Step 1: Setup

Loading the relavent libraries we will use later to analyze the data.

```
!pip install convokit
!pip install sklearn
# spacy setup
!python -m spacy download en_core_web_sm
# nltk setup
import nltk
nltk.download('punkt')

import convokit
from convokit import Corpus, download

import pandas as pd
import numpy as np
from numpy.linalg import norm, multi_dot, inv

import re
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: convokit in /usr/local/lib/python3.9/dist-packages (2.5.3)
Requirement already satisfied: dill>=0.2.9 in /usr/local/lib/python3.9/dist-packages (from convokit) (0.3.6)
Requirement already satisfied: spacy>=2.3.5 in /usr/local/lib/python3.9/dist-packages (from convokit) (3.5.1)
Requirement already satisfied: msgpack-numpy>=0.4.3.2 in /usr/local/lib/python3.9/dist-packages (from convokit) (0.
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.9/dist-packages (from convokit) (3.7.1)
Requirement already satisfied: clean-text>=0.1.1 in /usr/local/lib/python3.9/dist-packages (from convokit) (0.6.0)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.9/dist-packages (from convokit) (1.10.1)
Requirement already satisfied: nltk>=3.4 in /usr/local/lib/python3.9/dist-packages (from convokit) (3.8.1)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.9/dist-packages (from convokit) (1.2.
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.9/dist-packages (from convokit) (1.1.1)
Requirement already satisfied: pandas>=0.23.4 in /usr/local/lib/python3.9/dist-packages (from convokit) (1.4.4)
Requirement already satisfied: unidecode>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from convokit) (1.3.6)
Requirement already satisfied: emoji<2.0.0,>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from clean-text>=0.1.
Requirement already satisfied: ftfy<7.0,>=6.0 in /usr/local/lib/python3.9/dist-packages (from clean-text>=0.1.1->co
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=3.0.0-
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=3.0.0->convo
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=3.0.0->
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=3.0.0->
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=3.0.0-
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=3.0.0->con
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=3.0.0->c
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotli
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=3.0
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (from matplotlib>=3.0.0->conv
Requirement already satisfied: msgpack>=0.5.2 in /usr/local/lib/python3.9/dist-packages (from msgpack-numpy>=0.4.3.
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from nltk>=3.4->convokit) (4.65.0)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from nltk>=3.4->convokit) (8.1.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.9/dist-packages (from nltk>=3.4->convokit)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=0.23.4->convoki
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=0
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5->convokit) (3.1.
Requirement already satisfied: pathy>=0.10.0 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5->convokit
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4 in /usr/local/lib/python3.9/dist-packages (fro
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5->c
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.9/dist-packages (from spacy>=2
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from spacy>=2
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5->
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5->co
Requirement already satisfied: typer<0.8.0,>=0.3.0 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5->co
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5
Requirement already satisfied: thinc<8.2.0,>=8.1.8 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5->co
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5->convokit) (
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.9/dist-packages (from spacy>=2.3.5->co
Requirement already satisfied: wcwidth>=0.2.5 in /usr/local/lib/python3.9/dist-packages (from ftfy<7.0,>=6.0->clean
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (from importlib-resources>=3.2
```

```
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.9/dist-packages (from pydantic!=1
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.7->matpl
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests<3.0.0,>=2.13.0
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests<3.0.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests<3.0.0,>=
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests<3
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.9/dist-packages (from thinc<8.2.0,>=8.1
```

Loading our data

We are using Convokit's corpus's of reddit communities. We include data from a general corpus with 100 subreddits and 2,000,000 posts. We add too this a few more subreddit's that serve as approximations of reigonal groups.

```
corpora = ['reddit-corpus',           # Corpus with 100 Subreddits and a total of 2,000,000+ Posts

               'subreddit-concordia',     # Concordia Speakers
               'subreddit-mcgill',        # McGill Speakers
               'subreddit-AskUK',         # UK Speakers
               'subreddit-Pennsylvania',  # PA Speakers
               'subreddit-AskNYC',        # New York Speakers
               'subreddit-bostonhousing']        # Boston Speakers

corpus = Corpus(filename=download(corpora[0]))

"""
for corpus in corpora[1:]:
  temp = Corpus(filename=download(corpus))
  temp.print_summary_stats()
  corpus = Corpus.merge(corpus, temp)
"""
```

```
Dataset already exists at /root/.convokit/downloads/reddit-corpus
'          \nfor corpus in corpora[1:]:\n  temp = Corpus(filename=download(corpus))\n  temp.print_summary_stats()
\n  corpus = Corpus.merge(corpus, temp)\n'
```

```
corpus.print_summary_stats()
```

```
Number of Speakers: 521777
Number of Utterances: 2004262
Number of Conversations: 84979
```

