soc128d_example_notebook_6_21

June 21, 2021

0.0.1 Sociology 128D: Mining Culture Through Text Data: Introduction to Social Data Science

Sheridan Stewart - last updated June 21, 2021

0.1 Draft Assignment - Not for Circulation

Most class sessions will involve a notebook not unlike this one. A subset of notebooks will include graded content that students are required to complete. This notebook is intended as an example of what one of these graded notebooks might look like. The graded content (Exercises 1-6) can be found at the bottom. In these exercises, students will come up with a series of hypotheses, apply text analysis tools, create visualizations, and reflect on whether their results are consistent with their hypothesis and whether there may be confounding variables.

Each notebook will be used to follow along with lecture material explaining the use of these tools, various considerations when using them, and how dictionary methods have been used in published social research. The notebook will also factor into discussions of the assigned readings, which in any given week will span research published by social scientists and computer scientists.

0.2 Sentiment Analysis and Dictionary Methods using `empath`

In this notebook, we will learn to use dictionary methods for sentiment analysis and more general exploration of topics in social media posts using Python's `empath` library. You can read more about this library in the original paper here. Researchers more commonly use paid sets of dictionaries (or "lexicons") like those available from Linguistic Inquiry and Word Count (LIWC); however, the original paper behind `empath` demonstrates high correlations between seemingly similar categoriess. `empath` offers 194 lexicons in total–and it's free! Another benefit is the ease of creating custom dictionaries.

Dictionary methods are among the most straightforward of methods for computational text analysis. A "dictionary" or "lexicon" in this sense is a list of words that we have decided in advance are related to a given topic, or "lexical category." To use these dictionaries, we count the words in a given document, such as a post on a social media site, that are in that dictionary. If we have a list of positive words and a list of negative words, and more words from the first list are in a document, we might say the document is positive. Sentiment analysis gets much more sophisticated, but that's where we'll start.

We will consider a few ways to use dictionary methods to analyze social processes in a corpus of posts to Reddit's r/offmychest message board. Specifically, we will consider the following:

how and when people interact with social media (time of day, day of week) and how that relates to the content they post

the relationship between content and the success of a post (the number of upvotes, or "karma")

relationships between different types of content (specifically, lexical categories)

More generally, we will consider how these considerations relate to **social processes** and **social institutions**. For example, do users submit content to r/offmychest less frequently during standard working hours? The standard work day and standard work week exert considerable influence on the organization of social life. Indeed, Young and Lim (2014) find that both working and unemployed individuals exhibit similar changes in subjective wellbeing throughout the course of the week, with a likely explanation being that the working friends, family, and acquaintanences of people not currently working tend to be more available outside of standard working hours. A spike in wellbeing on the weekend corresponds to both increased availability of people working standard hours and cultural norms about appropriate behavior on weekends (versus weekdays).

We might also ask questions like whether users writing about a certain topic also tend to write about another topic or set of topics, and we can come up with hypotheses for why we might observe certain associations.

First, we will import the libraries we need for the analyses we'll run. `pyplot` is a standard but powerful library for visualizations. `seaborn` builds on `pyplot`'s functionality. `numpy` and `pandas` are powerful libraries for scientific computing, while `statsmodels` is a library for statistical analyses that may be more familiar to students with a background in statistics or using software like R or Stata.

[110]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
import warnings

from empath import Empath
from pingouin import rcorr

warnings.filterwarnings("ignore")

%matplotlib inline

sns.set_theme(style="darkgrid")
```

Next, we will load the corpus we will examine. This corpus comprises a subset of posts submitted to Reddit's r/offmychest message board in April, 2020. Of note, r/offmychest elected to create weekly COVID-19-related "Megathreads" where people would reply with COVID-19 related content. As a result, few of the submissions in this corpus primarily address COVID–though, as we will see, documents in this corpus address related themes like the idea of an "essential worker" and concerns that essential workers will be forgotten after the pandemic.

r/offmychest describes itself as "A Safe Community for Support" and provides the following description:

A mutually supportive community where deeply emotional things you can't tell people you know can be told. Whether it's long-standing baggage, happy thoughts, or recent trauma, posting it here may provide some relief. We'll listen, and if you want, we'll talk. We aim to keep this a safe space.

I have cleaned this corpus already. It includes 9,352 posts submitted to r/offmychest between April 1 and April 27,2020. The variables include the submission's score ("karma" or "upvotes"), the title of the submission, the text of the submission, and preprocessed text. To preprocess the text, I combined the title and text of the post and went through a sequence of fairly standard steps: lemmatizing words, lowercasing, removing stop words, and removing punctuation, numbers, and special characters. (Note: we discuss these preprocessing decisions and their effects elsewhere.) I have also included the date ("date") and separate variables for the day (1-27), hour (0-23), and minute (1-59). Finally, although Reddit allows for relative anonymity through the use of usernames/handles, I have provided an ID field ("user_id") in place of the username of the author.

```
[2]: df = pd.read_json("sample_assignment_df.json")
```

Let's take a quick look at the data. We'll get to an example post and some analyses of the content, but it's important to first have a sense of what the dataset is like overall.

```
[3]: df.head()
```

```
[3]:                  date  score  \
1 2020-04-01 00:01:27      2
2 2020-04-01 00:01:32      1
3 2020-04-01 00:02:37     11
5 2020-04-01 00:09:57      1
9 2020-04-01 00:21:43      2


                                                title  \
1                            i am so unhappy with my life
2   My SO takes the longest in the shower and it p…
3                            I told someone they're cute.
5                           I see this as an absolute win
9   How many people are considering major life cha…


                                             selftext  \
1   this rant isn't really going to be about anyth…
2   Always in there for an hour and a half or two…
3   Well, I asked her coworker to tell her for me,…
5   Kinda in this weird part of my life where thin…
9   Being trapped in a home I hate in a situation …


                                                 text  day  hour  minute  \
1   unhappy life rant go specific pretty messy try…    1     0       1
2   take long shower piss hour half min shower hai…    1     0       1
3   tell cute ask coworker tell bit weird ik atlea…    1     0       2
5   absolute win kinda weird life thing finally co…    1     0       9
```

```
9  people consider major life change current situ…    1     0      21

   dayofweek  user_id   subreddit
1  Wednesday      856  offmychest
2  Wednesday     7846  offmychest
3  Wednesday     1412  offmychest
5  Wednesday     3596  offmychest
9  Wednesday     6680  offmychest
```

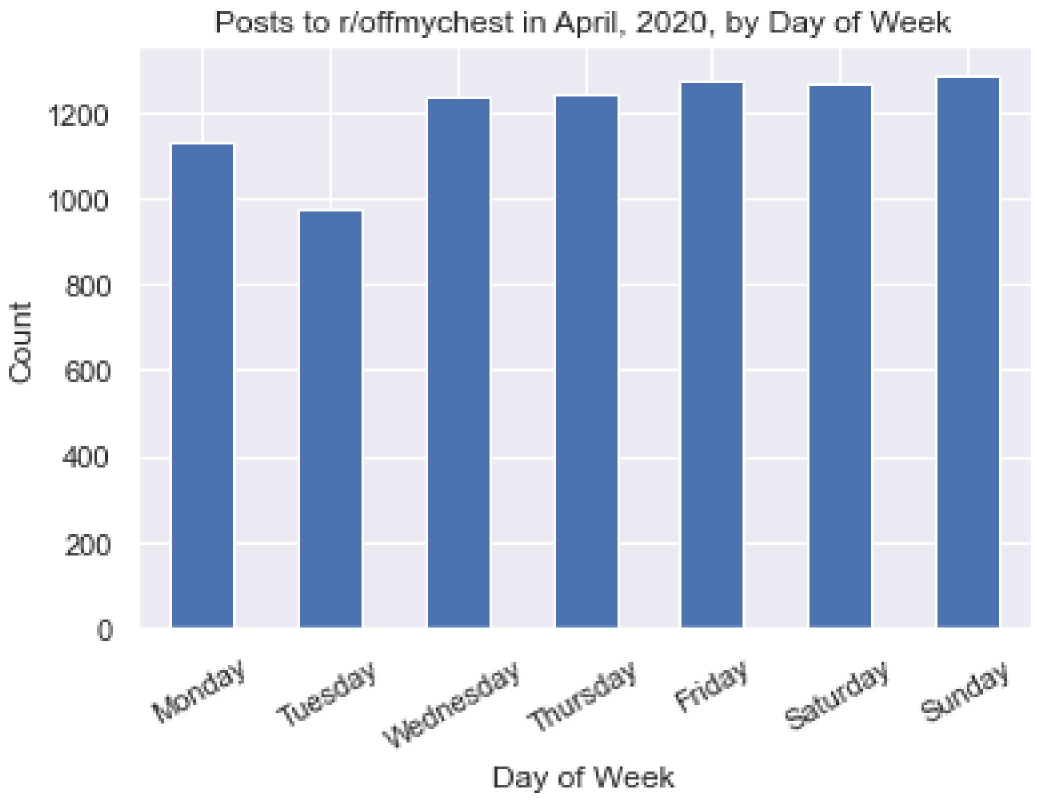
.shape provides the number of rows (posts) first, followed by the number of columns (variables)

