CORD-19 Topic Modelling using BERTopic - ✅ Complete

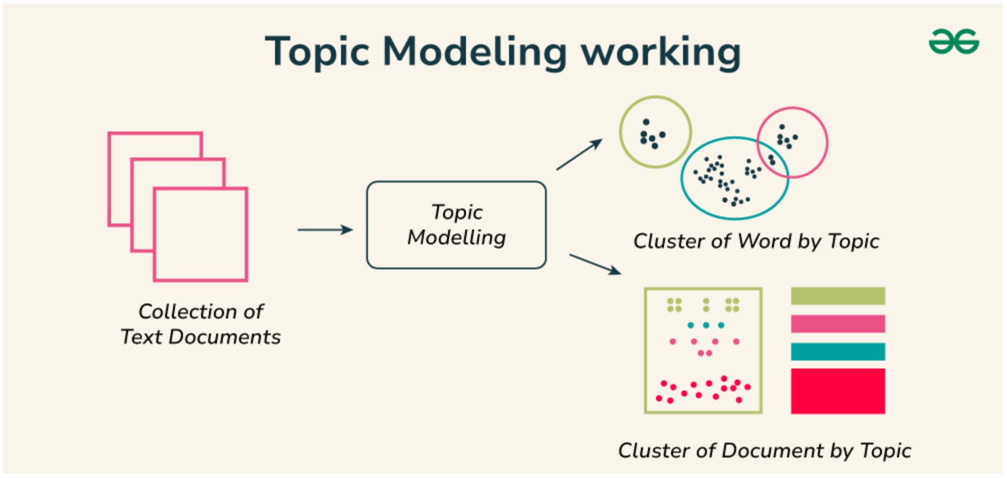


Image Source: GeeksforGeeks

Topic modeling is a technique in Natural Language Processing (NLP) and machine learning that is used to automatically discover the underlying topics in a collection of text documents. It helps uncover the hidden structure within the text by identifying groups of words that frequently occur together, which can be interpreted as topics.

For this experiment, we'll be using the BERTopic model.

The BERTopic is a topic modelling technique that leverages Google's BERT model. It is an advanced neural model that uses BERT(Bidirectional Encoder Representations from Transfromers) and c-TF-IDF to generate dense clusters that enable for easily interpretable subjects while retaining key terms in the topic descriptions.

In this notebook, we'll be performing expriments in steps.

1. Data Loading
2. Data Cleaning
3. Exploratory Data Analysis
4. Topic Modelling with BERTopic
5. Results - Distribution, Visualisation, Evaluation scores

📌 NOTE

- This notebook is categrosied into sections.
- To run this experiment:
    1. **Run the first cell** to install `BERTopic`.
    2. **Run the second cell** to import all necessary libraries.

3. Then proceed to the **`Experiment after cleaning`** section.

```
!pip install bertopic
```

```
import numpy as np # for mathematical operations
import pandas as pd # for data manipulation
import matplotlib.pyplot as plt # for visualisation
import seaborn as sns # For visualisation
from wordcloud import WordCloud, STOPWORDS #
from bertopic import BERTopic # For topic modelling
from tqdm import tqdm # for progress bar
import spacy #
import re
import gdown # for downloading from google drivve
from IPython.display import clear_output # Clearing output in Jupyter notebook cell
```

```
c:\Users\david\AppData\Local\Programs\Python\Python312\Lib\site-packages\tqdm\auto.py:21
  from .autonotebook import tqdm as notebook_tqdm
WARNING:tensorflow:From c:\Users\david\AppData\Local\Programs\Python\Python312\Lib\site-
```

Step 1 - Data Loading

Load dataset.

```
# # Download data from googe drive
# file_id = "1bF7HBuC6f7rC8FmJBwzb-vQwi6m81E68"

# # download
# gdown.download(f"https://drive.google.com/uc?id={file_id}", "metadata.csv", quiet=False)
```

```
# data = pd.read_csv('metadata.csv')
```

```
# display(data.head(10))
```

As you can see, there are numerous columns. Most columns are for reference purposes or typical metadata for archival and storage purposes. We highlight the relevant columns which will be used for visualisation, and for the topic modelling task.

```
# data.drop(
#     columns = ['doi','mag_id','arxiv_id', 'pdf_json_files', 'pmc_json_files',
#                'mag_id', 'who_covidence_id','doi', 'cord_uid', 'sha','s2_id','pubmed_id','
# data.dropna().reset_index(drop=True)

# display(data.head(10))
```

Now we've loaded data with our needed columns, we proceed to data cleaning. Before cleaning our dataset, there is about 1,056,660 papers

Step 2 - Data Cleaning - Skip

This involves identifying and removing duplicates, missing or irrelevant data.