Step 2: Lexical Variation

Binary Relative Frequencies

For our first task, we look at the frequencies of two different words. The pairs of words analyzed should be words that appear in a complimentary distribution ("going to" v.s. "gonna"). By observing the ratio of their frequencies, we can get an idea of how often each word is used.

```
def binary_lexical_variation(word1, word2, corp, meta='subreddit'):
  word_1 = re.compile(word1)
  word_2 = re.compile(word2)
  word_ratio_dict ={}
  # Sub : [# word1, # word2]
  total = 0
  for utt in corp.iter_utterances():
    sub = utt.meta[meta]
    if type(utt.meta[meta]) is str:
      sub = utt.meta[meta].lower()
    else:
      if utt.meta['score'] < 0:
        sub = -1
      else:
        sub = ((utt.meta['score']) // 10) * 10

    total += 1
```

```
      if sub not in word_ratio_dict:
        word_ratio_dict[sub] = [0,0]
      text = utt.text.lower()
      for word in re.findall(word_1, text):
        word_ratio_dict[sub][0]+=1
      for word in re.findall(word_2, text):
        word_ratio_dict[sub][1]+=1

    # Calculate Similarities
    for sub, values in word_ratio_dict.items():
      if values[1] != 0:
        word_ratio_dict[sub] = values[0]/values[1] # Adding 1 for Div by 0
      else:
        # If word_2 freq is 0, then the w1 ratio should be the maximum
        word_ratio_dict[sub] = -1
    if max(word_ratio_dict.values()) == 0:
      return 0
    # Normalize Similarities against the Max (0-1 Scale)
    """
    for sub, v in word_ratio_dict.items():
      if v != -1:
        word_ratio_dict[sub] = v / max(word_ratio_dict.values())
      else:
        word_ratio_dict[sub] = 1.0
    """
    return word_ratio_dict # word radio dict -> {subreddits: word_1 / word_2}
```

Picking Pairs of Words

- Contractions
  - 10 frequent contractions
  - Apostrophes are optional when parsing text
- British Spelling
  - 10 frequent Amrican-British spelling differences
- Acronyms
  - 5 frequent Acronyms that are used in place of a real word expression
    - We omit acronyms whose purpose isn't to shorten the

```
contractions = [["gonna", "going to"],
                ["i ain'?t", "i'?m not"],
                ["you'?re", "you are"],
                ["don'?t", "do not"],
                ["isn'?t", "is not"],
                ["he'?s", "he is"],
                ["they'?re", "they are"],
                ["can't", "cannot"],
                ["didn'?t", "did not"],
                ["shouldn't", "should not"]]
contractions_set = set([word1 for word1, word2 in contractions])
british =      [['colour', 'color'],
                ['centre', 'center'],
                ['grey', 'gray'],
                ['programme', 'program'],
                ['theatre', 'theater'],
                ['jewellery', 'jewelry'],
                ['labour', 'labor'],
                ['cheque', 'check']]
british_set = set([word1 for word1, word2 in british])
acronyms =     [["brb", "be right back"],
                ["idk", "i (don'?t|do not) know"],
                ["omg", "oh my god"],
                ["btw", "by the way"],
                ["irl", "in real life"]]
acronyms_set = set([word1 for word1, word2 in acronyms])
binary_words = contractions + british + acronyms
```

Reddit Comparison

Code to get different reddit's postings

```
flag = True
for word_1, word_2 in binary_words:
  output = binary_lexical_variation(word_1, word_2, corpus)
  if flag:
    sub_dict = {k: [v]  for k,v in output.items()}
    flag = False
  else:
    print(word_1, word_2)
    print(sub_dict)
    if output == 0:
       binary_words.remove([word_1,word_2])
       continue
    sub_dict = {k: sub_dict[k] + [v] for k,v in output.items()}
binary_df = pd.DataFrame.from_dict(sub_dict)
binary_df.index= [word1 + "/" + word2 for word1,word2 in binary_words]

binary_df
```

```
i ain'?t i'?m not
{'maliciouscompliance': [0.267515923566879], 'techsupport': [0.30303030303030304], 'cringepics': [0.443502824858757
you'?re you are
{'maliciouscompliance': [0.267515923566879, 0.005420054200542005], 'techsupport': [0.30303030303030304, 0.0], 'crin
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-6-f5a29365be13> in <module>
      1 flag = True
      2 for word_1, word_2 in binary_words:
----> 3   output = binary_lexical_variation(word_1, word_2, corpus)
      4   if flag:
      5     sub_dict = {k: [v]  for k,v in output.items()}

                                 1 frames
/usr/local/lib/python3.9/dist-packages/convokit/model/convoKitMeta.py in __getitem__(self, item)
     19
     20     def __getitem__(self, item):
---> 21         return dict.__getitem__(self, item)
     22
     23     @staticmethod

KeyboardInterrupt:

SEARCH STACK OVERFLOW
```

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

binary_df.to_csv("/content/drive/My Drive/binary_variation.csv")
```

McGill Reddit Scores

