[1]
```
import json
import pandas as pd
import random
import os
import zipfile
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]
```
with zipfile.ZipFile("drive/Shared drives/810/pmc_json.zip","r") as zf:
    zf.extractall('ppmc')
```

[4]
```
!pip3 install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.2.1.tar.gz (281.4 MB)
     |████████████████████████████████| 281.4 MB 38 kB/s
Collecting py4j==0.10.9.3
  Downloading py4j-0.10.9.3-py2.py3-none-any.whl (198 kB)
     |████████████████████████████████| 198 kB 51.2 MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.2.1-py2.py3-none-any.whl size=281853642 sha256=06ad659e14511964b01848d8a538c61d8ae790292b0d6ad4f5fe31cf5c755f45
  Stored in directory: /root/.cache/pip/wheels/9f/f5/07/7cd8017084dce4e93e84e92efd1e1d5334db05f2e83bcef74f
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.3 pyspark-3.2.1
```

```
from pyspark.sql.functions import lit
from pyspark.sql.types import (
    ArrayType,
    IntegerType,
    MapType,
    StringType,
    StructField,
    StructType,
)


def generate_cord19_schema():

  author_fields = [
        StructField("first", StringType()),
        StructField("middle", ArrayType(StringType())),
        StructField("last", StringType()),
        StructField("suffix", StringType()),
    ]

  authors_schema = ArrayType(
      StructType(
          author_fields
          + [
              # Uncomment to cast field into a JSON string. This field is not
              # well-specified in the source.
              StructField(
                  "affiliation",
                  StructType(
                      [
                          StructField("laboratory", StringType()),
                          StructField("institution", StringType()),
                          StructField(
                              "location",
                              StructType(
                                  [
                                      StructField("settlement", StringType()),
                                      StructField("country", StringType()),
                                  ]
                              ),
                          ),
                      ]
                  ),
              ),
              StructField("email", StringType()),
          ]
      )
  )
```

```
# used in `section_schema` for citations, references, and equations
spans_schema = ArrayType(
    StructType(
        [
            # character indices of inline citations
            StructField("start", IntegerType()),
            StructField("end", IntegerType()),
            StructField("text", StringType()),
            StructField("ref_id", StringType()),
        ]
    )
)

# A section of the paper, which includes the abstract, body, and back matter.
section_schema = ArrayType(
    StructType(
        [
            StructField("text", StringType()),
            StructField("cite_spans", spans_schema),
            StructField("ref_spans", spans_schema),
            # While equations don't appear in the abstract, but appear here
            # for consistency
            StructField("eq_spans", spans_schema),
            StructField("section", StringType()),
        ]
    )
)

bib_schema = MapType(
    StringType(),
    StructType(
        [
            StructField("ref_id", StringType()),
            StructField("title", StringType()),
            StructField("authors", ArrayType(StructType(author_fields))),
            StructField("year", IntegerType()),
            StructField("venue", StringType()),
            StructField("volume", StringType()),
            StructField("issn", StringType()),
            StructField("pages", StringType()),
            StructField(
                "other_ids",
                StructType([StructField("DOI", ArrayType(StringType()))]),
            ),
        ]
    ),
    True,
)
```

[5]

```
# Can be one of table or figure captions
ref_schema = MapType(
    StringType(),
    StructType(
        [
            StructField("text", StringType()),
            # Likely equation spans, not included in source schema, but
            # appears in JSON
            StructField("latex", StringType()),
            StructField("type", StringType()),
        ]
    ),
)

return StructType(
    [
        StructField("paper_id", StringType()),
        StructField(
            "metadata",
            StructType(
                [
                    StructField("title", StringType()),
                    StructField("authors", authors_schema),
                ]
            ),
            True,
        ),

        StructField("body_text", section_schema),
        StructField("bib_entries", bib_schema),
        StructField("ref_entries", ref_schema),
        StructField("back_matter", section_schema),
    ]
)
```

[6]
```
def extract_dataframe_kaggle(spark):
    """Extract a structured DataFrame from the semi-structured document dump.

    It should be fairly straightforward to modify this once there are new
    documents available. The date of availability (`crawl_date`) and `source`
    are available as metadata.
    """
    base = "ppmc/pmc_json"

    dataframe = None

    path = f"{base}/"
    df = (spark.read.json(path, schema=generate_cord19_schema(), multiLine=True)
        )
    if not dataframe:
        dataframe = df
    else:
        dataframe = dataframe.union(df)
    return dataframe
```

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
MAX_MEMORY = "50g"
spark = SparkSession \
    .builder \
    .appName("sparkdf") \
    .config("spark.executor.memory", MAX_MEMORY) \
    .config("spark.driver.memory", MAX_MEMORY) \
    .config("spark.memory.offHeap.enabled","true")\
    .config("spark.memory.offHeap.size","50g") \
    .config("spark.ui.port", "4040") \
    .master("local[*]") \
    .getOrCreate()
spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
`v3.2.1`
Master
`local[*]`
AppName
`sparkdf`