```
[4]: df.shape
```

```
[4]: (9352, 11)
```

First, let's examine whether there are substantial differences in the number of posts to r/offmychest between days of the week.

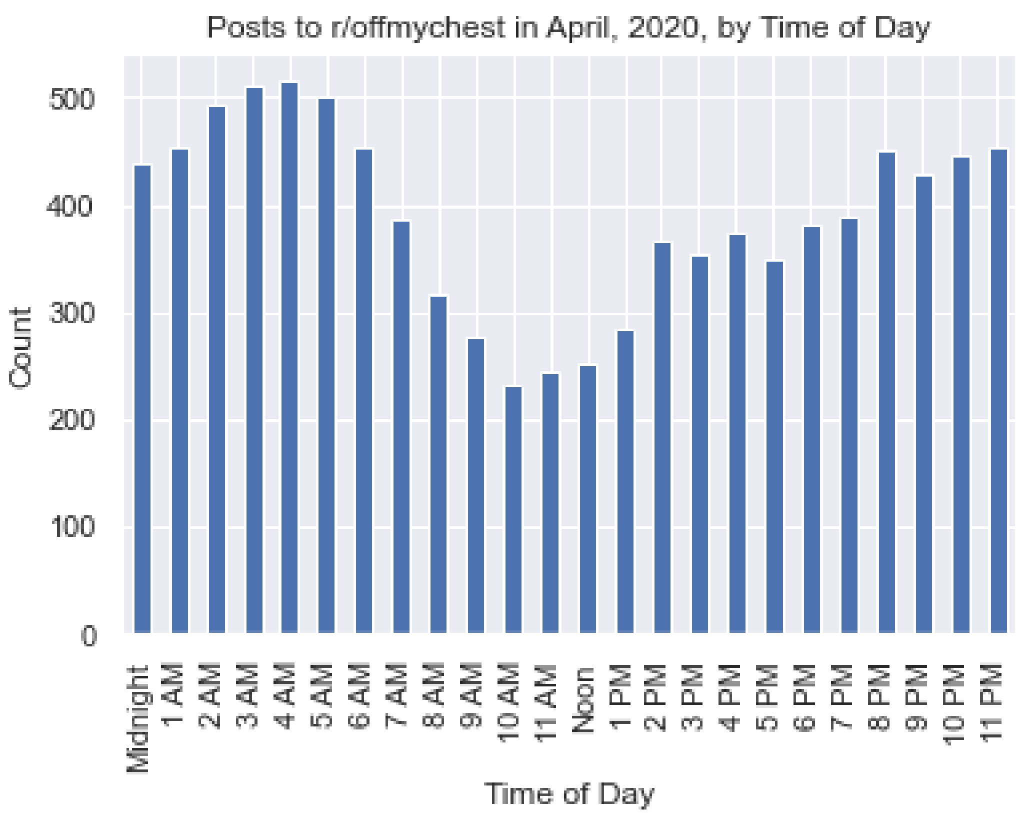
```
[66]: daysofweek = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",␣
       ↪"Saturday", "Sunday"]

      df["dayofweek"].value_counts().loc[daysofweek].plot(kind="bar")
      plt.title("Posts to r/offmychest in April, 2020, by Day of Week")
      plt.ylabel("Count")
      plt.xlabel("Day of Week")
      plt.xticks(rotation=30)
      plt.show()
```



Just comparing these counts visually, it does not appear that there is substantial variation in the number of posts between days. We could compare these counts using inferential statistics, but that is beyond the scope of this notebook.

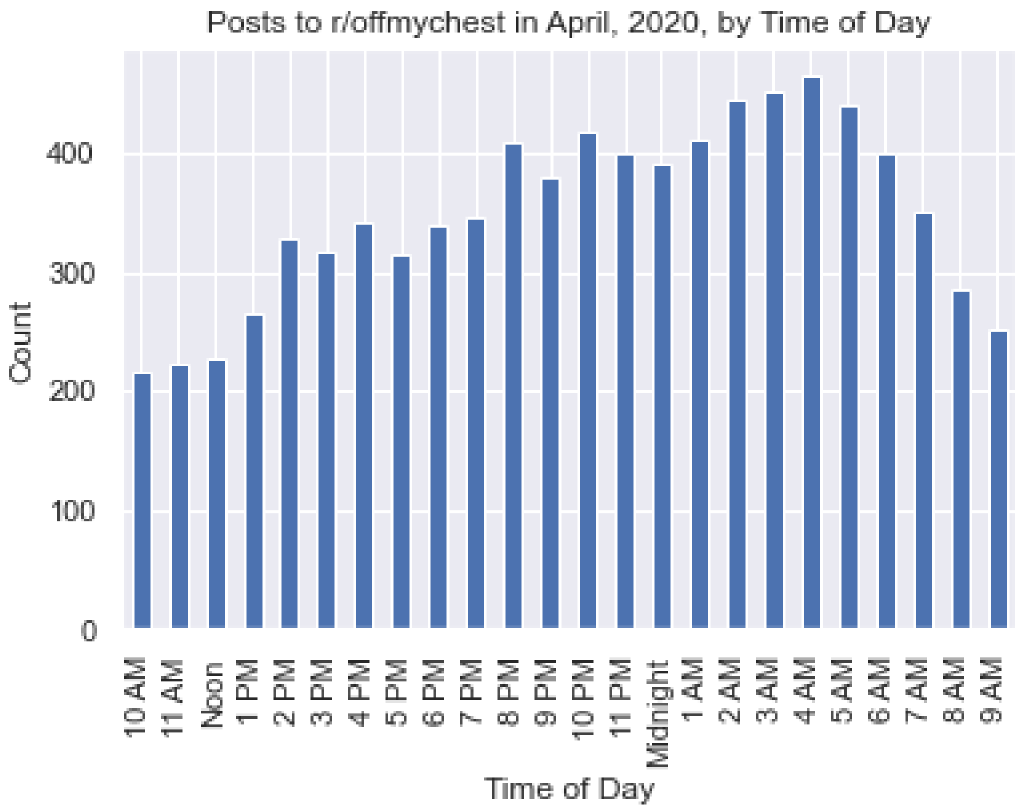
Now let's examine whether there is substantial variation in the average number of posts submitted in any given hour over the course of the day.

[6]:
```
hours = ["Midnight"] + [f"{i} AM" for i in range(1,12)] + ["Noon"] + [f"{i-12}␣
↪PM" for i in range(13,24)]

df["hour"].value_counts().loc[list(range(24))].plot(kind="bar")
plt.xticks(ticks=list(range(24)), labels=hours)
plt.title("Posts to r/offmychest in April, 2020, by Time of Day")
plt.ylabel("Count")
plt.xlabel("Time of Day")
plt.show()
```



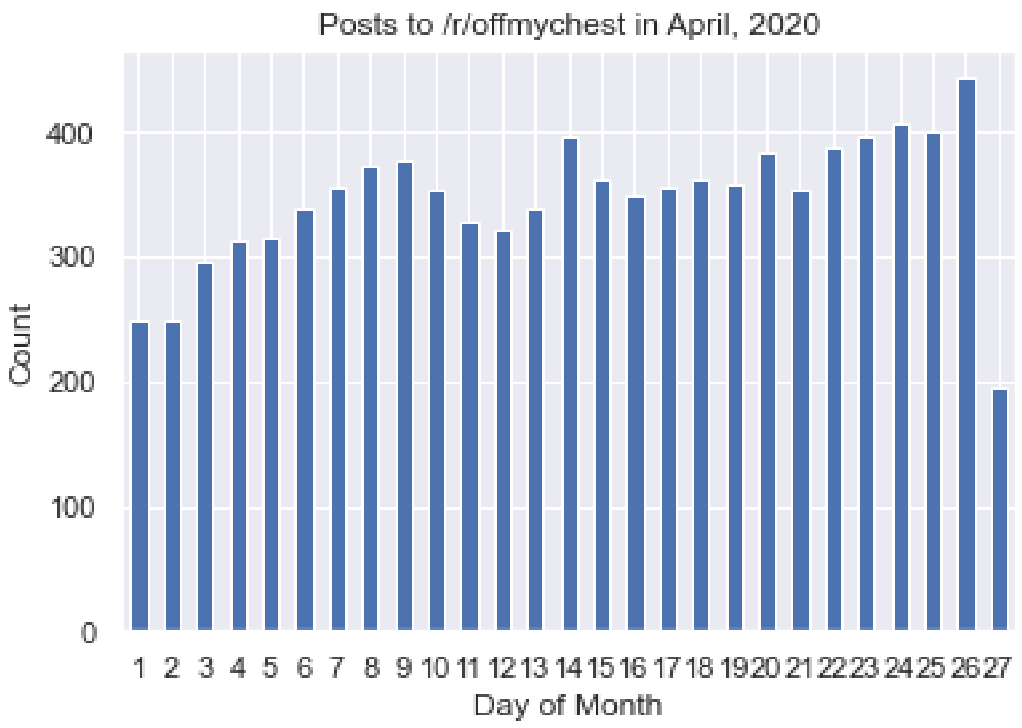
Let's reorder the graph so it starts with the least active hour (10 AM).