```
flag = True
none_found = set()
for word_1, word_2 in binary_words:
  output = binary_lexical_variation(word_1, word_2, corpus, meta="score")
  if flag:
    sub_dict = {k: [v]  for k,v in output.items()}
    flag = False
  else:
    print(word_1, word_2)
    print(sub_dict)
    if output == 0:
```

```
        print([word_1,word_2] in binary_words)
        binary_words.remove([word_1,word_2])
        continue
      sub_dict = {k: sub_dict[k] + [v] for k,v in output.items()}

binary_df = pd.DataFrame.from_dict(sub_dict)
binary_df.index= [word1 + "/" + word2 for word1,word2 in binary_words]

binary_df
```

```
i ain'?t i'?m not
{0: [0.5008735804317983], 10: [0.422339028854539], 20: [0.5726429115688042], 40: [0.08731444848067835], 50: [0.5820
you'?re you are
{0: [0.5008735804317983, 0.0036582179252678338], 10: [0.422339028854539, 0.0], 20: [0.5726429115688042, 0.0], 40: [
don'?t do not
{0: [0.5008735804317983, 0.0036582179252678338, 1.0], 10: [0.422339028854539, 0.0, 1.0], 20: [0.5726429115688042, 0
isn'?t is not
{0: [0.5008735804317983, 0.0036582179252678338, 1.0, 0.8964064602960968], 10: [0.422339028854539, 0.0, 1.0, 0.75272
he'?s he is
{0: [0.5008735804317983, 0.0036582179252678338, 1.0, 0.8964064602960968, 1.0], 10: [0.422339028854539, 0.0, 1.0, 0.
they'?re they are
{0: [0.5008735804317983, 0.0036582179252678338, 1.0, 0.8964064602960968, 1.0, 1.0], 10: [0.422339028854539, 0.0, 1.
can't cannot
{0: [0.5008735804317983, 0.0036582179252678338, 1.0, 0.8964064602960968, 1.0, 1.0, 0.7501311418080084], 10: [0.4223
didn'?t did not
{0: [0.5008735804317983, 0.0036582179252678338, 1.0, 0.8964064602960968, 1.0, 1.0, 0.7501311418080084, 0.6242118313
shouldn't should not
{0: [0.5008735804317983, 0.0036582179252678338, 1.0, 0.8964064602960968, 1.0, 1.0, 0.7501311418080084, 0.6242118313
colour color
{0: [0.5008735804317983, 0.0036582179252678338, 1.0, 0.8964064602960968, 1.0, 1.0, 0.7501311418080084, 0.6242118313
centre center
{0: [0.5008735804317983, 0.0036582179252678338, 1.0, 0.8964064602960968, 1.0, 1.0, 0.7501311418080084, 0.6242118313
grey gray
```

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

binary_df.to_csv("/content/drive/My Drive/mcgillcontractions.csv")
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force
cheque check
```

Reigonal Lexical Variation

We picked a few questions from the Data Collection assignment that saw the most variation or were the most clearly attributable to an idenity.

```
bite by the way
```

```
entry1 = ["silverware", "utensils", "cutlery"]
entry2 = ["foot[- ]long", "grinder", "hero", "hoagie", "sub"]
#entry3 = ["water fountain", "drinking fountain"]
entry4 = ["coke",  "cold", "drink", "fizzy drink", "[^\w]pop[^\w]", "soda", "soft drink", "tonic"]
entry5 = ["all[- ]dressed", "deluxe", "everything[- ]on[- ]it", "loaded", "supreme", "the works"]
entry6 = ["elementary school", "grade school", "grammar school", "primary school", "public school"]
entry7 = ["bathroom", "restroom"]
entry8 = ["corner shop", "corner store", "deli", "([^\w]dep[^\w]|[^\w]depanneur[^\w])", "variety store"]
entry9 = ["chesterfield", "couch", "davenport", "divan", "settee", "sofa"]

# Double Negatives

reigonal_variation = [entry1, entry2, entry4, entry5, entry6, entry7, entry8, entry9]
all_words = []
for line in reigonal_variation:
  for word in line:
    all_words.append(word)
all_words
```

```
['silverware',
 'utensils',
 'cutlery',
 'foot[- ]long',
 'grinder',
 'hero',
 'hoagie',
 'sub',
 'coke',
 'cold',
 'drink',
 'fizzy drink',
 '[^\\w]pop[^\\w]',
 'soda',
 'soft drink',
 'tonic',
```

```
        'all[- ]dressed',
        'deluxe',
        'everything[- ]on[- ]it',
        'loaded',
        'supreme',
        'the works',
        'elementary school',
        'grade school',
        'grammar school',
        'primary school',
        'public school',
        'bathroom',
        'restroom',
        'corner shop',
        'corner store',
        'deli',
        '([^\\w]dep[^\\w]|[^\\w]depanneur[^\\w])',
        'variety store',
        'chesterfield',
        'couch',
        'davenport',
        'divan',
        'settee',
        'sofa']
```

