

```
In [1]: #from google.colab import drive
#drive.mount('/content/gdrive', force_remount=True)

Mounted at /content/gdrive

In [1]: import pandas as pd
df = pd.read_csv('spct.tsv', sep='\t')

In [2]: df.head(5)

Out[2]:
   id      subreddit      title      selftext
0  6d8knd  talesfromtechsupport  Remember your command line switches...  Hi there, <b>The usual. Long time lanker, fi...
1  58mbft               teenmom          So what was Matt "addicted" to?  Did he ever say what his addiction was or is h...
2  BF7zs?              Harley          No Club Colors          Funny story. I went to college in Las Vegas. T...
3  66bre             mgdoorbell  Not door bell, but floodlight mount height.  I know this is a sub for the 'Ring Doorbell b...
4  77xso                intel  Worried about my 8700k small fltdata stress t...  Prime95 (regardless of version) and OCCT both...
```

```
In [3]: import sqlite3
con = sqlite3.connect('selfposts.db')
pd.read_csv('subreddit_info.csv').to_sql("categories", con)

In [4]: import re
def clean(s):
    s = s.replace(r'<lb>', "\n")
    s = s.replace(r'<tab>', "\t")
    s = re.sub(r'<br />', "\n", s)
    s = s.replace("&lt;", "<").replace("&gt;", ">").replace("&amp;", "&")
    s = s.replace("&#"; "&")
    # markdown urls
    s = re.sub(r'\\(https?:\\/\\(\\s)*', "", s)
    # normal urls
    s = re.sub(r'https?:\\/\\(\\s)*', "", s)
    s = re.sub(r'\\s+', ' ', s)
    s = re.sub(r'^+', '', s)
    return str(s)

In [5]: df['selftext_clean'] = ''
for i, row in df.iterrows():
    df.at[i, 'selftext_clean'] = clean(row.selftext)

In [6]: df.head()

Out[6]:
   id      subreddit      title      selftext      selftext_clean
0  6d8knd  talesfromtechsupport  Remember your command line switches...  Hi there, <b>The usual. Long time lanker, fi...  Hi there, \nThe usual. Long time lanker, firs...
1  58mbft               teenmom          So what was Matt "addicted" to?  Did he ever say what his addiction was or is h...  Did he ever say what his addiction was or is h...
2  BF7zs?              Harley          No Club Colors          Funny story. I went to college in Las Vegas. T...  Funny story. I went to college in Las Vegas. T...
3  66bre             mgdoorbell  Not door bell, but floodlight mount height.  I know this is a sub for the 'Ring Doorbell b...  I know this is a sub for the 'Ring Doorbell b...
4  77xso                intel  Worried about my 8700k small fltdata stress t...  Prime95 (regardless of version) and OCCT both...  Prime95 (regardless of version) and OCCT both...
```

```
In [7]: df.to_sql('posts', con)

In [8]: !pip install -U spacy -q
!python -m spacy download 'en_core_web_sm' -q
import spacy
nlp = spacy.load('en_core_web_sm')

ERROR: After October 2020 you may experience errors when installing or updating packages. This is because pip will change the way that it resolves dependency conflicts.

We recommend you use --use-feature=2020-resolver to test your packages with the new resolver before it becomes the default.

pycaret 2.3.4 requires spacy<2.4.0, but you'll have spacy 3.2.0 which is incompatible.
en-core-web-sm 2.3.1 requires spacy<2.4.0,>=2.3.0, but you'll have spacy 3.2.0 which is incompatible.
You can now load the package via spacy.load('en_core_web_sm')

In [9]: #got rid of data sampling as the DB will be used in all exercises
#re-ran notebook with this commented out so db has full dataset
#display(str(df.shape))
#dfShort = df.sample(20000)
#display(str(dfShort.shape))

In [10]: import time

In [11]: %time
for i, row in df.iterrows():
    if i % 1000 == 0:
        print(i)
    if (row["selftext_clean"] and len(str(row["selftext_clean"])) < 1000000):
        doc = nlp(str(row["selftext_clean"]))
        adjectives = []
        nouns = []
        verbs = []
        lemmas = []

        for token in doc:
            lemmas.append(token.lemma_)
            if token.pos_ == "ADJ":
                adjectives.append(token.lemma_)
            if token.pos_ == "NOUN" or token.pos_ == "PROPN":
                nouns.append(token.lemma_)
            if token.pos_ == "VERB":
                verbs.append(token.lemma_)

        df.at[i, "selftext_lemmas"] = " ".join(lemmas)
        df.at[i, "selftext_nouns"] = " ".join(nouns)
        df.at[i, "selftext_adjectives"] = " ".join(adjectives)
        df.at[i, "selftext_verbs"] = " ".join(verbs)
        df.at[i, "selftext_nav"] = " ".join(nouns+adjectives+verbs)
        df.at[i, "no_tokens"] = len(lemmas)
```