```
[69]: hours = [f"{i} AM" for i in range(10,12)] + ["Noon"] + [f"{i-12} PM" for i in␣
       ↪range(13,24)] + ["Midnight"] + [f"{i} AM" for i in range(1,10)]

      df["hour"].value_counts().loc[list(range(10,24)) + list(range(0,10))].
       ↪plot(kind="bar")
      plt.xticks(ticks=list(range(24)), labels=hours)
      plt.title("Posts to r/offmychest in April, 2020, by Time of Day")
      plt.ylabel("Count")
      plt.xlabel("Time of Day")
      plt.show()
```



Finally, let's look at activity over the course of the month.

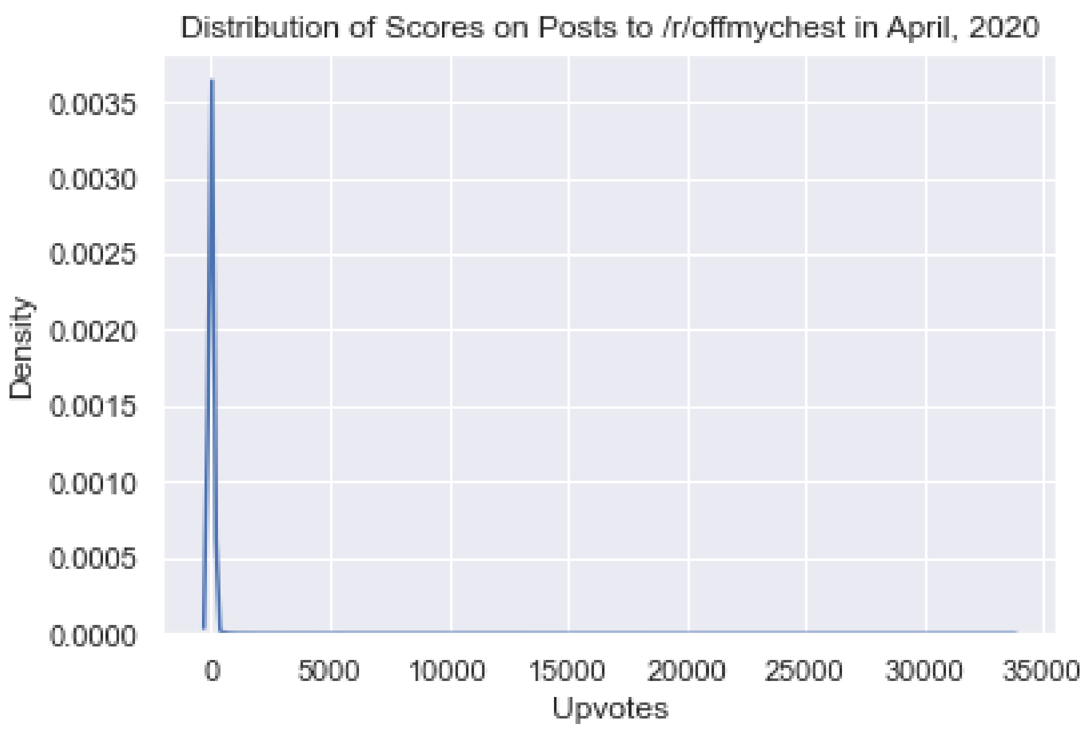
[8]:
```
last_day = 1 + max(df["day"].values)

df["day"].value_counts().loc[list(range(1,last_day))].plot(kind="bar")
plt.title("Posts to /r/offmychest in April, 2020")
plt.ylabel("Count")
plt.xlabel("Day of Month")
plt.xticks(rotation=0)
plt.show()
```



We've learned that there is a lot of variation in activity over the course of the day, from a low at 10 AM to a high at 4 AM. We can use that finding to develop hypotheses about the characteristics of people who use Reddit at different times, what events may prompt people to post to r/offmychest at different times, and so on.

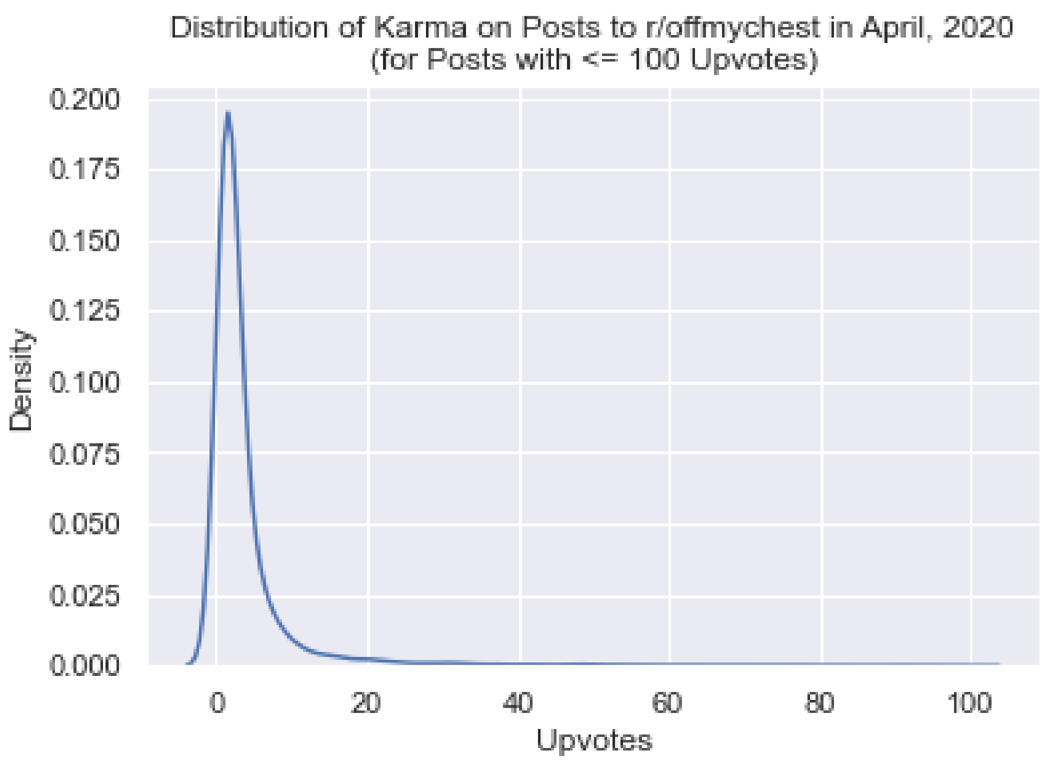
Before we get to that, let's also take a look at a few more things. We'll start with the distribution of scores (karma/upvotes).

[9]:
```
sns.kdeplot(df["score"].values)
plt.title("Distribution of Scores on Posts to /r/offmychest in April, 2020")
plt.xlabel("Upvotes")
plt.show()
```



Posts are tightly clustered near zero, but the distribution exhibits an extreme right skew. Let's zoom in and look at the distribution if we exclude these outliers.

[10]:
```
sns.kdeplot([s for s in df["score"].values if s <= 100])
plt.title("Distribution of Karma on Posts to r/offmychest in April, 2020\n(for␣
 ↪Posts with <= 100 Upvotes)")
plt.xlabel("Upvotes")
plt.show()
```



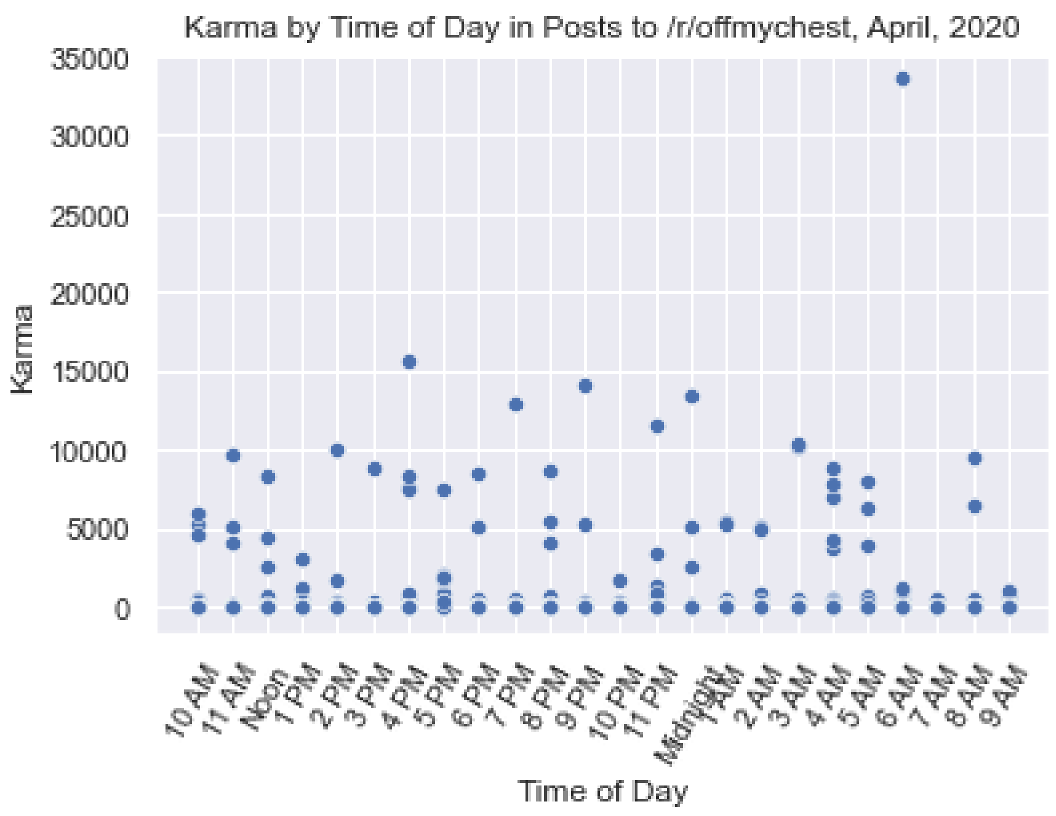
Let's look at the average score:

```
[11]: np.mean(df["score"].values)
```

```
[11]: 47.91894781864842
```

We can also visualize the variation in karma by time of day.

```
[12]: sns.scatterplot(x="hour", y="score", data=df)
      plt.xticks(ticks=list(range(24)), labels=hours, rotation=60)
      plt.ylabel("Karma")
      plt.xlabel("Time of Day")
      plt.title("Karma by Time of Day in Posts to /r/offmychest, April, 2020")
      plt.show()
```

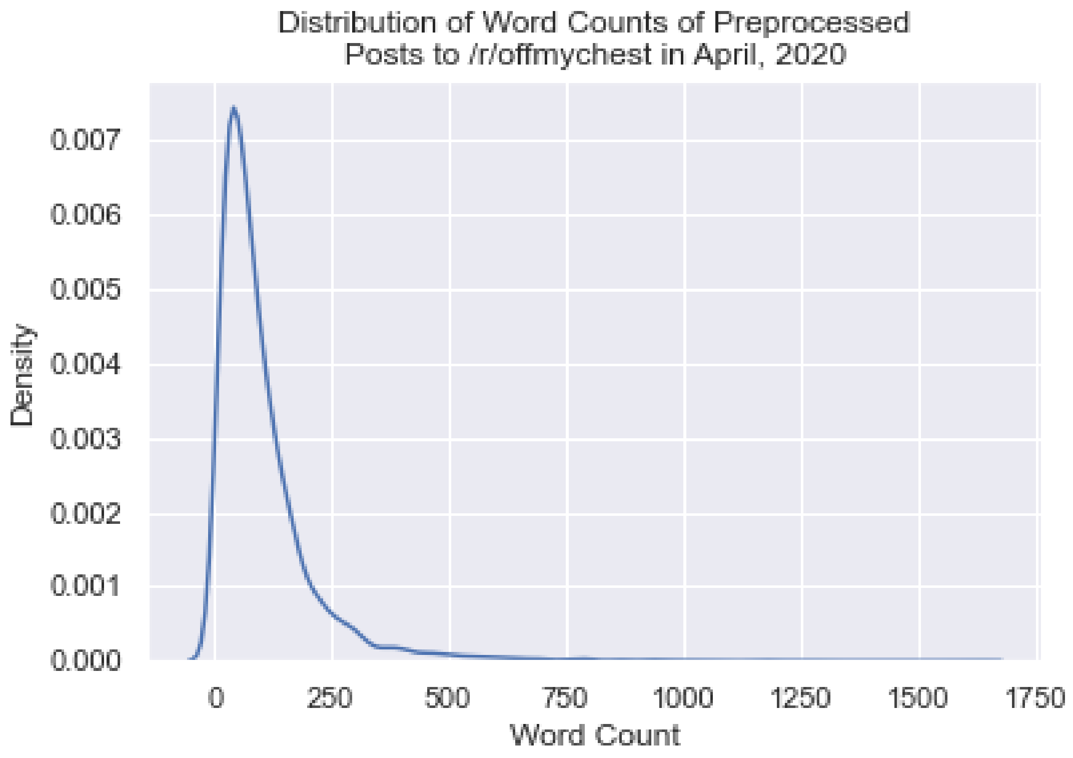


We also know that people write posts of different lengths. The length of a post could be related to the score, to the topics someone discusses, or other factors.

[13]:
```
df["prepped_wordcount"] = [len(t.split()) for t in df["text"]]
```

Now let's plot the distribution of word counts. We again see a long right tail, meaning there are some exceptionally long posts. We'll also print the mean word count. What about short posts, though? If we're interested in the relationship between different topics, we probably want to exclude very short posts that are unlikely to span multiple topics. We'll arbitrarily pick a threshold of 20 words for now.

[14]:
```
sns.kdeplot(df["prepped_wordcount"])
plt.xlabel("Word Count")
plt.title("Distribution of Word Counts of Preprocessed\nPosts to /r/offmychest␣
 ↪in April, 2020")
plt.show()
```



[15]:
```
print(np.mean(df["prepped_wordcount"]))
```

```
103.7953378956373
```

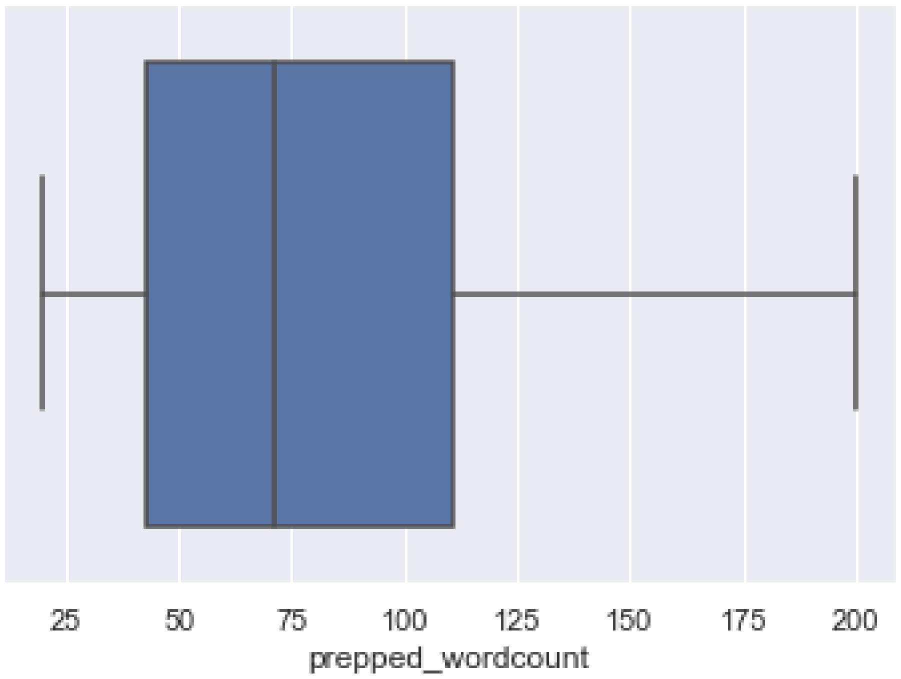
[16]:
```
df = df[df["prepped_wordcount"] >= 20]
```

[17]:
```
df.shape
```

[17]:
```
(8413, 12)
```

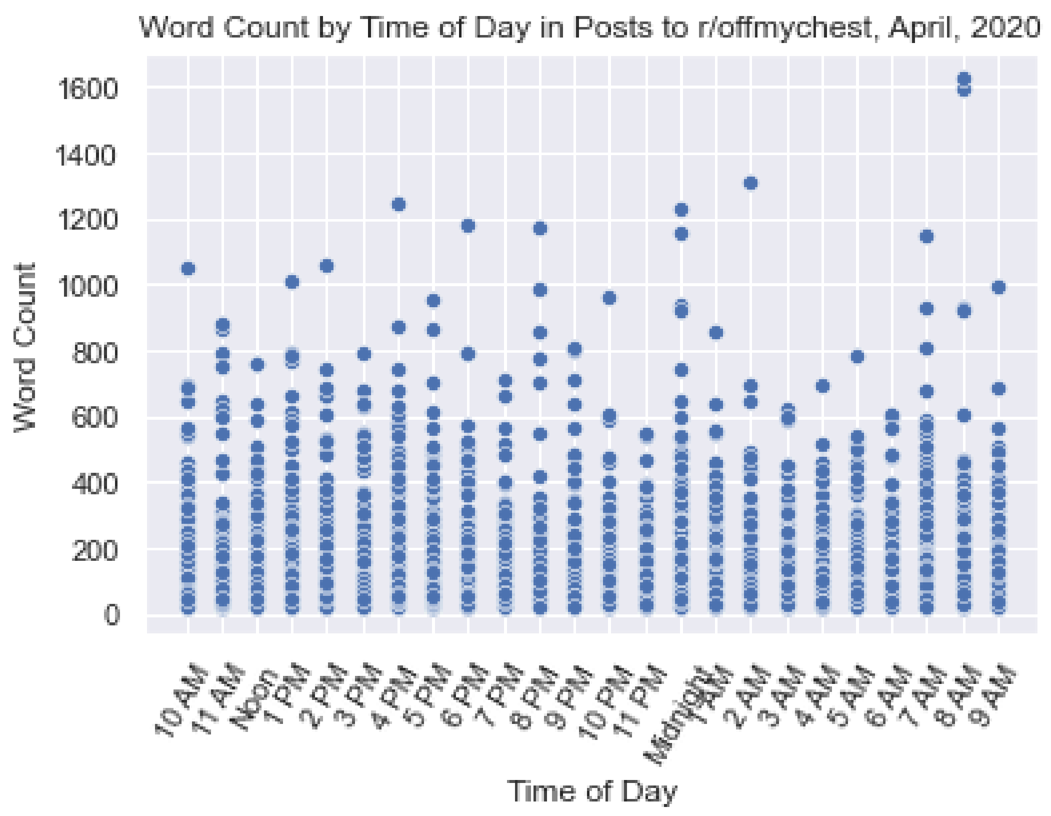
Now let's visualize the distribution of word counts using a boxplot, and we'll only look in the range (20,200) so we can get a sense of the dispersion of word counts around the mean in the absence of outliers

[18]:
```
sns.boxplot(df[df["prepped_wordcount"]<=200]["prepped_wordcount"])
plt.show()
```



Is there a relationship between word count and time of day? Let's check!