Double-click (or enter) to edit

```
flag = True
subreddits = {}

print(all_words)
for words in reigonal_variation:
  regex_list = [re.compile(word) for word in words]
  for utt in corpus.iter_utterances():
    sub = utt.meta['subreddit'].lower()
    if sub in subreddits:
      if words[0] not in subreddits[sub]:
        subreddits[sub].update({word: 0 for word in words})
    else:
      subreddits[sub] = {word: 0 for word in words}
    text = utt.text.lower() # our version of preprocessing
    for i, reg in enumerate(regex_list):
      if re.search(reg, text): # If there was a regex match of the word
        subreddits[sub][words[i]] += 1
  max_freq = {k: 0.0 for k in words}
  min_freq = {k: np.inf for k in words}

  for sub, values in subreddits.items():
      word_sub_values = [subreddits[sub][word] for word in words]
      if sum(word_sub_values) == 0:
        continue
      for word in words:
        value = subreddits[sub][word]
        subreddits[sub][word] = value/sum(word_sub_values) # w freq over total freq
        if value > max_freq[word]:
          max_freq[word] = value
        if value < min_freq[word]:
          min_freq[word] = value
  print(subreddits)

subreddits
```

```
          'couch': 0.75,
          'davenport': 0.0,
          'divan': 0.0,
          'settee': 0.0,
          'sofa': 0.125},
         'apple': {'silverware': 0.0,
          'utensils': 0.0,
          'cutlery': 1.0,
          'foot[- ]long': 0.0,
          'grinder': 0.0,
          'hero': 0.020642201834862386,
          'hoagie': 0.0,
          'sub': 0.9793577981651376,
          'coke': 0.02247191011235955,
          'cold': 0.23595505617977527,
          'drink': 0.06741573033707865,
          'fizzy drink': 0.0,
          '[^\\w]pop[^\\w]': 0.6741573033707865,
          'soda': 0.0,
          'soft drink': 0.0,
          'tonic': 0.0,
          'all[- ]dressed': 0.0,
          'deluxe': 0.0,
          'everything[- ]on[- ]it': 0.0,
          'loaded': 0.9113924050632911,
          'supreme': 0.0,
          'the works': 0.08860759493670886,
          'elementary school': 0.0,
          'grade school': 0.0,
          'grammar school': 0.0,
          'primary school': 0.25,
          'public school': 0.75,
          'bathroom': 1.0,
          'restroom': 0.0,
          'corner shop': 0.0,
          'corner store': 0.0,
          'deli': 0.994413407821229,
          '([^\\w]dep[^\\w]|[^\\w]depanneur[^\\w])': 0.00558659217877095,
          'variety store': 0.0,
          'chesterfield': 0.0,
          'couch': 1.0,
          'davenport': 0.0,
          'divan': 0.0,
          'settee': 0.0,
          'sofa': 0.0},
         'politics': {'silverware': 1.0,
          'utensils': 0.0,
          'cutlery': 0.0,
          'foot[- ]long': 0.0,
```

```
complex_dict = pd.DataFrame.from_dict(subreddits)
concat_df = pd.concat([complex_dict, binary_df])
norm_df = concat_df.div(concat_df.sum(axis=1), axis=0)
norm_df
```

	maliciouscompliance	techsupport	cringepics	lgbt	minecraft	no
silverware	0.057340	0.000000	0.000000	0.000000	0.000000	0.000
utensils	0.006693	0.000000	0.040155	0.040155	0.000000	0.000
cutlery	0.012373	0.000000	0.000000	0.000000	0.000000	0.000
foot[-]long	0.098671	0.075908	0.000000	0.000000	0.000000	0.000

```
dataf=((concat_df-concat_df.min())/(concat_df.max()-concat_df.min()))
dataf
```

	maliciouscompliance	techsupport	cringepics	lgbt	minecraft	nofap	conspiracy	guns	frugal	
silverware	0.046729	0.075099	0.025694	0.018293	0.048154	0.024007	0.057471	0.054422	0.001623	0.0
utensils	0.032710	0.075099	0.051387	0.036585	0.048154	0.024007	0.114943	0.054422	0.012987	0.0
cutlery	0.032710	0.075099	0.025694	0.018293	0.048154	0.024007	0.057471	0.054422	0.003247	0.0
foot[-]long	0.028260	0.075557	0.025694	0.018293	0.048154	0.024007	0.057471	0.054816	0.000069	0.0
grinder	0.028556	0.075099	0.025745	0.018330	0.052040	0.024053	0.057471	0.055210	0.000483	0.0
...	...	...	...	...	...	...	...	...	...	
brb/be right back	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.017857	0.3
idk/i (don'?tldo not) know	0.036627	0.088990	0.040652	0.028171	0.067340	0.039441	0.076859	0.069221	0.006033	0.0
omg/oh my god	0.107477	0.250329	0.083505	0.065331	0.112360	0.066687	0.179598	0.163265	0.043367	0.1
btw/by the way	0.079685	0.240316	0.061379	0.047797	0.172788	0.089805	0.190097	0.212740	0.085714	0.1

