


Use abstracts:



======================




Top 5 most similar articles in corpus:
+-----------------------------------------------------------------+-------
-----------------------------------------------------------+--------------
-----+--------+
|                                Abstract                                 |
Title                                      |      Journal        | Score  |
+-----------------------------------------------------------------------+-------
-----------------------------------------------------------------+--------------
-----+--------+
| since the severe acute respiratory syndrome coronavirus 2  sars cov 2 |
nan                                      |        nan        | 0.7367 |
|                                 disease                               |
|                   |        |
|                                                                       |
|                   |        |
|                                                                       |
|                   |        |
|     1 the outbreak of novel coronavirus pneumonia  ncp  caused by     |
nan                                      |        nan        | 0.7062 |
|                                                                       |
|                   |        |
|                                                                       |
|                   |        |
|        the emergence of severe acute respiratory syndrome  sars       |  A
ten-year China-US laboratory collaboration: improving response to   | BMC Public
Health | 0.667  |
|                                                                       |
influenza threats in China and the world, 2004-2014           |
|        |
|                                                                       |
|                   |        |
|                                                                       |
|                   |        |
|                        a number of virologic and                      |
Mechanisms of viral emergence                      |      Vet Res      | 0.6585
|
|                                                                       |
|                   |        |
|                                                                       |
|                   |        |
|       endemic and seasonally recurring respiratory viruses are a      |
Surveillance of respiratory viruses in the outpatient setting in rural |
Wellcome Open Res | 0.6473 |
|                                                                       |
coastal Kenya: baseline epidemiological observations          |
|        |
|                                                                       |
|                   |        |
|                                                                       |
|                   |        |
+-----------------------------------------------------------------------+-------
-----------------------------------------------------------------+--------------
-----+--------+
======================
```

```
[7]: # asking the user (slower)
     print('\nUse full text:')
     new_searches.sem_search(query, embedder, df_covid['body_text'].to_list(),␣
      ↪df_covid['body_embeddings'].to_list(),df_covid)
```

```
Use full text:


======================



Top 5 most similar articles in corpus:
+-----------------------------------------------------------------------------+--
-----------------------------------------------------------------------------+----
------------------+--------+
|                                  Abstract                                   |
Title                                         |       Journal        | Score  |
+-----------------------------------------------------------------------------+--
-----------------------------------------------------------------------------+----
------------------+--------+
|                                                                             |
Niclosamide Is a Proton Carrier and Targets Acidic Endosomes with Broad   |
PLoS Pathog      | 0.6462 |
|                                                                             |
Antiviral Effects                                |                       |        |
|                                                                             |
|                      |        |
|                                                                             |
|                      |        |
|                                                                             |
Comparison of fit factors among healthcare providers working in the     |
Medicine (Baltimore) | 0.5974 |
|                                                                             |
Emergency Department Center before and after training with three types of   |
|        |
|                                                                             |
N95 and higher filter respirators                      |                       |
|
|                                                                             |
|                      |        |
|                                                                             |
|                      |        |
|   feline infectious peritonitis  fip  is a severe fatal immune augmented   |
In Vitro Antiviral Activity of Circular Triple Helix Forming        |    Biomed
Res Int    | 0.5784 |
|  disease in cat population. it is caused by fip virus  fipv , a virulent   |
Oligonucleotide RNA towards Feline Infectious Peritonitis Virus Replication |
|        |
| mutant strain of feline enteric coronavirus  fecv . current treatments and |
|                      |        |
| prophylactics are not effective. the in vitro antiviral properties of five |
|                      |        |
|  circular triple helix forming oligonucleotide  tfo  rnas  tfo1 to tfo5 ,  |
|                      |        |
|  which target the different regions of virulent feline coronavirus  fcov   |
|                      |        |
| strain fipv wsu 79 1146 genome, were tested in fipv infected crandell rees |
|                      |        |
|  feline kidney  crfk  cells. rt qpcr results showed that the circular tfo  |
|                      |        |
|  rnas, except tfo2, inhibit fipv replication, where the viral genome copy  |
|                      |        |
|  numbers decreased significantly by 5 fold log 10 from 10 14 in the virus  |
|                      |        |
|    inoculated cells to 10 9 in the circular tfo rnastransfected cells.     |
|                      |        |
|  furthermore, the binding of the circular tfo rna with the targeted viral  |
|                      |        |
|   genome segment was also confirmed using electrophoretic mobility shift   |
|                      |        |
|   assay. the strength of binding kinetics between the tfo rnas and their   |
|                      |        |
|    target regions was demonstrated by nanoitc assay. in conclusion, the    |
|                      |        |
|   circular tfos have the potential to be further developed as antiviral    |
|                      |        |
|                       agents against fipv infection.                       |
|                      |        |
|                                                                            |
|                      |        |
|                                                                            |
|                      |        |
|                                                                            |
"The Duty to Prevent" during an epidemic situation like 2015 Korean MERS  |
Epidemiol Health   | 0.5156 |
|                                                                            |
outbreak                                   |                      |        |
|                                                                            |
|                      |        |
|                                                                        |
|                     |       |
|                                                                        |
A benchmark driven guide to binding site comparison: An exhaustive     |   PLoS
Comput Biol   | 0.5155 |
|                                                                        |
evaluation using tailor-made data sets (ProSPECCTs)            |
|       |
|                                                                        |
|                     |       |
|                                                                        |
|                     |       |
+------------------------------------------------------------------------+--
----------------------------------------------------------------------+----
-----------------+--------+


====================
```

6 Conclusion

This project taught us how complex models that are truly applicable and easily transferable can (fairly) quickly yield amazing results on massive and diverse amount of data, in this case natural language data. Transformers model (which we were even able to fine-tune for our domain-specific project) have been perfected to an extent that we can perform both supervised and unsupervised tasks in a timely manner, given the widely available hardware accelerators in the community.
For the CORD-19 challenge, we decided to fine-tuned Sentence-BERT on SciBERT, so that we had our own Sentence-SciBERT to obtain embeddings from our data to use for unsupervised downstream tasks. Thus, we used these embeddings to cluster the articles. HDBSCAN proved quite innefficient, as given the high condensity of the dataset the model did not yield very interpretable results. On the other hand, a classic K-means model performed well on such dense and "spherical" dataset, by learning 10 clusters of articles. Unfortunately, our embeddings could not be used to try topic modeling algorithms, such as LDA.
Finally, we had fun experimenting with a simple algorithm based on cosine similarity to search through the dataset. By using embeddings of articles and of the query, this straightforward application can output the most relevant articles for the user searches in a pretty table. Go ahead and download this notebook from our GitHub Repo to play around with it!
We hope we were able to prove the potentiality and power of these machine learning models and approaches, and we wish that the research community will continue to have such a fundamental impact on current and urgent problems!