After several cleaning attempts, we produced a one general cleaned dataset which we will use for the experiment. So this initial step is therefore redudant. So skip

```
# # remove empty abstracts or abstracts with 'NaN'
# data = data[data['abstract'] != '' ]
# data = data[data['abstract'].notna()]


# # remove abstract with incorrect description
# data =  data[data['abstract'] != '[Figure: see text].']
# data = data.reset_index(drop=True)


# def clean_abstract(text):
#     return re.sub(r'^(OBJECTIVE:|BACKGROUND)\s*', '', text)

# # clean abstract
# data['abstract'] = data['abstract'].apply(clean_abstract)


####################################################
# CUTTING DATASET SIZE DUE TO COMPUTE
####################################################
# data = data[:500]
```

Noticed some abstracts aren't well written. They included "BACKGROUND:" or "INTRODUCTION:" at the start of their abstract. This might affect the model's prediction.

```
# # remove stopwords
# nlp = spacy.load('en_core_web_sm')

# def remove_stopwords(text):
#     doc = nlp(text)
#     # Filter out stop words and punctuation
#     filtered_words = [token.text for token in doc if not token.is_stop and not token.is_pu
#     return ' '.join(filtered_words)


# data_50_sets = data[:50]
# data_10k_sets = data[:10000]
# data_100k_sets = data[:100000]
# data_all_sets = data[:800000]

# # Apply the function to the 'abstract' column
# tqdm.pandas()
# data_50_sets['abstract'] = data_50_sets['abstract'].progress_apply(remove_stopwords)
# data_10k_sets['abstract'] = data_10k_sets['abstract'].progress_apply(remove_stopwords)
# data_100k_sets['abstract'] = data_100k_sets['abstract'].progress_apply(remove_stopwords)
# data_all_sets['abstract'] = data_all_sets['abstract'].progress_apply(remove_stopwords)


# all cleaned sets
# data_50_sets.tocsv('cleaned_data50.csv', index=False)
# data_10k_sets.tocsv('cleaned_data10k.csv', index=False)
# data_100k_sets.tocsv('cleaned_data100k.csv', index=False)
# data_all_sets.tocsv('cleaned_data_all.csv', index=False)

# # Download the CSV file
# import google.colab.files as files
# files.download('cleaned_data50.csv')
# files.download('cleaned_data10k.csv')
# files.download('cleaned_data100k.csv')
# files.download('cleaned_data_all.csv')
```

For a task like topic modelling, removing stopwords is crucial as it helps reduce noise and prevents overfitting.

Now this process takes loads of time becasue of the size of our data. For this reason, I have perfomed this experiment and saved its clean version.

```
# # download cleaned sets
# file_id50 = "1kjwcxTJB" # 50 samples
# file_id10k = "1kjwcxTp" # 10k samples
# file_id100k = "1kjwcxT" # 100k samples
# file_id_all = "1kjwcxTJBNgdp" # all samples

# gdown.download(f"https://drive.google.com/uc?id={file_id50}", "cleaned50.csv", quiet=False
# gdown.download(f"https://drive.google.com/uc?id={file_id10k}", "cleaned10k.csv", quiet=Fal
# gdown.download(f"https://drive.google.com/uc?id={file_id100k}", "cleaned100k.csv", quiet=F
# gdown.download(f"https://drive.google.com/uc?id={file_id_all}", "cleaned_all.csv", quiet=F
# # For all samples
# data = pd.read_csv('cleaned_all.csv')

# # For 50 samples
# # data = pd.read_csv('cleaned50.csv')

# # For 10k samples
# # data = pd.read_csv('cleaned10k.csv')

# # For 100k samples
# # data = pd.read_csv('cleaned100k.csv')
```

```
Start coding or generate with AI.
```

Step 3 - Exploratory Data Analysis (EDA)

```
# # Number of papers
# print(f"Total Length of dataset: {len(data)}")
```

About 820,799 sets after cleaning.

```
# data.describe()

# # print(f"Total Length of dataset: {len(data)}")
```

3.1.1 - Sources

```
# sources = data['source_x'].str.split(';').explode().str.strip()

# unique_sources = sorted(sources.unique())

# print(f"Number of unique sources: {len(unique_sources)} \n")

# for i in range(len(unique_sources)):
#     print(f"{i+1}. {unique_sources[i]}")
```

There are about 7 unique sources, as mentioned in the CORD-19 research paper. These are all relevant sources for any research in the medical, or healthcare field.

3.1.2 - Journals

```
# # uniquue journals
# unique_journal = data['journal'].unique()
# print(f"Number of unique journals: {len(unique_journal)} \n")


# # top journals
# top_journals = data['journal'].value_counts().head(10)
# print(top_journals)

# # chart for top journals
# plt.figure(figsize=(10, 6))
# sns.barplot(x = top_journals.index, y = top_journals.values)
# plt.xlabel('Journal')
# plt.ylabel('Count')
# plt.title('Top 10 Journals')
# plt.xticks(rotation=90)
# plt.show()
```

There are about 54,994 unique journals are here are the top 10.

PloS One - is a popular open-source .... bioRxiv - is ....

3.1.3 - Licenses

```
# # unquie license
# unique_license = data['license'].unique()
# print(f"Number of unique license: {len(unique_license)} \n")


# for i in range(len(unique_license)):
#     print(f"{i+1}. {unique_license[i]}")
```

Unique licnes

```
# # top licenses
# top_licenses = data['license'].value_counts().head(10)
# print(top_licenses)

# # plot chart for top licenses
# plt.figure(figsize=(10, 6))
# sns.barplot(x=top_licenses.index, y=top_licenses.values)
# plt.ylabel('Count')
# plt.title('Top 10 Licenses')
# plt.xticks(rotation=90)
# plt.show()
```

For comparison, here are some common license types that are typically used in publishing papers:

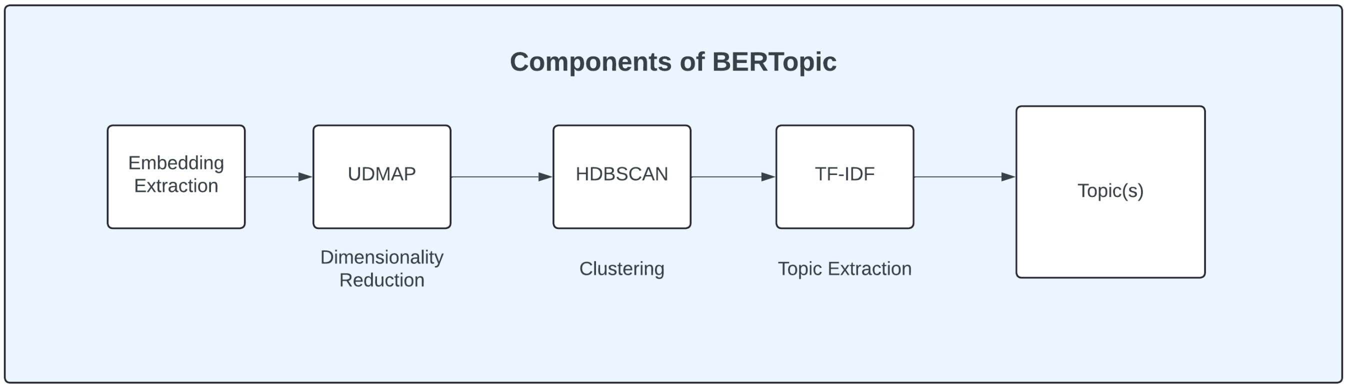
Open Access:

- CC BY: Creative Commons Attribution (allows distribution and modification with attribution).
- CC BY-SA: Creative Commons Attribution-ShareAlike (requires sharing under the same license).
- CC BY-ND: Creative Commons Attribution-NoDerivatives (allows redistribution but no modification).
- CC BY-NC: Creative Commons Attribution-NonCommercial (allows non-commercial use only).

```
# # show a column with a license name
# data[data['license'] == 'unk']
```

Step 3 - Using BERTopic

The BERTopic model (Ref) is a model that ....



- **Extraction:** First we extract
- **Dimensionality reduction**: We .....
- **Clustering**
- **Topic Extraction**

```
# !pip install bertopic
# clear_output()

# from bertopic import BERTopic # For topic modelling
```

```
# Dataset sets
# abstracts_trial = data['abstract'][:50]
# abstracts_set1 = data['abstract'][:100000]
# abstracts_set2 = data['abstract']
# abstracts = data['abstract']
```

We will be running experiments with 3 samples of the dataset due to compute constraints.

- Trial set: This is a subset sample of which we intend to use for practice and to understand how BERTopic works. We use this set for trials and understanfing the model pipeline and its variations.
- Set 1 : This will be our primary set for our experiment which contains about 100,000 data samples. This is well enough for our expeirment and it meets the required assessement.
- Set 2 : The outcome of the previous sets will determine if we will neeed to use this. This contains all the 840, ... data samples. Due to our budget on compute resources, we might not be able to use this set.

Trial run

```
# # get first 50 abstracts
# abstracts = data['abstract'].head(50)

# display(abstracts)
```

```
# # using first 50 abstracts
# abstract_set = abstracts[:50]

# # convert to list
# abstract_list = abstract_set.tolist()
```

```
# len(abstract_list)
```

```
# abstract_list = abstracts.tolist()
```

```
# topic_model = BERTopic()
# topics, probs = topic_model.fit_transform(abstract_list)
```

```
#  # view topics
#  topic_model.get_topic_info()
```

```
# # generate topic labels
# topic_labels = topic_model.generate_topic_labels( nr_words= 4 , topic_prefix= False , worc

# topic_model.set_topic_labels(topic_labels)
# print(topic_labels)


# topic_model.get_topic_info()


# # visualise - Barchart
# topic_model.visualize_barchart( width = 280, height = 330, top_n_topics= 8 , n_words = 10)


# # visualise - Heat map - Topics that are related?
# topic_model.visualize_heatmap( n_clusters = 20)


# # visualise - Document and topics
# topic_model.visualize_documents(abstract_list, topics = list(range(15)) , custom_labels =
```

2.0 - EXPERIMENT AFTER CLEANING OF DATA

After performing data cleaning, we download the cleaned datasets from Google Drive using the `gdown` library.

The `gdown` library is useful for downloading large files directly from Google Drive links in Python environments.

2.1 - Data Loading

```
# DOWNLOAD DATASETS
# file IDs
# whole_data = "14grxbT_XyV61sBMLuXJZJZhr0LeqZBfW"
# abstracts = "1-FkOy5FAqIn5ca0v3qN1JVGlwV6IxCK1"
# titles = '17t3ZPW5MwDdoWxxC8k_KypU35683uMaS'
# title_abstracts = "1-OVe4xaHzYRJvz8fupqoovQoAHl9TcN8"



# # download
# gdown.download(f"https://drive.google.com/uc?id={whole_data}", "cleaned_data.csv", quiet=F
# gdown.download(f"https://drive.google.com/uc?id={abstracts}", "cleanedabstracts.csv", quie
# gdown.download(f"https://drive.google.com/uc?id={titles}", "cleanedtitles.csv", quiet=Fals
# gdown.download(f"https://drive.google.com/uc?id={title_abstracts}", "cleanedtitlesabstract


# RUN THIS IF YOU'VE NOT DOWNLOADED
final_cleaning_set = "1NLIsEi7AFtlPAgPl5AsukjXBmVqaEwS3"
gdown.download(f"https://drive.google.com/uc?id={final_cleaning_set}", "cleaned_data.csv", c
# data = pd.read_csv('cleaned_data.csv')
# abstracts = pd.read_csv('cleanedabstracts.csv')
# titles = pd.read_csv('cleanedtitles.csv')
# title_abstracts = pd.read_csv('cleanedtitlesabstracts.csv')

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