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

norm_df.to_csv("/content/drive/My Drive/norm_df.csv")
dataf.to_csv("/content/drive/My Drive/dataf.csv")
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force
```

Subreddit Similarity

From Source: "The data is extracted from publicly available Reddit data of 2.5 years from Jan 2014 to April 2017.

This file generates one numerical vector in low dimensional space (a.k.a. embeddings) for each subreddit. The embeddings are 300 dimensions each. Two subreddit embeddings are similar if the users who post in them are similar."

TLDR: *Embeddings were created by looking at user overlap, and by taking the cosine similarity of two reddit's embeddings, we get a score of their similarity*

Academic Paper: (https://cs.stanford.edu/~srijan/pubs/conflict-paper-www18.pdf)

Data Source: (https://snap.stanford.edu/data/web-RedditEmbeddings.html)

```
from google.colab import drive
drive.mount('/content/drive/')
reddit_emeddings = pd.read_csv("/content/web-redditEmbeddings-subreddits.csv",index_col=0, header=None)
binary_df = pd.read_csv("/content/drive/My Drive/binarylexicalvariation2.csv",index_col=0, header=None)
binary_df
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force
```

0	1	2	3	4	
NaN	maliciouscompliance	techsupport	cringepics	lgbt	minecra
gonna/going to	0.2692418327511814	0.30498533724340177	0.44636413340623293	0.28672168042010504	0.3016277005031074
i'?m not/i ain'?t	0.16414590747330962	1.0	0.07829181494661921	0.1631079478054567	0.2126334519572953
you'?re/you are	0.568854803367547	0.5991945887521356	0.7630496678266372	0.49600263320116594	0.539566808774383
don'?t/do not	0.26793995396928993	0.22739640830288427	0.43089382020542066	0.2275444889920518	0.364967996061053
isn'?t/is not	0.4406119304424389	0.5450542902542374	0.5814887427270428	0.4057122525553112	0.442548608355535
he'?s/he is	0.15248890105211946	0.039163134712872937	0.10673485356583949	0.11654615834063689	0.281150793650793
they'?re/they are	0.710977836354183	0.4468822170900693	0.8168226571046554	0.5078074318187715	0.754642032332563
can't/cannot	0.2885375494071146	0.21437795953508393	0.5010351966873706	0.2465217391304348	0.2149980055843637
didn'?t/did not	0.3126344086021505	0.20218082689686506	0.37688172043010754	0.22376480195998366	0.2662871600253004
shouldn't/should not	0.23107890499194847	0.345679012345679	0.4066811909949165	0.40993985438429886	0.3731762065095398
colour/color	0.05263157894736842	0.031007751937984496	0.05913978494623656	0.03551912568306011	0.03972366148531951
centre/center	0.02223719676549865	0.03308270676691729	0.018487394957983194	0.016923076923076926	0.02668463611859838
grey/gray	0.06516290726817042	0.045112781954887216	0.0886143931256713	0.06390977443609022	0.1071428571428571
catalogue/catalog	0.6666666666666666	0.0	1.0	0.0	1.
programme/program	0.38253119429590016	0.03357024043550582	0.3109243697478991	0.0	0.567201426024955
theatre/theater	0.14054054054054055	0.16216216216216214	0.08108108108108107	0.14864864864864863	1.
traveller/traveler	0.0	1.0	1.0	0.0	0.
jewellery/jewelry	0.06666666666666667	1.0	0.0	0.0	1.
labour/labor	0.006717245069561229	0.00730647709320695	0.01733164333737463	0.008767772511848342	0.
cheque/check	0.10642479213907785	0.0	0.0	0.0	0.
brb/be right back	1.0	1.0	1.0	1.0	1.
idk/I don't know	1.0	1.0	1.0	1.0	1.
omg/oh my god	0.14912280701754388	0.12280701754385966	0.11842105263157894	0.13533834586466167	0.0701754385964912
btw/by the way	0.13419309864064133	0.16026490066225169	0.10117733627667402	0.1174962614825892	0.188546941955590
irl/in real life	0.20758483033932135	1.0	0.22706586826347305	0.3213572854291417	0.02337260961947073

