I. Installing the Unified Medical Language System (UMLS)

In [1]:
```
%load_ext autoreload
%autoreload 2

import sys
sys.path.insert(0,'../../trove')
```

A. Download the UMLS Release Files

Trove requires access to the Unified Medical Language System (UMLS) which is freely available after signing up for an account with the National Library of Medicine (NLM).

Visit the link below and download the latest "UMLS Metathesaurus Files" release 2020AB. This file is quite large (5.3 GB compressed), so it may take some time to download. **Please note, "full" release zip files are currently not supported.**

Alternatively, if you have an existing API KEY you can use the following script to download the zip file from the command line. See https://documentation.uts.nlm.nih.gov/automating-downloads.html for technical details on NLM authentication.

```
python download_umls.py \
   --apikey XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX \
   --url https://download.nlm.nih.gov/umls/kss/2020AB/umls-2020AB-
metathesaurus.zip
```

B. Installation

Currently there are 2 ways to initalize the UMLS:

- From the source zip file (e.g., `umls-2020AB-metathesaurus.zip`)
- From the Rich Release Format (RRF) files generated by MetamorphoSys
- **TBD** An existing database instance

Depending on your machine, this should take 2-5 minutes.

Option 1: Install from NLM Zip File

In [2]:
```
%%time
from trove.labelers.umls import UMLS

# setup defaults
UMLS.config(
    cache_root = "~/.trove/umls2022AB",
```

```
    backend = 'pandas'
)

NLM_ZIPFILE_PATH = "umls-2020AB-metathesaurus.zip"#edited this line from full ish
if not UMLS.is_initalized():
    print("Initializing the UMLS from zip file...")
    UMLS.init_from_nlm_zip(NLM_ZIPFILE_PATH)
```

```
Initializing the UMLS from zip file...
```

```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
File <timed exec>:12, in <module>

File ~\Hubs\trove\tutorials\../../trove\trove\labelers\umls.py:326, in UMLS.init_from
_nlm_zip(fpath, outdir, backend, use_checksum, keep_original_rrfs)
    323         outfile.write_bytes(zipfile.read(fname))
    325 # init from RRFs
--> 326 UMLS.init_from_rrfs(tmp, outdir, backend)
    327 if not keep_original_rrfs:
    328     shutil.rmtree(tmp)

File ~\Hubs\trove\tutorials\../../trove\trove\labelers\umls.py:386, in UMLS.init_from
_rrfs(indir, outdir, backend)
    383 with open(f'{indir}/MRCONSO.RRF', 'r') as fp, open(
    384         f'{outdir}/concepts.tsv', 'w') as op:
    385     op.write('SAB\tTUI\tCUI\tTERM\n')
--> 386     for line in fp:
    387         row = line.strip().split('|')
    388         cui, sab, term = row[0], row[11], row[14]

File ~\Miniconda3\envs\navidi\lib\encodings\cp1252.py:23, in IncrementalDecoder.decod
e(self, input, final)
     22 def decode(self, input, final=False):
---> 23     return codecs.charmap_decode(input,self.errors,decoding_table)[0]

UnicodeDecodeError: 'charmap' codec can't decode byte 0x81 in position 4213: characte
r maps to <undefined>
```