```
data.head(5)
```

```
abstracts = pd.read_csv("abstracts.csv")
titles = pd.read_csv("titles.csv")
title_abstracts = pd.read_csv("title_and_abstracts.csv")
```

```
# abstracts["abstract"][5]
abstracts.head(5)
```

	processed_abstract
0	country affect number people infect dead first...
1	travel course build hempel legacy environmenta...
2	past research study social determinant attitud...
4	simple summary review clarify relationship hea...

```
len(abstracts)
```

```
100000
```

```
abstracts['processed_abstract'] = abstracts['processed_abstract'].fillna('')
titles['processed_title'] = titles['processed_title'].fillna('')
title_abstracts['processed_title_and_abstract'] = title_abstracts['processed_title_and_abstr
```

```
# #DEBUGGING CODE
# # check empty strings
# print(titles['processed_title'].isna().sum())

# # get indices
# nan_indices = titles[titles['processed_title'].isna()].index
# print(nan_indices)
# # dEBUGGINGG CODE
# # # show sample
# print(f"Value at index 602: {titles.loc[602, 'processed_title']}")
# print(f"Value at index 603: {titles.loc[603, 'processed_title']}")
# print(f"Value at index 661: {titles.loc[661, 'processed_title']}")
```

2.2 - Data Restructuring - Skip

From viewing the data above, we notice `"['nidovirus', 'subgenomic', 'mrnas',]"` isn't formatted properly. Instead of being Python lists, they are **strings representing tokenized abstracts**. This likely occurred during the cleaning process.

```
# import ast

# def clean_abstract(abstract):
#   words_list = ast.literal_eval(abstract)
#   words_list = [word.strip() for word in words_list if word.strip()]
#   cleaned_abstract = " ".join(words_list)
#   return cleaned_abstract


# abstract
abstracts["abstract"] = abstracts["abstract"].apply(clean_abstract)

# title
titles["title"] = titles["title"].apply(clean_abstract)

# title abstract
title_abstracts["cleaned_text"] = title_abstracts["cleaned_text"].apply(clean_abstract)
```

2.3 - Lemmatization - Skip

After initial trials, we observed that although BERTopic successfully derived meaningful topics, there were some inconsistencies in token grouping. For example, we encountered topics like:

- `cna`, `cnas`
- `student`, `students`

To address this, we apply lemmatization, specifically targeting nouns. By reducing words to their base form (e.g., `students → student`), we aim to improve topic quality and reduce redundancy.

```
# # Load the English model
# nlp = spacy.load("en_core_web_sm")

# allowed_postags = ["NOUN", "ADJ", "VERB", "ADV"]
# def lemmatize_text(text):
#     doc = nlp(text)
#     lemmatized = [token.lemma_ for token in doc if token.pos_ in allowed_postags]
#     return " ".join(lemmatized)


# # abstract
# abstracts["abstract"] = abstracts["abstract"].apply(lemmatize_text)

# # title
# titles["title"] = titles["title"].apply(lemmatize_text)

# # title abstract
# title_abstracts["cleaned_text"] = title_abstracts["cleaned_text"].apply(lemmatize_text)
```

2.4 - Data Subsampling - Skip

Due to the large size of the dataset, running the full experiment would be computationally expensive and time-consuming.

To make the process more efficient, we **subsample** the data by selecting only the **first 100,000 samples**. This allows us to run experiments faster while still working with a sufficiently large and diverse subset of the data.

```
# RUN THIS CODE IF WORKING ON FIRST 100K SAMPLES

# data = data[:100_000]
# abstracts = abstracts[:100_000]
# titles = titles[:100_000]
# title_abstracts =  title_abstracts[:100_000]


# len(data)
```

Continue

Now we are done. we can see how it looks and proceed to training our model

```
abstracts.head(5)
```

	processed_abstract
0	country affect number people infect dead first...
1	travel course build hempel legacy environmenta...
2	past research study social determinant attitud...
3	american society ae official statement provide...
4	simple summary review clarify relationship hea...

```
# # ABSTRACT
# cleaned_abstract_lists = abstracts['abstract'].tolist()
# cleaned_abstract_lists = list(tqdm(cleaned_abstract_lists, desc="Processing Abstracts"))

# # TITLES
# cleaned_title_lists = titles['title'].tolist()
# cleaned_title_lists = list(tqdm(cleaned_title_lists, desc="Processing Titles"))

# # TITLES ABSTRACT
# cleaned_titleab_lists = title_abstracts['cleaned_text'].tolist()
# cleaned_titleab_lists = list(tqdm(cleaned_titleab_lists, desc="Processing..."))