```
[19]: sns.scatterplot(x="hour", y="prepped_wordcount", data=df)
      plt.xticks(ticks=list(range(24)), labels=hours, rotation=60)
      plt.ylabel("Word Count")
      plt.xlabel("Time of Day")
      plt.title("Word Count by Time of Day in Posts to r/offmychest, April, 2020")
      plt.show()
```



At long last, let's look at an example post.

[20]:
```
id_ = 3701

title = df[df["user_id"]==id_]["title"].values[0]
selftext = df[df["user_id"]==id_]["selftext"].values[0]
text = df[df["user_id"]==id_]["text"].values[0]
```

[21]:
```
print(title, "\n\n", selftext)
```

```
Suddenly we're important…

 I am so sick of seeing all these commercials from these big companies saying
how appreciative they are for their employees working through this though time.
Toyota, Honda, Walmart, Amazon, etc.. Like seriously all this "essential" worker
nonsense. A month ago we were all replaceable now we're essential? Now they they
care???A month ago they had excuses for why they couldn't pay us more and now
there's a thing called "hero" pay. I sure hope they all remember just how
"essential" we all are after this craziness is over. What a joke.
```

Now let's look at the preprocessed version of the title and text:

```
[22]: text
```

```
[22]: 'suddenly important sick see commercial big company say appreciative employee
      work time toyota honda walmart amazon etc like seriously essential worker
      nonsense month ago replaceable essential care a month ago excuse pay thing call
      hero pay sure hope remember essential craziness joke'
```

Now let's move on to analyzing posts using dictionary methods via the `empath` library. We'll create the object `lexicon` that will analyze posts using 194 separate dictionaries, which range from positive or negative emotion to topics as diverse as optimism, terrorism, fashion, and eating.

```
[23]: lexicon = Empath()
```

```
[24]: # Number of categories
      len(lexicon.cats.keys())
```

```
[24]: 196
```

Here is the full list of categories:

Lexicons Provided by empath

achievement

affection

aggression

air_travel

alcohol

ancient

anger

animal

anonymity

anticipation

appearance

art

attractive

banking

beach

beauty

blue_collar_job

body

breaking

business

car

celebration

cheerfulness

childish

children

cleaning

clothing

cold

college

communication

competing

computer

confusion

contentment

cooking

crime

dance

death

deception

disappointment

disgust

dispute

divine

domestic_work

dominant_heirarchical

dominant_personality

driving

eating

economics

emotional

envy

exasperation

exercise

exotic

fabric

family

farming

fashion

fear

feminine

fight

fire

friends

fun

furniture

gain

giving

government

hate

healing

health

hearing

help

heroic

hiking

hipster

home

horror

hygiene

independence

injury

internet

irritability

journalism

joy

kill

law

leader

legend

leisure

liquid

listen

love

lust

magic

masculine

medical_emergency

medieval

meeting

messaging

military

money

monster

morning

movement

music

musical

negative_emotion

neglect

negotiate

nervousness

night

noise

occupation

ocean

office

optimism

order

pain

party

payment

pet

philosophy

phone

plant

play

politeness

politics

poor

positive_emotion

power

pride

prison

programming

rage

reading

real_estate

religion

restaurant

ridicule

royalty

rural

sadness

sailing

school

science

sexual

shame

shape_and_size

ship

shopping

sleep

smell

social_media

sound

speaking

sports

stealing

strength

suffering

superhero

surprise

swearing_terms

swimming

sympathy

technology

terrorism

timidity

tool

torment

tourism

toy

traveling

trust

ugliness

urban

vacation

valuable

vehicle

violence

war

warmth

water

weakness

wealthy

weapon

weather

wedding

white_collar_job

work

worship

writing

Let's pick a few lexicons like `negative_emotion` and `positive_emotion` and analyze the example post we saw above. The code below will print the name of the lexicon and the number of words from that lexicon that are found in the document.

[25]:
```
results = lexicon.analyze(text)

test_cats = ["negative_emotion", "positive_emotion", "blue_collar_job",␣
↪"horror", "domestic_work", "valuable", "work"]

for cat in test_cats:
    print(f"{cat}: {results[cat]}")
```

```
negative_emotion: 2.0
positive_emotion: 2.0
blue_collar_job: 2.0
horror: 0.0
domestic_work: 0.0
valuable: 2.0
work: 3.0
```

One issue with this kind of approach is now apparent: even in documents that seem to be clearly negative or positive, we may see words that suggest the opposite. In this example, two words are in the `negative_emotion` lexicon, but two words are also in the `positive_emotion` lexicon. We will talk about document classification, including sentiment, elsewhere, but results like this can be problematic for different kinds of analyses.

One other thing to keep in mind is that these are raw word counts. Two words are from `negative_sentiment`, two words are from `positive_sentiment`, and so on. However, we also know there is quite a bit of variation in how long documents are, and the length of documents could be associated with the breadth of topics covered. Let's check the document length.

```
[26]: len(text.split())
```

```
[26]: 42
```

This example is 42 words long. We can use the raw word counts, like we do above, but we can also normalize the value for each dictionary using the normalize=True argument, as in the code below. This divides the number of words for a given dictionary by the total number of words in the document. In other words, this gives us the percentage of words in a document that are in a particular lexicon.

We can see that the normalized value of `blue_collar_job` is approximately 0.048. We can see in the line of code below that that this is simply 2 divided by 42, where 2 is the number of words in the `blue_collar_job` lexicon found in the document and 42 is the length of the document.

```
[27]: print(lexicon.analyze(text, normalize=False)["blue_collar_job"])
      print(lexicon.analyze(text, normalize=True)["blue_collar_job"])
```

```
2.0
0.047619047619047616
```

```
[28]: 2/42
```

```
[28]: 0.047619047619047616
```

If we run lexicon.analyze(text), we get the valule for each lexicon. We can use code like following to check for any nonzero values, meaning we can check which dictionaries have words that actually appear in the document. We see topics as varied as office, medical_emergency, ridicule, and superhero.