3890
3900
3910
3920
3930
3940
3950
3960
3970
3980
3990
4000
4010
4020
4030
4040
4050
4060
4070
4080
4090
4100
4110
4120
4130
4140
4150
4160
4170
4180
4190
4200
4210
4220
4230
4240
4250
4260
4270
4280
4290
4300
4310
4320
4330
4340
4350
4360
4370
4380
4390
4400
4410
4420
4430
4440
4450
4460
4470
4480
4490
4500
4510
4520
4530
4540
4550
4560
4570
4580
4590
4600
4610
4620
4630
4640
4650
4660
4670
4680
4690
4700
4710
4720
4730
4740
4750
4760
4770
4780
4790
4800
4810
4820
4830
4840
4850
4860
4870
4880
4890
4900
4910
4920
4930
4940
4950
4960
4970
4980
4990
5000
5010
5020
5030
5040
5050
5060
5070
5080
5090
5100
5110
5120
5130
5140
5150
5160
5170
5180
5190
5200
5210
5220
5230
5240
5250
5260
5270
5280
5290
5300
5310
5320
5330
5340
5350
5360
5370
5380
5390
5400
5410
5420
5430
5440
5450
5460
5470
5480
5490
5500
5510
5520
5530
5540
5550
5560
5570
5580
5590
5600
5610
5620
5630
5640
5650
5660
5670
5680
5690
5700
5710
5720
5730
5740
5750
5760
5770
5780
5790
5800
5810
5820
5830
5840
5850
5860
5870
5880
5890
5900
5910
5920
5930
5940
5950
5960
5970
5980
5990
6000
6010
6020
6030
6040
6050
6060
6070
6080
6090
6100
6110
6120
6130
6140
6150
6160
6170
6180
6190
6200
6210
6220
6230
6240
6250
6260
6270
6280
6290
6300
6310
6320
6330
6340
6350
6360
6370
6380
6390
6400
6410
6420
6430
6440
6450
6460
6470
6480
6490
6500
6510
6520
6530
6540
6550
6560
6570
6580
6590
6600
6610
6620
6630
6640
6650
6660
6670
6680
6690
6700
6710
6720
6730
6740
6750
6760
6770
6780
6790
6800
6810
6820
6830
6840
6850
6860
6870
6880
6890
6900
6910
6920
6930
6940
6950
6960
6970
6980
6990
7000
7010
7020
7030
7040
7050
7060
7070
7080
7090
7100
7110
7120
7130
7140
7150
7160
7170
7180
7190
7200
7210
7220
7230
7240
7250
7260
7270
7280
7290
7300
7310
7320
7330
7340
7350
7360
7370
7380
7390
7400
7410
7420
7430
7440
7450
7460
7470
7480
7490
7500
7510
7520
7530
7540
7550
7560
7570
7580
7590
7600
7610
7620
7630
7640
7650
7660
7670
7680
7690
7700
7710
7720
7730
7740
7750
7760
7770
7780
7790
7800
7810
7820
7830
7840
7850
7860
7870
7880
7890
7900
7910
7920
7930
7940
7950
7960
7970
7980
7990
8000
8010
8020
8030
8040
8050
8060
8070
8080
8090
8100
8110
8120
8130
8140
8150
8160
8170
8180
8190
8200
8210
8220
8230
8240
8250
8260
8270
8280
8290
8300
8310
8320
8330
8340
8350
8360
8370
8380
8390
8400
8410
8420
8430
8440
8450
8460
8470
8480
8490
8500
8510
8520
8530
8540
8550
8560
8570
8580
8590
8600
8610
8620
8630
8640
8650
8660
8670
8680
8690
8700
8710
8720
8730
8740
8750
8760
8770
8780
8790
8800
8810
8820
8830
8840
8850
8860
8870
8880
8890
8900
8910
8920
8930
8940
8950
8960
8970
8980
8990
9000
9010
9020
9030
9040
9050
9060
9070
9080
9090
9100
9110
9120
9130
9140
9150
9160
9170
9180
9190
9200
9210
9220
9230
9240
9250
9260
9270
9280
9290
9300
9310
9320
9330
9340
9350
9360
9370
9380
9390
9400
9410
9420
9430
9440
9450
9460
9470
9480
9490
9500
9510
9520
9530
9540
9550
9560
9570
9580
9590
9600
9610
9620
9630
9640
9650
9660
9670
9680
9690
9700
9710
9720
9730
9740
9750
9760
9770
9780
9790
9800
9810
9820
9830
9840
9850
9860
9870
9880
9890
9900
9910
9920
9930
9940
9950
9960
9970
9980
9990
10000

908000
909000
910000
911000
912000
913000
914000
915000
916000
917000
918000
919000
920000
921000
922000
923000
924000
925000
926000
927000
928000
929000
930000
931000
932000
933000
934000
935000
936000
937000
938000
939000
940000
941000
942000
943000
944000
945000
946000
947000
948000
949000
950000
951000
952000
953000
954000
955000
956000
957000
958000
959000
960000
961000
962000
963000
964000
965000
966000
967000
968000
969000
970000
971000
972000
973000
974000
975000
976000
977000
978000
979000
980000
981000
982000
983000
984000
985000
986000
987000
988000
989000
990000
991000
992000
993000
994000
995000
996000
997000
998000
999000
1000000
1001000
1002000
1003000
1004000
1005000
1006000
1007000
1008000
1009000
1010000
1011000
1012000

CPU times: user 3h 38min 24s, sys: 3.41 s, total: 3h 38min 28s
Wall time: 3h 38min 59s

In [12]: df.to_sql('posts_nlp', con)

[illegible]