# ABSTRACT
cleaned_abstract_lists = abstracts["processed_abstract"].tolist()
cleaned_abstract_lists = list(tqdm(cleaned_abstract_lists, desc="Processing Abstracts"))

# TITLES
cleaned_title_lists = titles["processed_title"].tolist()
cleaned_title_lists = list(tqdm(cleaned_title_lists, desc="Processing Titles"))

# TITLES ABSTRACT
cleaned_titleab_lists = title_abstracts["processed_title_and_abstract"].tolist()
cleaned_titleab_lists = list(tqdm(cleaned_titleab_lists, desc="Processing..."))
```

```
Processing Abstracts: 100%|██████████| 100000/100000 [00:00<00:00, 2988651.93it/s]
Processing Titles: 100%|██████████| 100000/100000 [00:00<00:00, 4378827.80it/s]
Processing...: 100%|██████████| 99995/99995 [00:00<00:00, 2965092.92it/s]
```

```
display(cleaned_abstract_lists[0])
```

```
'country affect number people infect dead first wave describe rapid rollout population
health clinical organizational response preparedness capability support first wave
pandemic italian province review process challenge face describe excess demand hospital
service successfully mitigate thus overwhelm healthcare service avoid collapse local
health care system analysis bed occupancy region predict first week epidemic seir model
estimate number infected people different containment measure community resource
mobilize reduce provincial hospital burden care population health approach base radical
reorganization workflow emergency patient management implement bed saturation measure
newly implement intensive care unit bed occupancy centralized governance dashboard bed
```

We passed the `cleaned_abstract_lists`, which is a list of abstracts, to BERTopic for modelling.

2.5 - Model Training - BERTopic

Before training, we would like to use GPU for traing and also record our hardware specifications For this case we do this

```
# Use GPU
import torch

# Check if GPU is available
if torch.cuda.is_available():
    print("GPU is enabled:", torch.cuda.get_device_name(0))
else:
    print("No GPU detected, using CPU.")
```

```
GPU is enabled: NVIDIA GeForce RTX 3050 Ti Laptop GPU
```

```
!nvidia-smi
```

```
Fri Apr 11 13:46:42 2025
+-----------------------------------------------------------------------------------------
| NVIDIA-SMI 572.83                 Driver Version: 572.83         CUDA Version: 12.8
|-----------------------------------------+------------------------+---------------------
| GPU  Name                  Driver-Model | Bus-Id          Disp.A | Volatile Uncorr. EC
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M
|                                         |                        |               MIG M
|=========================================+========================+=====================
|   0  NVIDIA GeForce RTX 3050 ...  WDDM  |   00000000:01:00.0 Off |                  N/
| N/A   54C    P0             14W /   75W |     716MiB /   4096MiB |     19%      Defaul
|                                         |                        |                  N/
+-----------------------------------------+------------------------+---------------------

+-----------------------------------------------------------------------------------------
| Processes:
|  GPU   GI   CI              PID   Type   Process name                        GPU Memor
|        ID   ID                                                             Usage
|=========================================================================================
|    0   N/A  N/A            1716    C+G   ..._cw5n1h2txyewy\SearchHost.exe      N/A
|    0   N/A  N/A            5456    C+G   ...indows\System32\ShellHost.exe      N/A
|    0   N/A  N/A            6812    C+G   ...yb3d8bbwe\Microsoft.Notes.exe      N/A
|    0   N/A  N/A            7608    C+G   ...5n1h2txyewy\TextInputHost.exe      N/A
|    0   N/A  N/A           14904    C+G   ...mba6cd70vzyy\ArmouryCrate.exe      N/A
|    0   N/A  N/A           17032    C+G   ....0.3179.54\msedgewebview2.exe      N/A
|    0   N/A  N/A           19784    C+G   ...\Creative Cloud UI Helper.exe      N/A
|    0   N/A  N/A           20568    C+G   ...c\AdobeNotificationClient.exe      N/A
|    0   N/A  N/A           26876    C+G   ...4__cv1g1gvanyjgm\WhatsApp.exe      N/A
|    0   N/A  N/A           28300    C+G   ....0.3179.54\msedgewebview2.exe      N/A
|    0   N/A  N/A           28708    C+G   ...y\StartMenuExperienceHost.exe      N/A
|    0   N/A  N/A           29008    C+G   ...8bbwe\PhoneExperienceHost.exe      N/A
|    0   N/A  N/A           31980    C+G   ...ogram Files\Zotero\zotero.exe      N/A
|    0   N/A  N/A           34712    C+G   ...ms\Microsoft VS Code\Code.exe      N/A
|    0   N/A  N/A           35000    C+G   ...App_cw5n1h2txyewy\LockApp.exe      N/A
+-----------------------------------------------------------------------------------
```

```
## Hardware Specifications
import platform
import psutil

print("Operating System:", platform.system(), platform.version())
print("Processor:", platform.processor())
print("CPU Cores:", psutil.cpu_count(logical=False))
print("Logical CPUs:", psutil.cpu_count(logical=True))
print("RAM:", round(psutil.virtual_memory().total / (1024**3), 2), "GB")
```

```
Operating System: Windows 10.0.26100
Processor: Intel64 Family 6 Model 140 Stepping 1, GenuineIntel
CPU Cores: 4
Logical CPUs: 8
RAM: 23.7 GB
```

```
# USE CUDA
import torch
print("PyTorch CUDA available:", torch.cuda.is_available())
print("CUDA device:", torch.cuda.get_device_name(0) if torch.cuda.is_available() else "No GF
```

```
PyTorch CUDA available: True
CUDA device: NVIDIA GeForce RTX 3050 Ti Laptop GPU
```

Now we can . Before that, We need to track the carbon emmisions.