```
[29]: [key for key, value in lexicon.analyze(text).items() if value > 0]
```

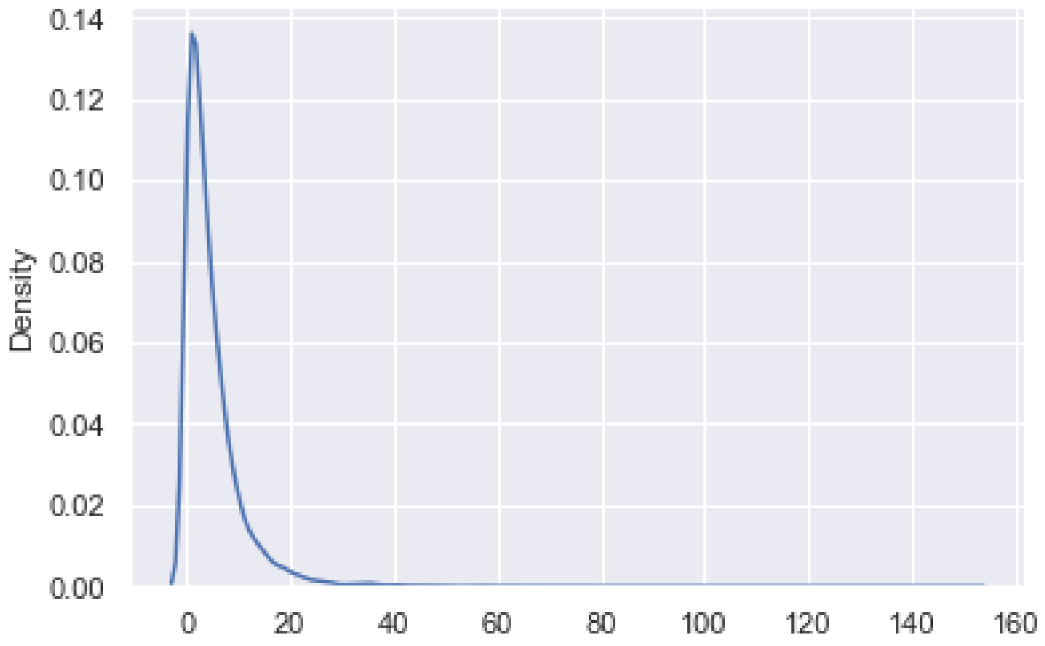
```
[29]: ['office',
       'money',
       'medical_emergency',
       'hate',
       'occupation',
       'health',
       'wealthy',
       'banking',
       'blue_collar_job',
       'ridicule',
       'optimism',
       'superhero',
       'business',
       'surprise',
       'movement',
       'zest',
       'healing',
       'legend',
       'heroic',
       'communication',
       'hearing',
       'trust',
       'politeness',
       'speaking',
       'listen',
       'phone',
       'work',
       'valuable',
       'fun',
       'fashion',
       'shape_and_size',
       'pain',
       'negative_emotion',
       'cleaning',
       'payment',
       'children',
       'giving',
       'positive_emotion']
```

[30]:
```
def score_category(s, category, normalize=False):
    """
    This function returns the score for a given document for a particular␣
↪category
    using empath's dictionaries. The optional argument `normalize` defaults to␣
↪False.
    """
    res = lexicon.analyze(s, categories=[category], normalize=normalize)
    return res[category]
```

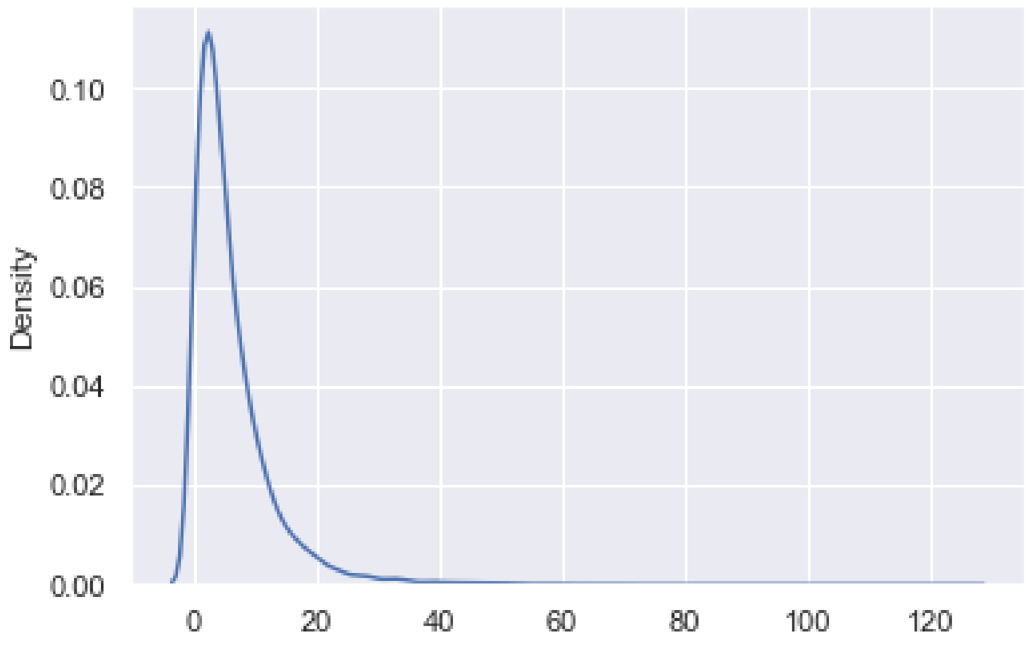
Now let's look at trends in the entire corpus. We'll start with positive and negative sentiment dictionaries, namely `positive_emotion` and `negative_emotion`.

[31]:
```
df["pos"] = [score_category(s, "positive_emotion") for s in df["text"].values]
df["neg"] = [score_category(s, "negative_emotion") for s in df["text"].values]
```

[32]:
```
sns.kdeplot(df["pos"].values)
plt.show()
```



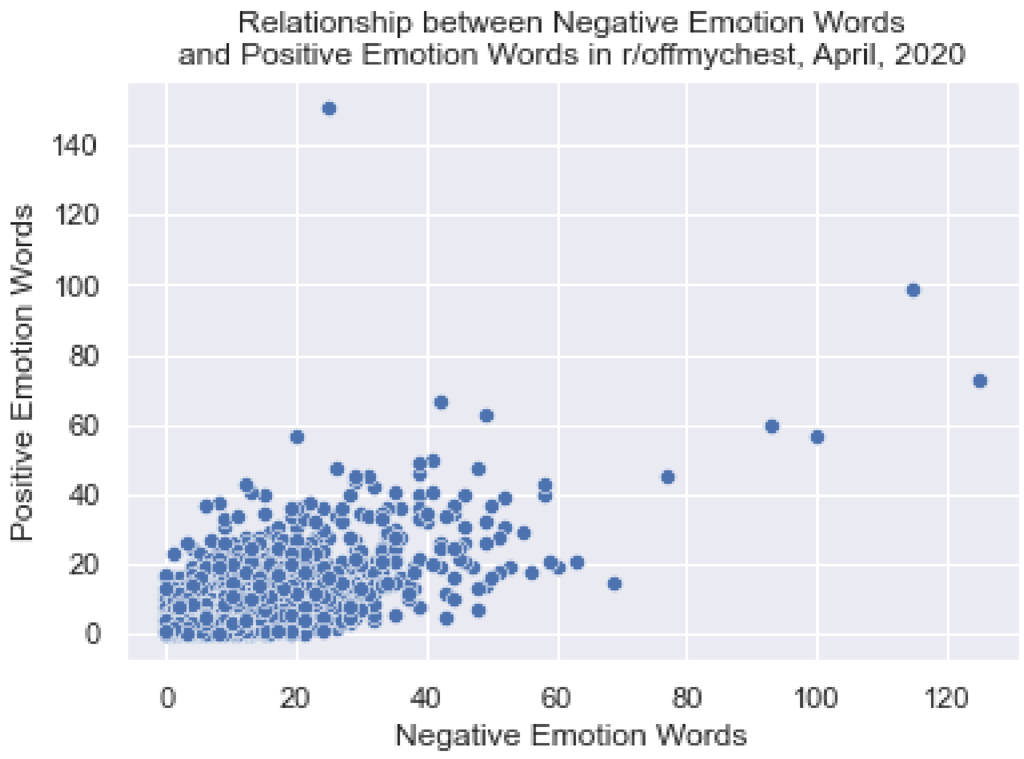
[33]:
```
sns.kdeplot(df["neg"].values)
plt.show()
```



As we saw with the example above, it appears that negative and positive sentiment words co-occur.

We see a positive correlation between the two.

[34]:
```
sns.scatterplot(x="neg", y="pos", data=df)
plt.xlabel("Negative Emotion Words")
plt.ylabel("Positive Emotion Words")
plt.title("Relationship between Negative Emotion Words\nand Positive Emotion␣
 ↪Words in r/offmychest, April, 2020")
plt.show()