```
newheaders = binary_df.iloc[0]
binary_df = binary_df[1:]
binary_df.columns =newheaders
binary_df
```

	nan	maliciouscompliance	techsupport	cringepics	lgbt	minecra
	0					
gonna/going to	0.2692418327511814	0.30498533724340177	0.44636413340623293	0.28672168042010504	0.301627700503107	
i'?m not/i ain'?t	0.16414590747330962	1.0	0.07829181494661921	0.1631079478054567	0.212633451957295	
you'?re/you are	0.568854803367547	0.5991945887521356	0.7630496678266372	0.49600263320116594	0.53956680877438	
don'?t/do not	0.26793995396928993	0.22739640830288427	0.43089382020542066	0.2275444889920518	0.36496799606105	
isn'?t/is not	0.4406119304424389	0.5450542902542374	0.5814887427270428	0.4057122525553112	0.44254860835553	
he'?s/he is	0.15248890105211946	0.039163134712872937	0.10673485356583949	0.11654615834063689	0.28115079365079	
they'?re/they are	0.710977836354183	0.4468822170900693	0.8168226571046554	0.5078074318187715	0.75464203233256	
can't/cannot	0.2885375494071146	0.21437795953508393	0.5010351966873706	0.2465217391304348	0.214998005584363	
didn'?t/did not	0.3126344086021505	0.20218082689686506	0.37688172043010754	0.22376480195998366	0.266287160025300	
shouldn't/should not	0.23107890499194847	0.345679012345679	0.4066811909949165	0.40993985438429886	0.373176206509539	
colour/color	0.05263157894736842	0.031007751937984496	0.05913978494623656	0.03551912568306011	0.0397236614853195	
centre/center	0.02223719676549865	0.03308270676691729	0.018487394957983194	0.016923076923076926	0.0266846361185983	
grey/gray	0.06516290726817042	0.045112781954887216	0.0886143931256713	0.06390977443609022	0.107142857142857	
catalogue/catalog	0.6666666666666666	0.0	1.0	0.0		
programme/program	0.38253119429590016	0.03357024043550582	0.3109243697478991	0.0	0.56720142602495	
theatre/theater	0.14054054054054055	0.16216216216216214	0.08108108108108107	0.14864864864864863		
traveller/traveler	0.0	1.0	1.0	0.0		

```
import numpy as np
from numpy.linalg import norm, multi_dot, inv
def get_overlaps(U, Sigma, VT, subreddit, k, df):
    #    U     SIGMA     VT
    # (len(df.index), 30), (30,), (30, 300))
    doc_index = df.index.get_loc(subreddit)
    Sigma = np.diag(Sigma)
    compareDocLow = Sigma.dot(U[doc_index])
    Norm1 = norm(compareDocLow)

    docDict = {}
    print(subreddit)
    for i, doc in enumerate(U):
        if i == doc_index:
            continue
        docLow = Sigma.dot(doc)
        Norm2 = norm(docLow)
        docDict[i] = np.dot(compareDocLow,docLow) / (Norm1 * Norm2)
    docDict = {df.index[k]: v for k,v in sorted(docDict.items(), key=lambda item: item[1], reverse=True)[:k]}


    return docDict



from sklearn.utils.extmath import randomized_svd
norm_df = binary_df
subreddits = list(norm_df.columns)
reddit_emeddings = reddit_emeddings[reddit_emeddings.index.isin(subreddits)]
U, Sigma, VT = randomized_svd(reddit_emeddings, n_components=15, n_iter=100, random_state=42)
U.shape, Sigma.shape, VT.shape
```

```
((100, 15), (15,), (15, 300))
```

```
overlaps = {}
for subreddit in reddit_emeddings.index:
```

```
    overlaps1 = get_overlaps(U, Sigma, VT, subreddit, 5, reddit_emeddings)
    overlaps[subreddit] = overlaps1
overlaps
     funny
     the_donald
     videos
     news
     leagueoflegends
     pics
     aww
     worldnews
     music
     politics
     dota2
     trees
     explainlikeimfive
     buildapc
     movies
     business
     technology
     askscience
     squaredcircle
     nba
     politic
     wtf
     gifs
     soccer
     techsupport
     wow
     2007scape
     conspiracy
     relationships
     pokemontrades
     fitness
     electronic_cigarette
     teenagers
     nfl
     nofap
     pokemon
     bitcoin
     anime
     android
     games
     mma
     hockey
     makeupaddiction
     atheism
     minecraft
     apple
     drugs
     sex
     science
     books
     magictcg
     canada
     cfb
     australia
     tifu
     askmen
     unitedkingdom
     conservative


def linguistic_sim(subreddit, l,gg):

  subDict = {}
  Norm1 = norm(list(gg[subreddit]))

  for i, sub in enumerate(gg.columns):
    if sub == subreddit:
      continue
    Norm2 = norm(list(gg[sub]))
    norms = (Norm1 * Norm2)
    subDict[sub] = np.dot(Norm1,Norm2) / (Norm1*Norm2)
  subDict = {k: v for k,v in sorted(subDict.items(), key=lambda item: item[1], reverse=True) }
  return subDict
```