```
!pip install codecarbon
clear_output()
```

2.5.1.0 - Abstract Only

```
import time
from codecarbon import OfflineEmissionsTracker
from bertopic import BERTopic
import torch

# Check if GPU is available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
# Print whether the GPU is ready or not
if device.type == "cuda":
    print("GPU ready")
else:
    print("Using CPU")

# Start the experiment and emissions tracker
start_time = time.time()
tracker = OfflineEmissionsTracker(country_iso_code="IRL", allow_multiple_runs=True)
tracker.start()

# Instantiate BERTopic
topic_model = BERTopic()
topics, probs = topic_model.fit_transform(cleaned_abstract_lists)

# Stop emissions tracking
tracker.stop()

# end time
end_time = time.time()
duration = end_time - start_time

print(f"Experiment took {duration} seconds")
```

```
GPU ready
Experiment took 269.655246257782 seconds
```

```
# Save model
topic_model.save("models/abstracts-model")
```

```
2025-04-11 14:13:37,291 - BERTopic - WARNING: When you use `pickle` to save/load a BERTo
```

2.5.1.1 - Topic Over Time(Abstract)

```
timestamps = pd.read_csv('timestamps.csv')

# datetime_format = "%Y-%m-%d"
# topics_over_time = topic_model.topics_over_time(tweets, timestamps, datetime_format="%b%M"
# then this
# topic_model.visualize_topics_over_time(topics_over_time, top_n_topics=20)
```

```
timestamps["publish_time"] = pd.to_datetime(timestamps["publish_time"], errors="coerce")
```

```
timestamps = timestamps[:100_000]
timestamps = timestamps['publish_time'].tolist()
```

```
timestamps
```

```
#### TOPICS OVER TIME
topics_over_time = topic_model.topics_over_time(cleaned_abstract_lists, timestamps)
```

2.5.2.0 - Title Only

```
# Start emissions tracker
start_time = time.time()
tracker = OfflineEmissionsTracker(country_iso_code="IRL",allow_multiple_runs=True)
tracker.start()

# Instatiate BERTopic
topic_model = BERTopic()
topics, probs = topic_model.fit_transform(cleaned_title_lists)

# Stop emissions tracking
tracker.stop()

# Record the duration
end_time = time.time()
duration = end_time - start_time

print(f"Experiment took {duration} seconds")
```

```
Experiment took 79.7363293170929 seconds
```

```
# Save model
topic_model.save("models/titles-model")
```

```
2025-04-11 14:15:13,894 - BERTopic - WARNING: When you use `pickle` to save/load a BERTc
```

2.5.3.0 - Title + Abstract

```
# Start the experiment and emissions tracker
start_time = time.time()
tracker = OfflineEmissionsTracker(country_iso_code="IRL",allow_multiple_runs=True)
tracker.start()

# Instatiate BERTopi
topic_model = BERTopic()
topics, probs = topic_model.fit_transform(cleaned_titleab_lists)

# Stop emissions tracking
```

```
tracker.stop()

# Record the duration
end_time = time.time()
duration = end_time - start_time
print(f"Experiment took {duration} seconds")
```

```
Experiment took 253.95688724517822 seconds
```

```
# Save model
topic_model.save("models/abstractitles-model")
```

```
2025-04-11 14:19:35,800 - BERTopic - WARNING: When you use `pickle` to save/load a BERTo
```

2.6 - Model Loading

Now we can *proceed with analyzing the results* from the trained model.

📌 NOTE:

We have already trained the model, so if you're not training it from scratch, you can skip the training code. Instead, we'll be loading the pre-trained model to proceed with generating the results.

```
# title only
title_model = BERTopic.load("models/titles-model")
# title and abstract
ab_tit_model= BERTopic.load("models/abstractitles-model")
# abstract only
abstract_model= BERTopic.load("models/abstracts-model")
```

```
# Topic Over Time ---  Not yet ready
# time_title_model = BERTopic.load("models/titles-model")
# time_ab_tit_model= BERTopic.load("models/abstractitles-model")
# time_abstract_model= BERTopic.load("models/abstracts-model")
```

2.7 - Results - Visualisations, Others

2.7.1 - Abstract Topic Modelling

```
# Generate topic labels
topic_labels = abstract_model.generate_topic_labels(nr_words = 5, topic_prefix= False, word_
abstract_model.set_topic_labels(topic_labels)
```

Basically made it simpler to read bu adding a separator "-"

```
# see first 10 topics
display(topic_labels[:10])
```

```
['cell - health - sar - virus - protein',
 'student - teaching - education - teacher - learn',
 'child - pediatric - multisystem - pim - inflammatory',
 'thrombosis - vte - anticoagulation - thrombotic - coagulation',
 'ct - chest - consolidation - opacity - lung',
 'epidemic - model - mathematical - reproduction - parameter',
 'image - ray - deep - segmentation - dataset',
 'anxiety - psychological - depression - mental - fear',
 'privacy - iot - blockchain - app - tracing',
 'protease - compound - drug - mpro - docking']
```