print(f"Correlation between pos and neg: {df['pos'].corr(df['neg'])}")
```

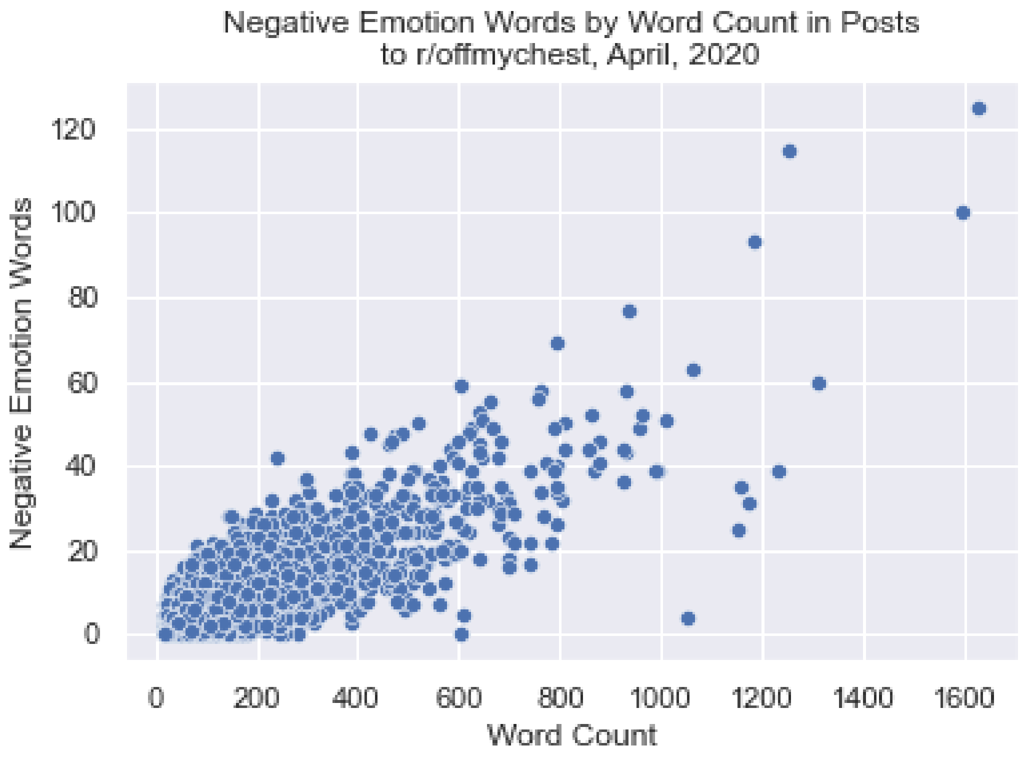


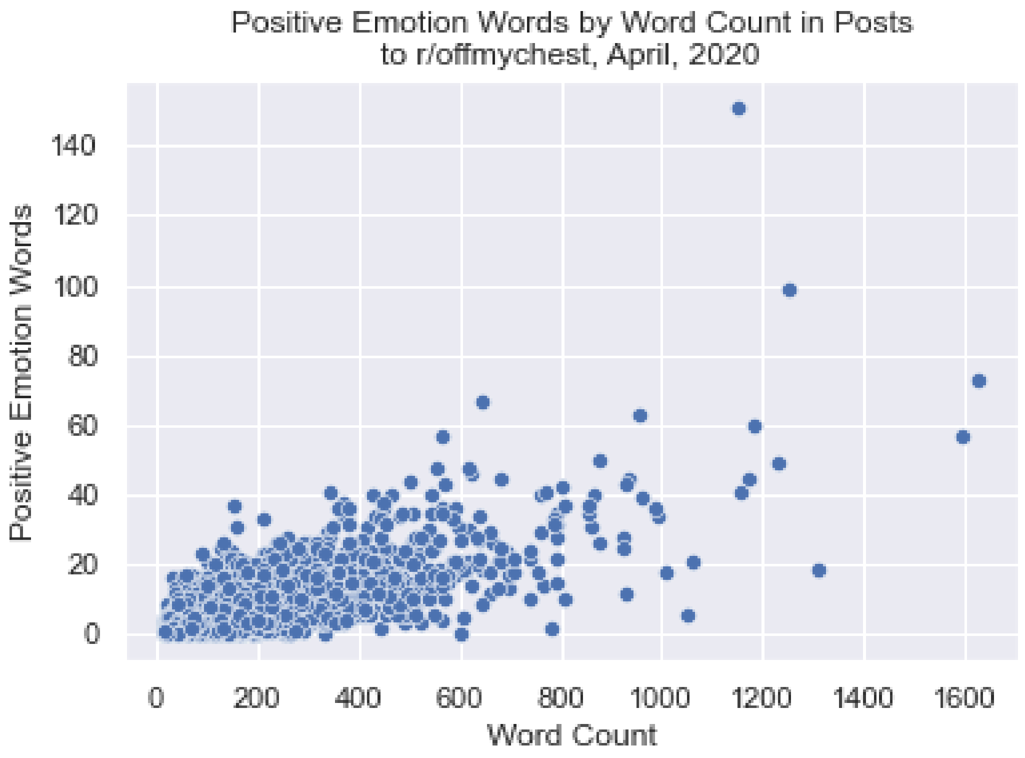
```
Correlation between pos and neg: 0.6530934463177493
```

Do people who write longer posts to use more emotion words?

[35]:
```
sns.scatterplot(x="prepped_wordcount", y="neg", data=df)
plt.xlabel("Word Count")
plt.ylabel("Negative Emotion Words")
plt.title("Negative Emotion Words by Word Count in Posts\nto r/offmychest,␣
 ↪April, 2020")
plt.show()