```python
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

df = extract_dataframe_kaggle(spark)
df.printSchema()

df.createOrReplaceTempView("cord19")
```

```
root
 |-- paper_id: string (nullable = true)
 |-- metadata: struct (nullable = true)
 |    |-- title: string (nullable = true)
 |    |-- authors: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- first: string (nullable = true)
 |    |    |    |-- middle: array (nullable = true)
 |    |    |    |    |-- element: string (containsNull = true)
 |    |    |    |-- last: string (nullable = true)
 |    |    |    |-- suffix: string (nullable = true)
 |    |    |    |-- affiliation: struct (nullable = true)
 |    |    |    |    |-- laboratory: string (nullable = true)
 |    |    |    |    |-- institution: string (nullable = true)
 |    |    |    |    |-- location: struct (nullable = true)
 |    |    |    |    |    |-- settlement: string (nullable = true)
 |    |    |    |    |    |-- country: string (nullable = true)
 |    |    |    |-- email: string (nullable = true)
 |-- body_text: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- text: string (nullable = true)
 |    |    |-- cite_spans: array (nullable = true)
 |    |    |    |-- element: struct (containsNull = true)
 |    |    |    |    |-- start: integer (nullable = true)
 |    |    |    |    |-- end: integer (nullable = true)
 |    |    |    |    |-- text: string (nullable = true)
 |    |    |    |    |-- ref_id: string (nullable = true)
 |    |    |-- ref_spans: array (nullable = true)
 |    |    |    |-- element: struct (containsNull = true)
 |    |    |    |    |-- start: integer (nullable = true)
 |    |    |    |    |-- end: integer (nullable = true)
 |    |    |    |    |-- text: string (nullable = true)
 |    |    |    |    |-- ref_id: string (nullable = true)
 |    |    |-- eq_spans: array (nullable = true)
 |    |    |    |-- element: struct (containsNull = true)
 |    |    |    |    |-- start: integer (nullable = true)
 |    |    |    |    |-- end: integer (nullable = true)
 |    |    |    |    |-- text: string (nullable = true)
 |    |    |    |    |-- ref_id: string (nullable = true)
 |    |    |-- section: string (nullable = true)
```

```
 |-- bib_entries: map (nullable = true)
 |    |-- key: string
 |    |-- value: struct (valueContainsNull = true)
 |    |    |-- ref_id: string (nullable = true)
 |    |    |-- title: string (nullable = true)
 |    |    |-- authors: array (nullable = true)
 |    |    |    |-- element: struct (containsNull = true)
 |    |    |    |    |-- first: string (nullable = true)
 |    |    |    |    |-- middle: array (nullable = true)
 |    |    |    |    |    |-- element: string (containsNull = true)
 |    |    |    |    |-- last: string (nullable = true)
 |    |    |    |    |-- suffix: string (nullable = true)
 |    |    |-- year: integer (nullable = true)
 |    |    |-- venue: string (nullable = true)
 |    |    |-- volume: string (nullable = true)
 |    |    |-- issn: string (nullable = true)
 |    |    |-- pages: string (nullable = true)
 |    |    |-- other_ids: struct (nullable = true)
 |    |    |    |-- DOI: array (nullable = true)
 |    |    |    |    |-- element: string (containsNull = true)
 |-- ref_entries: map (nullable = true)
 |    |-- key: string
 |    |-- value: struct (valueContainsNull = true)
 |    |    |-- text: string (nullable = true)
 |    |    |-- latex: string (nullable = true)
 |    |    |-- type: string (nullable = true)
 |-- back_matter: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- text: string (nullable = true)
 |    |    |-- cite_spans: array (nullable = true)
 |    |    |    |-- element: struct (containsNull = true)
 |    |    |    |    |-- start: integer (nullable = true)
 |    |    |    |    |-- end: integer (nullable = true)
 |    |    |    |    |-- text: string (nullable = true)
 |    |    |    |    |-- ref_id: string (nullable = true)
 |    |    |-- ref_spans: array (nullable = true)
 |    |    |    |-- element: struct (containsNull = true)
 |    |    |    |    |-- start: integer (nullable = true)
 |    |    |    |    |-- end: integer (nullable = true)
 |    |    |    |    |-- text: string (nullable = true)
 |    |    |    |    |-- ref_id: string (nullable = true)
 |    |    |-- eq_spans: array (nullable = true)
 |    |    |    |-- element: struct (containsNull = true)
 |    |    |    |    |-- start: integer (nullable = true)
 |    |    |    |    |-- end: integer (nullable = true)
 |    |    |    |    |-- text: string (nullable = true)
 |    |    |    |    |-- ref_id: string (nullable = true)
 |    |    |-- section: string (nullable = true)
```