```
def graphData(df):
  normDict = []
  for i, sub in enumerate(df.columns):
    # norm of the subreddit higher more contracted
    normDict.append([sub, overlaps[sub], np.average([int(100*float(df.loc["colour/color",sub])), int(100*float(df.loc["l
  nd = pd.DataFrame(normDict, columns=["subreddit", "overlaps", "british"])
  return nd

nd = graphData(norm_df)
```

```
index = nd.index
nd = nd.sort_values(by='british', ascending=False)
nd.index = index
nd
```

	subreddit	overlaps	british
0	unitedkingdom	{'toronto': 0.8634963447575094, 'soccer': 0.77...	100.000000
1	australia	{'canada': 0.7355492698353323, 'worldnews': 0....	51.666667
2	singapore	{'python': 0.7853330904588672, 'philosophy': 0...	38.333333
3	toronto	{'unitedkingdom': 0.8634963447575094, 'canada'...	36.333333
4	canadapolitics	{'canada': 0.9015161690633853, 'socialism': 0....	24.666667
...	...	...	...
95	conservative	{'mensrights': 0.9361281690530877,	0.000000

```
import ast
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.colors import LogNorm
# Read in the dataframe
```

```
# create graph object
G = nx.Graph()
color_map = []
color_map2 = []
# add edges between nodes
for index, row in nd.iterrows():
    subreddit = row['subreddit']
    overlap = row['overlaps']
    #sim = row['linguistic similarity']
    # GRAPHS SIM REDDIT

    for overlap, weight in overlaps[subreddit].items():
      G.add_edge(subreddit, overlap, weight=float(weight))
   #for overlap, weight in sim.items():
     # G2.add_edge(subreddit, overlap, weight=float(weight))
    color_map.append(round(float(row['british']),2))
    #color_map2.append(float(row['gonna']))
    """
    #GRAPH'S CONTRACTION SIM
    for overlap, cont_weight in contraction.items():
        G.add_edge(subreddit, overlap, weight=float(weight))
    """
```

```
print(color_map)
# Draw the graph

fig, ax = plt.subplots(figsize=(75, 75))
pos = nx.spring_layout(G,iterations=200)

num_edges = dict(G.degree())
num_labels = dict(G.nodes())

cmap = plt.cm.RdBu


#nx.draw_networkx_nodes(G, pos, node_size=5000,ax=ax, node_shape="o", node_color="black")
nx.draw_networkx_nodes(G, pos, node_size=5000,ax=ax, node_shape="o", cmap=cmap, node_color=range(100))
#nx.draw_networkx_nodes(G2, pos2, node_size=2500,ax=ax, node_shape="o", cmap=cmap, node_color=cmap_values)
#nx.draw_networkx_nodes(G, pos, node_size=3500,ax=ax, node_shape="o", node_color=list(pol_color.values()), cmap=cmap2)
nx.draw_networkx_edges(G, pos, edge_color='grey')
nx.draw_networkx_labels(G, pos, font_size=18, font_family='sans-serif', font_weight="bold",bbox=dict(facecolor='white')
#nx.draw_networkx_labels(G2, pos2, font_size=18, font_family='sans-serif', font_weight="bold",bbox=dict(facecolor='blue'
#edge_labels = {(u, v): float(d.values()) for u, v, d in G.edges(data=True)}
#nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=8)



plt.axis('off')
plt.show()
```

Data Validation Assignment Full

Computing the relative frequencies of words from the questionaire used in the Data Validation assignment.

```
answers_list = [['facuet', 'tap', 'spigot', 'faucet'],
                ['water fountain', 'drinking fountain', 'bubbler'],
                ['bucket', 'pail'],
```

```
['gutters', 'eavestroughs'],
['soda', 'pop', 'soft drink', 'fizzy drink', 'coke', 'cold drink'],
['everything-on-it',
'supreme',
'the works',
'all-dressed',
'deluxe',],
['foot-long', 'sub', 'hoagie', 'hero', 'grinder'],
['frosting', 'icing'],
['dinner', 'supper', 'tea'],
['utensils', 'silverware', 'cutlery'],
['washcloth', 'face cloth', 'face flannel'],
['sneakers',
'tennis shoes',
'running shoes',
'trainers',
'runners',
'gym shoes'],
['dressing gown', 'robe', 'bathrobe', 'housecoat'],
['romper', 'jumpsuit', 'playsuit', 'onesie'],
['dresser', 'bureau', 'chest of drawers'],
['couch', 'sofa', 'divan', 'settee'],
['see[- ]saw', 'teeter[- ]totter'],
['elementary school', 'grade school', 'primary school', 'grammar school'],
['first grade', 'grade one', 'year one', 'first form'],
['grade', 'mark'],
['backpack', 'schoolbag', 'rucksack', 'bookbag', 'satchel', 'knapsack'],
['notebook', 'excercise book', 'copybook'],
['toiletry bag',
'toiletry kit',
'toilet bag',
'wash bag',
'toilet case',
'toilet kit',
'dopp kit'],
['purse', 'pocketbook', 'handbag', 'bag'],
['wallet', 'billfold'],
['cottage', 'cabin', 'lake house', 'summer house', 'camp', 'chalet'],
['living room',
'family room',
'sitting room',
'front room',
'lounge',
'parlor'],
['studio',
'bed-sit',
'bachelor',
'efficiency',
#'(one|two|three|1|2|3)&&([- ]and[- ]a[- ]half',
'loft'],
['bathroom',
'restroom',
'lavaroty',
"(men'?s|women'?s) room",
'toilet',
'washrrom',
'the loo',
"gent'?s|ladie'?s",
'powder room',
'[^\w]wc[^\w]'],
['convenience store',
'corner store',
'deli',
'bodega',
'corner shop',
'd[ée]panneur|dep'],
['(parking)?garage', 'car park', 'parkade', 'parking deck'],
['(the|a) check-out',
'(the|a) cashier',
'(the|a) counter',
'(the|a) register',
```