```
abstract_model.get_topic_info()
```

	Topic	Count	Name	CustomName	Representation
0	-1	45784	-1_cell_health_sar_virus	cell - health - sar - virus - protein	[cell, health, sar, virus, protein, infection,...
1	0	3922	0_student_teaching_education_teacher	student - teaching - education - teacher - learn	[student, teaching, education, teacher, learn,...
2	1	878	1_child_pediatric_multisystem_pim	child - pediatric - multisystem - pim - inflam...	[child, pediatric, multisystem, pim, inflammat...
3	2	810	2_thrombosis_vte_anticoagulation_thrombotic	thrombosis - vte - anticoagulation - thromboti...	[thrombosis, vte, anticoagulation, thrombotic,...
4	3	785	3_ct_chest_consolidation_opacity	ct - chest - consolidation - opacity - lung	[ct, chest, consolidation, opacity, lung, ggo,...
...	...	...	...	...	...
730	729	10	729_conference_meeting_hold_symposium	conference - meeting - hold - symposium - annual	[conference, meeting, hold, symposium, annual,...
				ecrswnp - allergic -	[ecrswnp,

Intertopic Distance Map

is a visual representation that show in an intuitive way, how to understand the relationships between topics discovered by the model.

```
# intertopic distance map
abstract_model.visualize_topics()
```

This map helps us understand the similarities and differences between the topics.

- Topics that are closer together indicate that they are more similar, while topics that are farther apart are more distinct or dissimilar.

Topic Word Scores

```
abstract_model.visualize_barchart( width= 280 , height= 330 , top_n_topics= 10 , n_words  =
```

```
abstract_model.visualize_documents(cleaned_abstract_lists, topics = list(range(30)), custom_
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File ~\AppData\Roaming\Python\Python312\site-
packages\IPython\core\formatters.py:984, in IPythonDisplayFormatter.__call__(self,
obj)
    982 method = get_real_method(obj, self.print_method)
    983 if method is not None:
--> 984     method()
    985     return True

File c:\Users\david\AppData\Local\Programs\Python\Python312\Lib\site-
packages\plotly\basedatatypes.py:833, in BaseFigure._ipython_display_(self)
    830 import plotly.io as pio
    832 if pio.renderers.render_on_display and pio.renderers.default:
--> 833     pio.show(self)
    834 else:
    835     print(repr(self))

File c:\Users\david\AppData\Local\Programs\Python\Python312\Lib\site-
packages\plotly\io\_renderers.py:425, in show(fig, renderer, validate, **kwargs)
    420     raise ValueError(
    421         "Mime type rendering requires ipython but it is not installed"
    422     )
    424 if not nbformat or Version(nbformat.__version__) < Version("4.2.0"):
--> 425     raise ValueError(
    426         "Mime type rendering requires nbformat>=4.2.0 but it is not
installed"
    427     )
    429 display_jupyter_version_warnings()
    431 ipython_display.display(bundle, raw=True)

ValueError: Mime type rendering requires nbformat>=4.2.0 but it is not installed
```

Documents and

2.7.2 - Titles Topic Modelling

```
# Generate topic labels
topic_labels = title_model.generate_topic_labels(nr_words = 5, topic_prefix= False, word_ler
title_model.set_topic_labels(topic_labels)
```

```
# see first 10 topics
display(topic_labels[:10])
```

```
['social - virus - protein - model - viral',
 'food - nutrition - eat - insecurity - nutritional',
 'teaching - teacher - online - education - distance',
 'sar - transmission - genomic - lineage - genome',
 'ray - image - deep - chest - convolutional',
 'telemedicine - telehealth - telepsychiatry - tele - teleconsultatio',
 'pneumonia - acquire - ct - chest - ultrasound',
 'pregnancy - pregnant - maternal - obstetric - birth',
 'covid - covid19 - post - long - toe',
 'coronavirus - disease - characteristic - novel - epidemiological']
```

```
title_model.get_topic_info()
```

	Topic	Count	Name	CustomName	Representa
0	-1	41307	-1_social_virus_protein_model	social - virus - protein - model - viral	[social, protein, r viral, he
1	0	1056	0_food_nutrition_eat_insecurity	food - nutrition - eat - insecurity - nutritional	[food, nut eat, inse nutriti
2	1	942	1_teaching_teacher_online_education	teaching - teacher - online - education - dist...	[tea teacher, c educ dis
3	2	875	2_sar_transmission_genomic_lineage	sar - transmission - genomic - lineage - genome	transmi genomic, lir geno
4	3	782	3_ray_image_deep_chest	ray - image - deep - chest - convolutional	[ray, image, convolu se
...	...	...	...	...	
1049	1048	10	1048_depression_anxiety_phenomenon_precaution	depression - anxiety - phenomenon - precaution...	[depre a phenom precaut

Intertopic distance map