```
] df.show(3)
```

```
] df.count()
```

```
[9] from pyspark.sql import Window

    body_text = (
        df.select("paper_id", F.posexplode("body_text").alias("pos", "value"))
        .select("paper_id", "pos", "value.text")
        .withColumn("ordered_text", F.collect_list("text").over(Window.partitionBy("paper_id").orderBy("pos")))
        .groupBy("paper_id")
        .agg(F.max("ordered_text").alias("sentences"))
        .select("paper_id", F.array_join("sentences", " ").alias("body_text"))
        .withColumn("words", F.size(F.split("body_text", "\s+")))
    )

    # body_text.show(n=5)
```

```
[10] import pyspark
     def dbg(x):
       """ A helper function to print debugging information on RDDs """
       if isinstance(x, pyspark.RDD):
         print([(t[0], list(t[1]) if
                 isinstance(t[1], pyspark.resultiterable.ResultIterable) else t[1])
                if isinstance(t, tuple) else t
                for t in x.take(100)])
       else:
         print(x)
```

```
dbg(body_text)
```

```
DataFrame[paper_id: string, body_text: string, words: int]
```

```
[12] def get_headlines(row):
         return row['body_text']
```

```
[13] body_text1000 = body_text.limit(1000)
```

```
[14] bodytextrdd1000 = body_text1000.rdd.map(get_headlines)
```

```
[16] bodytextlist=[bodytextrdd1000.collect()]
```

```
[17] paperidrdd1000=body_text1000.select("paper_id").rdd.flatMap(lambda x: x).collect()
```

```
[18] paperidlist=[paperidrdd1000]
```

```
[19] import nltk
     nltk.download('vader_lexicon')
     from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA

     sia = SIA()
     results = []
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
/usr/local/lib/python3.7/dist-packages/nltk/twitter/__init__.py:20: UserWarning: The twython library has not been installed. Some functionality from the twitter package will not be available.
  warnings.warn("The twython library has not been installed. "
```

```
results=[]
for lines,paperid in zip(bodytextlist,paperidlist):

  for line,id in zip(lines,paperid) :

    try:
      pol_score = sia.polarity_scores(line)
      pol_score['body_text'] = line
      pol_score['paper_id']=id
      results.append(pol_score)
    except:
      print('Skipped')
```

```
[25] import numpy as np
     df_result = pd.DataFrame(results)

     df_result['label'] = np.zeros(len(df_result)).tolist()
     df_result.loc[df_result['compound'] > 0.98, 'label'] = 1
     df_result.loc[df_result['compound'] < -0.98 , 'label'] = -1
     df_result.to_csv('body_text_results.csv')
```

```
df_result
```

	neg	neu	pos	compound	body_text	paper_id	label
0	0.017	0.922	0.061	0.9997	Severe acute respiratory syndrome (SARS) first...	PMC7152143	1.0
1	0.053	0.893	0.054	-0.9862	Virus-host interactions are crucial for the ou...	PMC7130089	-1.0
2	0.036	0.897	0.066	0.9992	The control of diseases associated with highly...	PMC8652887	1.0
3	0.020	0.857	0.124	0.9999	SARS is a new infectious disease that emerged ...	PMC7995808	1.0
4	0.043	0.872	0.085	0.9991	Hepatitis C virus (HCV, a member of the Flaviv...	PMC7158344	1.0

```
df_result
```

	neg	neu	pos	compound	body_text	paper_id	label
0	0.017	0.922	0.061	0.9997	Severe acute respiratory syndrome (SARS) first...	PMC7152143	1.0
1	0.053	0.893	0.054	-0.9862	Virus-host interactions are crucial for the ou...	PMC7130089	-1.0
2	0.036	0.897	0.066	0.9992	The control of diseases associated with highly...	PMC8652887	1.0
3	0.020	0.857	0.124	0.9999	SARS is a new infectious disease that emerged ...	PMC7995808	1.0
4	0.043	0.872	0.085	0.9991	Hepatitis C virus (HCV, a member of the Flaviv...	PMC7158344	1.0
...	...	...	...	...	...	...	...
995	0.036	0.921	0.044	0.9615	Acute appendicitis (AA) is the most common gen...	PMC7978675	0.0
996	0.090	0.818	0.092	0.9258	The emergence of novel coronavirus disease 201...	PMC6357155	0.0
997	0.111	0.857	0.033	-0.9998	In December 2019, an outbreak of a novel coron...	PMC8307643	-1.0
998	0.029	0.889	0.083	0.9985	The World Health Organization (WHO) has been e...	PMC8398366	1.0
999	0.028	0.927	0.045	0.9992	Design automation and safety of autonomous sys...	PMC4252826	1.0

1000 rows × 7 columns

```
[ ] spark.stop()
```