```
                '(the|a) till',
                '(the|a) cash'],
                ['cashpoint', 'hole[- ]in[- ]the[- ]wall', 'cash[- ]machine', 'bank[- ]machine']]
```

```
flag = True
subreddits = {}
word_counts = {}
for i, words in enumerate(answers_list):
  print(words)
  regex_list = [re.compile(word) for word in words]
  for utt in corpus.iter_utterances():
    sub = utt.meta['subreddit'].lower()
    if sub in subreddits:
      if words[0] not in subreddits[sub]:
        subreddits[sub].update({word: 0.0 for word in words})
    else:
      subreddits[sub] = {word: 0.0 for word in words}
    text = utt.text.lower() # our version of preprocessing
    for i, reg in enumerate(regex_list):
      if re.search(reg, text): # If there was a regex match of the word
        subreddits[sub][words[i]] += 1
  max_freq = {k: 0.0 for k in words}
  min_freq = {k: np.inf for k in words}

  for sub, values in subreddits.items():
      word_sub_values = [subreddits[sub][word] for word in words]
      if sum(word_sub_values) == 0:
        continue
      for word in words:
        value = subreddits[sub][word]
        word_counts[word] = value
        subreddits[sub][word] = value/sum(word_sub_values) # w freq over total freq
        if value > max_freq[word]:
          max_freq[word] = value
        if value < min_freq[word]:
          min_freq[word] = value
```

```
['facuet', 'tap', 'spigot', 'faucet']
['water fountain', 'drinking fountain', 'bubbler']
['bucket', 'pail']
['gutters', 'eavestroughs']
['soda', 'pop', 'soft drink', 'fizzy drink', 'coke', 'cold drink']
['everything-on-it', 'supreme', 'the works', 'all-dressed', 'deluxe']
['foot-long', 'sub', 'hoagie', 'hero', 'grinder']
['frosting', 'icing']
['dinner', 'supper', 'tea']
['utensils', 'silverware', 'cutlery']
['washcloth', 'face cloth', 'face flannel']
['sneakers', 'tennis shoes', 'running shoes', 'trainers', 'runners', 'gym shoes']
['dressing gown', 'robe', 'bathrobe', 'housecoat']
['romper', 'jumpsuit', 'playsuit', 'onesie']
['dresser', 'bureau', 'chest of drawers']
['couch', 'sofa', 'divan', 'settee']
['see[- ]saw', 'teeter[- ]totter']
['elementary school', 'grade school', 'primary school', 'grammar school']
['first grade', 'grade one', 'year one', 'first form']
['grade', 'mark']
['backpack', 'schoolbag', 'rucksack', 'bookbag', 'satchel', 'knapsack']
['notebook', 'excercise book', 'copybook']
['toiletry bag', 'toiletry kit', 'toilet bag', 'wash bag', 'toilet case', 'toilet kit', 'dopp kit']
['purse', 'pocketbook', 'handbag', 'bag']
['wallet', 'billfold']
['cottage', 'cabin', 'lake house', 'summer house', 'camp', 'chalet']
['living room', 'family room', 'sitting room', 'front room', 'lounge', 'parlor']
['studio', 'bed-sit', 'bachelor', 'efficiency', 'loft']
['bathroom', 'restroom', 'lavaroty', "(men'?s|women'?s) room", 'toilet', 'washrrom', 'the loo', "gent'?s|ladie'?s",
['convenience store', 'corner store', 'deli', 'bodega', 'corner shop', 'd[ée]panneur|dep']
['(parking)?garage', 'car park', 'parkade', 'parking deck']
['(the|a) check-out', '(the|a) cashier', '(the|a) counter', '(the|a) register', '(the|a) till', '(the|a) cash']
['cashpoint', 'hole[- ]in[- ]the[- ]wall', 'cash[- ]machine', 'bank[- ]machine']
```

```
#subreddits["word counts"] = word_counts
#all_reigonal = pd.DataFrame.from_dict(subreddits)
#norm_all_reigonal = all_reigonal.div(all_reigonal.sum(axis=1), axis=0)
#word_counts


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

all_reigonal.to_csv("/content/drive/My Drive/all_reigonal.csv")
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force
```