```
# intertopic distance map
title_model.visualize_topics()
```

Topic Word Scores

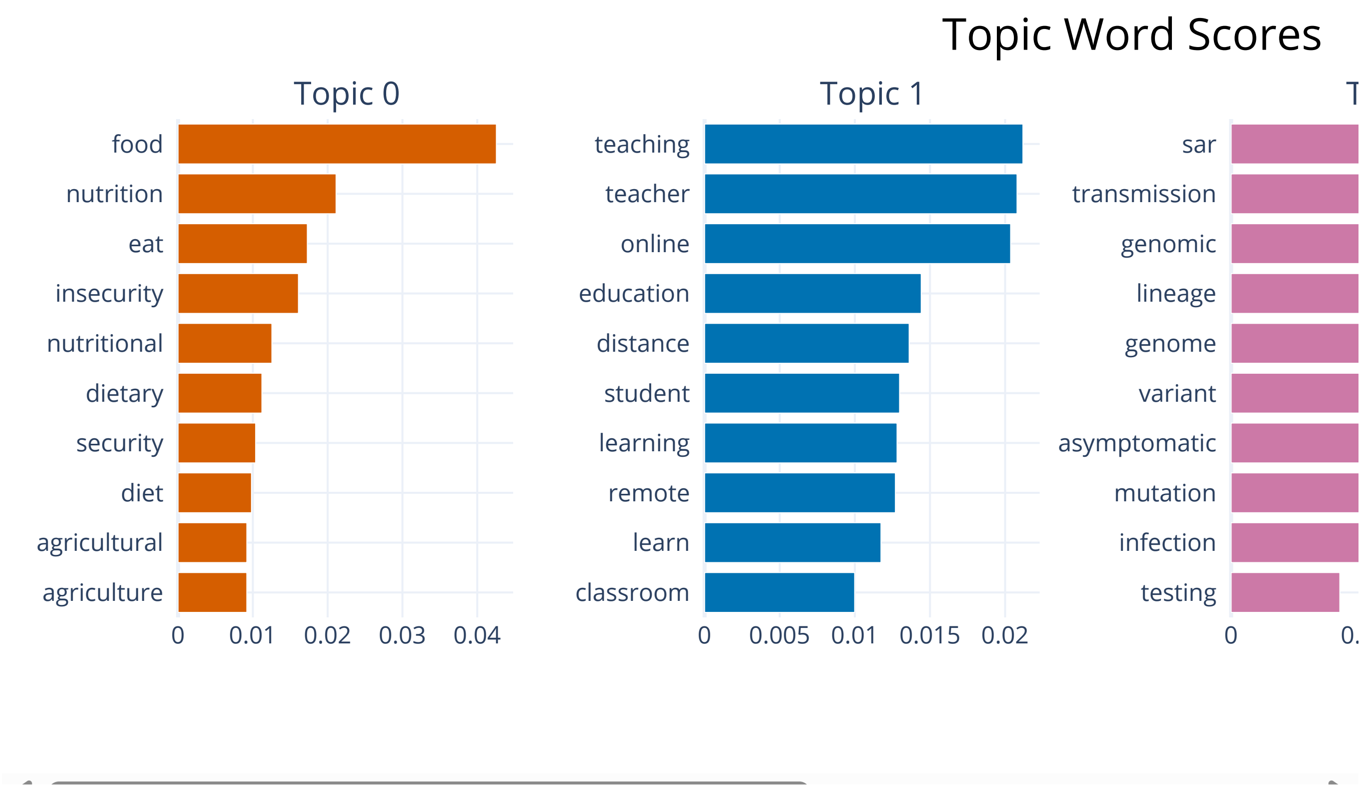
```
title_model.visualize_barchart( width= 280 , height= 330 , top_n_topics= 10 , n_words  = 10)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File ~\AppData\Roaming\Python\Python312\site-packages\IPython\core\formatters.py:984,
in IPythonDisplayFormatter.__call__(self, obj)
    982 method = get_real_method(obj, self.print_method)
    983 if method is not None:
--> 984     method()
    985     return True

File c:\Users\david\AppData\Local\Programs\Python\Python312\Lib\site-
packages\plotly\basedatatypes.py:833, in BaseFigure._ipython_display_(self)
    830 import plotly.io as pio
    832 if pio.renderers.render_on_display and pio.renderers.default:
--> 833     pio.show(self)
    834 else:
    835     print(repr(self))

File c:\Users\david\AppData\Local\Programs\Python\Python312\Lib\site-
packages\plotly\io\_renderers.py:425, in show(fig, renderer, validate, **kwargs)
    420     raise ValueError(
    421         "Mime type rendering requires ipython but it is not installed"
    422     )
    424 if not nbformat or Version(nbformat.__version__) < Version("4.2.0"):
--> 425     raise ValueError(
    426         "Mime type rendering requires nbformat>=4.2.0 but it is not installed"
    427     )
    429 display_jupyter_version_warnings()
    431 ipython_display.display(bundle, raw=True)

ValueError: Mime type rendering requires nbformat>=4.2.0 but it is not installed
```



Word Graph

```
title_model.visualize_documents(cleaned_title_lists, topics = list(range(30)), custom_labels
```

2.7.3 - Abstract and Titles Topic Modelling

```
# Generate topic labels
topic_labels = ab_tit_model.generate_topic_labels(nr_words = 5, topic_prefix= False, word_le
ab_tit_model.set_topic_labels(topic_labels)
```

```
# see first 10 topics
display(topic_labels[:10])
```

```
['health - cell - study - datum - pandemic',
 'teacher - teaching - learning - learn - student',
 'child - pediatric - multisystem - inflammatory - pim',
 'replication - glycan - protein - rna - virus',
 'image - ray - deep - segmentation - chest'
```