In [5]:
```
from platform import python_version
python_version()
```

Out[5]:
```
'3.9.7'
```

In [4]:
```
#see libraries below
!pip3 list
```

```
Package                 Version
----------------------- ------------------
absl-py                 0.13.0
aiobotocore             2.2.0
aiohttp                 3.8.1
aioitertools            0.10.0
aiosignal               1.2.0
argon2-cffi             21.3.0
argon2-cffi-bindings    21.2.0
asttokens               2.0.5
astunparse              1.6.3
async-timeout           4.0.2
attrs                   21.4.0
backcall                0.2.0
bleach                  4.1.0
botocore                1.24.21
cachetools              4.2.4
certifi                 2021.10.8
cffi                    1.15.0
charset-normalizer      2.0.12
colorama                0.4.4
cramjam                 2.5.0
cvxpy                   1.2.0
cycler                  0.11.0
DateTime                4.4
debugpy                 1.5.1
decorator               5.1.1
defusedxml              0.7.1
ecos                    2.0.10
entrypoints             0.4
executing               0.8.3
fastparquet             0.8.0
flatbuffers             1.12
fonttools               4.29.1
frozenlist              1.3.0
fsspec                  2022.3.0
gast                    0.4.0
google-auth             1.35.0
google-auth-oauthlib    0.4.4
google-pasta            0.2.0
grpcio                  1.34.1
h5py                    3.1.0
idna                    3.3
ipykernel               6.9.1
ipython                 8.1.1
ipython-genutils        0.2.0
ipywidgets              7.6.5
jedi                    0.18.1
Jinja2                  3.0.3
jmespath                1.0.0
joblib                  1.1.0
jsonschema              4.4.0
jupyter                 1.0.0
jupyter-client          7.1.2
jupyter-console         6.4.0
jupyter-core            4.9.2
jupyterlab-pygments     0.1.2
jupyterlab-widgets      1.0.2
keras-nightly           2.5.0.dev2021032900
Keras-Preprocessing     1.1.2
```

```
kiwisolver              1.3.2
Markdown                3.3.4
MarkupSafe              2.1.0
matplotlib              3.5.1
matplotlib-inline       0.1.3
mistune                 0.8.4
msgpack                 1.0.4
multidict               6.0.2
nbclient                0.5.11
nbconvert               6.4.2
nbformat                5.1.3
nest-asyncio            1.5.4
notebook                6.4.8
numpy                   1.19.5
oauthlib                3.2.0
opt-einsum              3.3.0
osqp                    0.6.2.post5
packaging               21.3
pandas                  1.4.1
pandocfilters           1.5.0
parso                   0.8.3
patsy                   0.5.2
pickleshare             0.7.5
Pillow                  9.0.1
pip                     21.2.4
plotly                  5.6.0
prometheus-client       0.13.1
prompt-toolkit          3.0.28
protobuf                3.19.4
pure-eval               0.2.2
pyasn1                  0.4.8
pyasn1-modules          0.2.8
pycparser               2.21
Pygments                2.11.2
pyparsing               3.0.7
pyreadstat              1.1.4
pyrsistent              0.18.1
python-dateutil         2.8.2
pytz                    2021.3
pywin32                 303
pywinpty                2.0.5
pyzmq                   22.3.0
qdldl                   0.1.5.post0
qtconsole               5.2.2
QtPy                    2.0.1
requests                2.27.1
requests-oauthlib       1.3.1
rsa                     4.8
s3fs                    2022.3.0
scikit-learn            1.0.2
scipy                   1.8.0
scs                     3.2.0
seaborn                 0.11.2
Send2Trash              1.8.0
setuptools              58.0.4
six                     1.15.0
sklearn                 0.0
stack-data              0.2.0
statsmodels             0.13.2
tenacity                8.0.1
```

```
tensorboard             2.5.0
tensorboard-data-server 0.6.1
tensorboard-plugin-wit  1.8.1
tensorflow              2.5.0
tensorflow-estimator    2.5.0
termcolor               1.1.0
terminado               0.13.2
testpath                0.6.0
threadpoolctl           3.1.0
torch                   1.11.0
torchvision             0.12.0
tornado                 6.1
traitlets               5.1.1
typing_extensions       4.1.1
urllib3                 1.26.8
wcwidth                 0.2.5
webencodings            0.5.1
Werkzeug                2.0.3
wheel                   0.37.1
widgetsnbextension      3.5.2
wincertstore            0.2
wrapt                   1.12.1
yarl                    1.7.2
zope.interface          5.4.0
```

Option 2: Install from RRF Files

If you have installed the UMLS before using MetamorphoSys to create custom vocabulary subsets you can directly use the generated RRF files.

In [4]:
```
%%time

RRF_FILE_PATH = ""
if not UMLS.is_initalized():
    print("Initializing the UMLS from RRFs...")
    UMLS.init_from_rrfs(RRF_FILE_PATH)
```

```
Initializing the UMLS from RRFs...
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
File <timed exec>:4, in <module>

File ~\Hubs\trove\tutorials\../../trove\trove\labelers\umls.py:349, in UMLS.init_from_rrfs(indir, outdir, backend)
    347 for fname in ['MRCONSO.RRF', 'MRSTY.RRF', 'MRSAB.RRF']:
    348     if not os.path.exists(f"{indir}/{fname}"):
--> 349         raise FileNotFoundError(
    350             errno.ENOENT,
    351             os.strerror(errno.ENOENT),
    352             fname
    353         )
    355 # Source terminologies - MRSAB.RRF
    356 sabs = {}

FileNotFoundError: [Errno 2] No such file or directory: 'MRCONSO.RRF'
```

Option 3: Install from an Existing Database Instance

TBD: If you have a live UMLS database instance running, you can initialize Trove as follows.

In [1]:
```
# if not UMLS.is_initalized():
#     UMLS.init_from_dbconn(engine='mysql', dbname='UMLS2020AB')
```

3. Test the Installation

Here we apply some common term transformations. This should run in 2-4 minutes.

In [5]:
```
%%time
from trove.labelers.umls import UMLS
from trove.transforms import SmartLowercase

# english stopwords
stopwords = set(open('data/stopwords.txt','r').read().splitlines())
stopwords = stopwords.union(set([t[0].upper() + t[1:] for t in stopwords]))

# options for filtering terms
config = {
    "type_mapping"  : "TUI",  # TUI = semantic types, CUI = concept ids
    'min_char_len'  : 2,
    'max_tok_len'   : 8,
    'min_dict_size' : 500,
    'stopwords'     : stopwords,
    'transforms'    : [SmartLowercase()],
    'languages'     : {"ENG"},
    'filter_sabs'   : {"SNOMEDCT_VET"},
    'filter_rgx'    : r'''^[-+]*[0-9]+([.][0-9]+)*$'''  # filter numbers
}

umls = UMLS(**config)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
File <timed exec>:5, in <module>

FileNotFoundError: [Errno 2] No such file or directory: 'data/stopwords.txt'
```

Look at semantic type assignments for an example term `acetaminophen` from the Medical Subject Headings (MeSH®) terminology.

In [6]:
```
from trove.labelers.umls import SemanticGroups

semgroups = SemanticGroups()
stys = umls.terminologies['MSH']['acetaminophen']
print(stys)
print([semgroups.types[sty] for sty in stys])
```

```
{'T109', 'T121'}
['Organic Chemical', 'Pharmacologic Substance']
```

In [ ]:
```
```