sns.scatterplot(x="prepped_wordcount", y="pos", data=df)
```

```
plt.xlabel("Word Count")
plt.ylabel("Positive Emotion Words")
plt.title("Positive Emotion Words by Word Count in Posts\nto r/offmychest,␣
 ↪April, 2020")
plt.show()
```

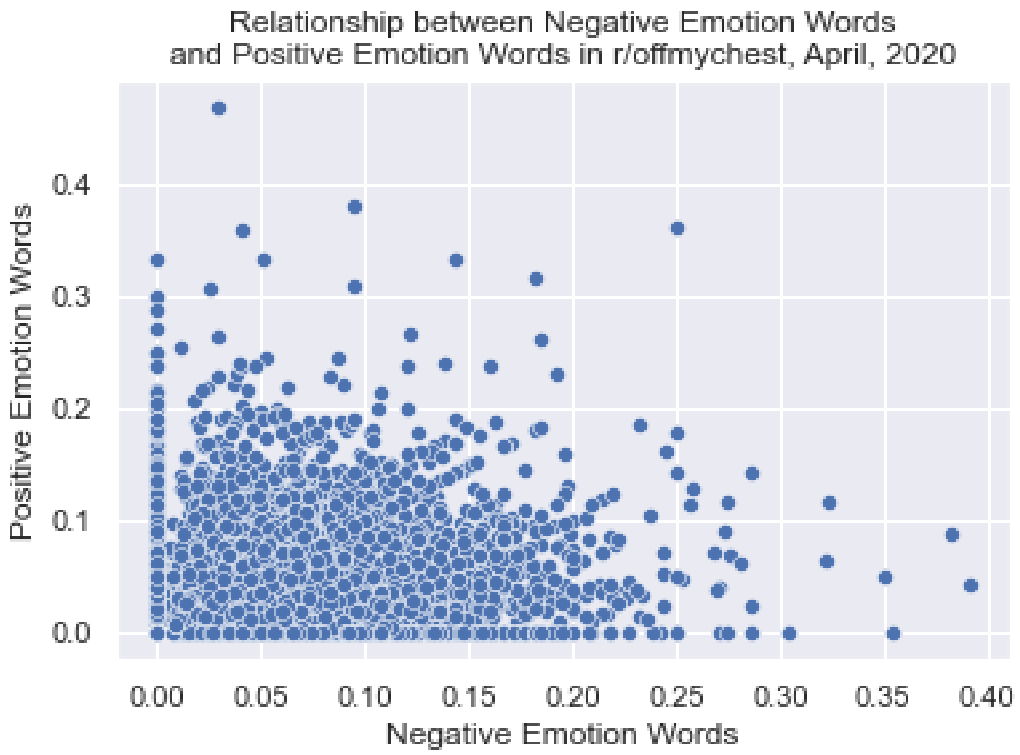




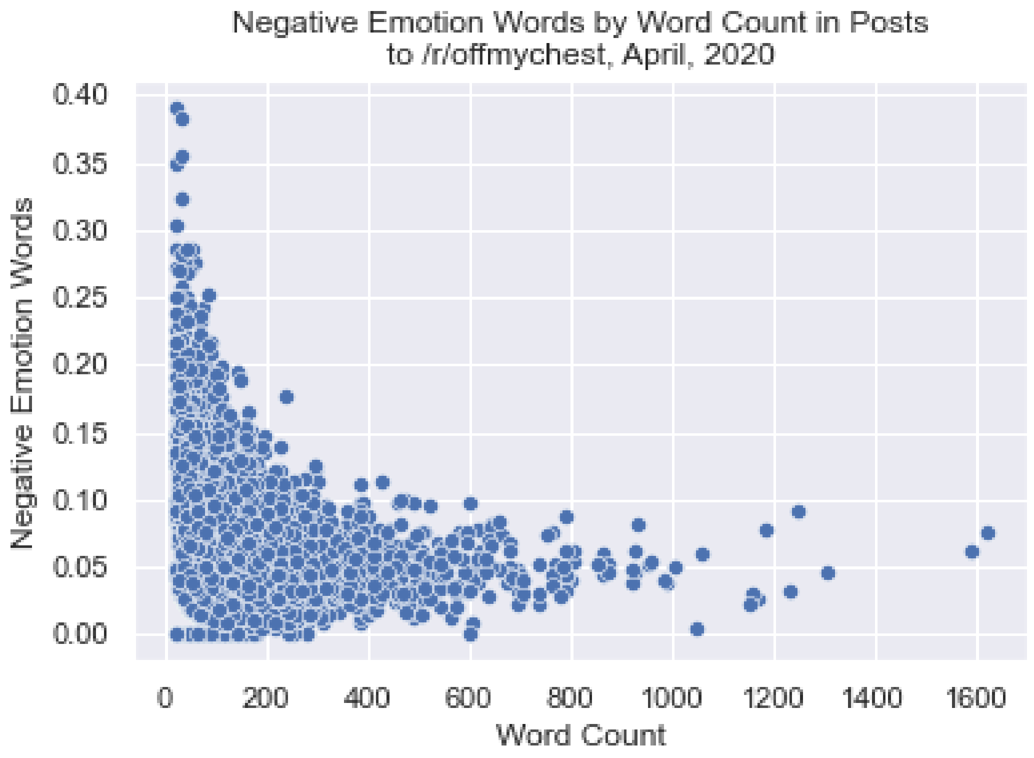
One thing you might notice about this approach is that we would expect these seemingly linear relationships based on increasing word counts alone. Longer posts will have more of the same words unless longer posts differ in content from shorter posts. Let's normalize the measures we're using to address this. The code below will replace the "pos" and "neg" variable we just created with new versions normalized by the length of each post.

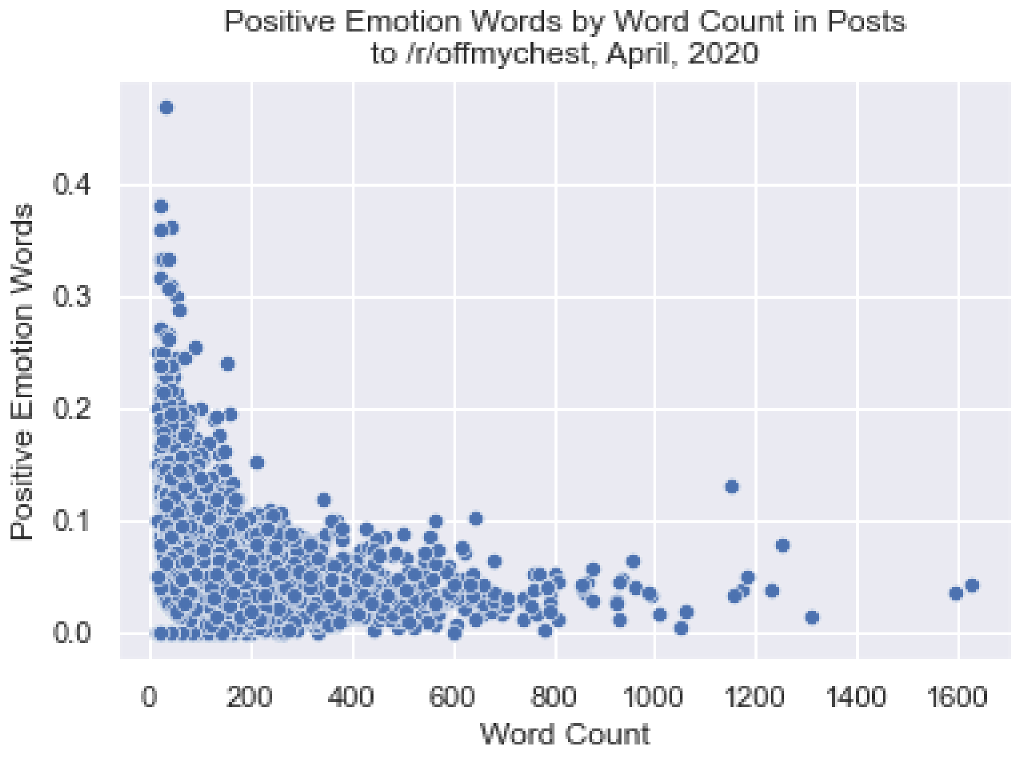
[38]:
```
df["pos"] = [score_category(s, "positive_emotion", normalize=True) for s in␣
 ↪df["text"].values]
df["neg"] = [score_category(s, "negative_emotion", normalize=True) for s in␣
 ↪df["text"].values]
```

[39]:
```
sns.scatterplot(x="neg", y="pos", data=df)
plt.xlabel("Negative Emotion Words")
plt.ylabel("Positive Emotion Words")
plt.title("Relationship between Negative Emotion Words\nand Positive Emotion␣
 ↪Words in r/offmychest, April, 2020")
plt.show()

print(f"Correlation between pos and neg: {df['pos'].corr(df['neg'])}")

sns.scatterplot(x="prepped_wordcount", y="neg", data=df)
plt.xlabel("Word Count")
plt.ylabel("Negative Emotion Words")
plt.title("Negative Emotion Words by Word Count in Posts\nto /r/offmychest,␣
 ↪April, 2020")
plt.show()

sns.scatterplot(x="prepped_wordcount", y="pos", data=df)
plt.xlabel("Word Count")
plt.ylabel("Positive Emotion Words")
plt.title("Positive Emotion Words by Word Count in Posts\nto /r/offmychest,␣
 ↪April, 2020")
plt.show()
```



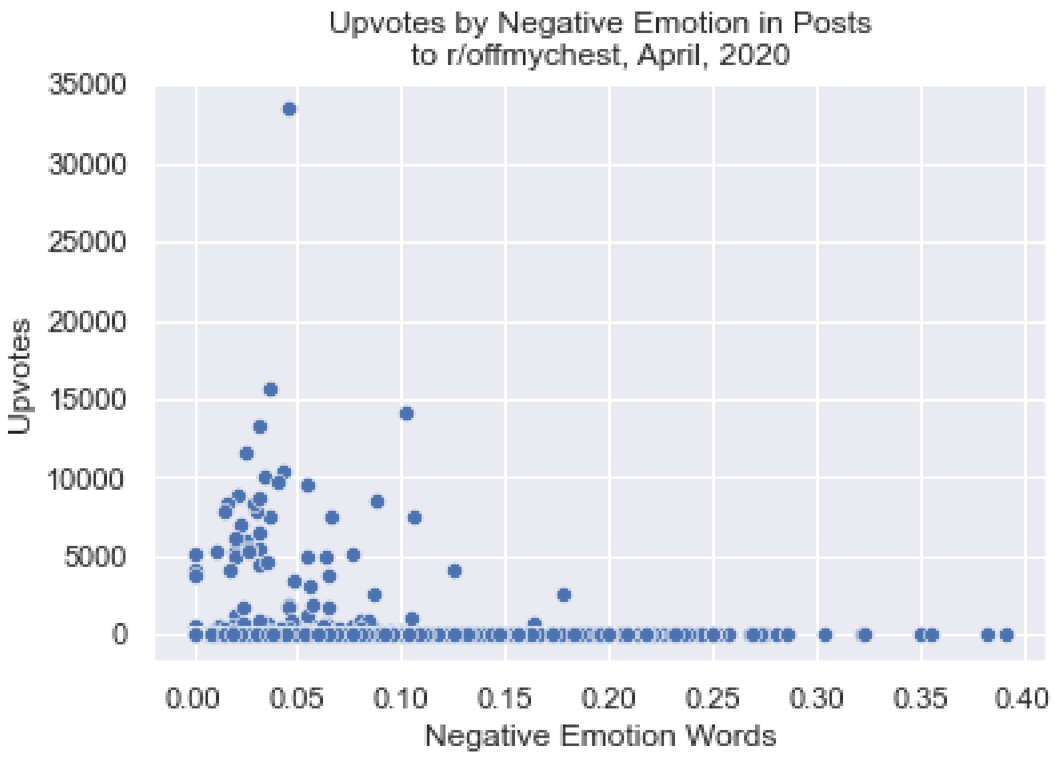
```
Correlation between pos and neg: 0.11394409871240206
```

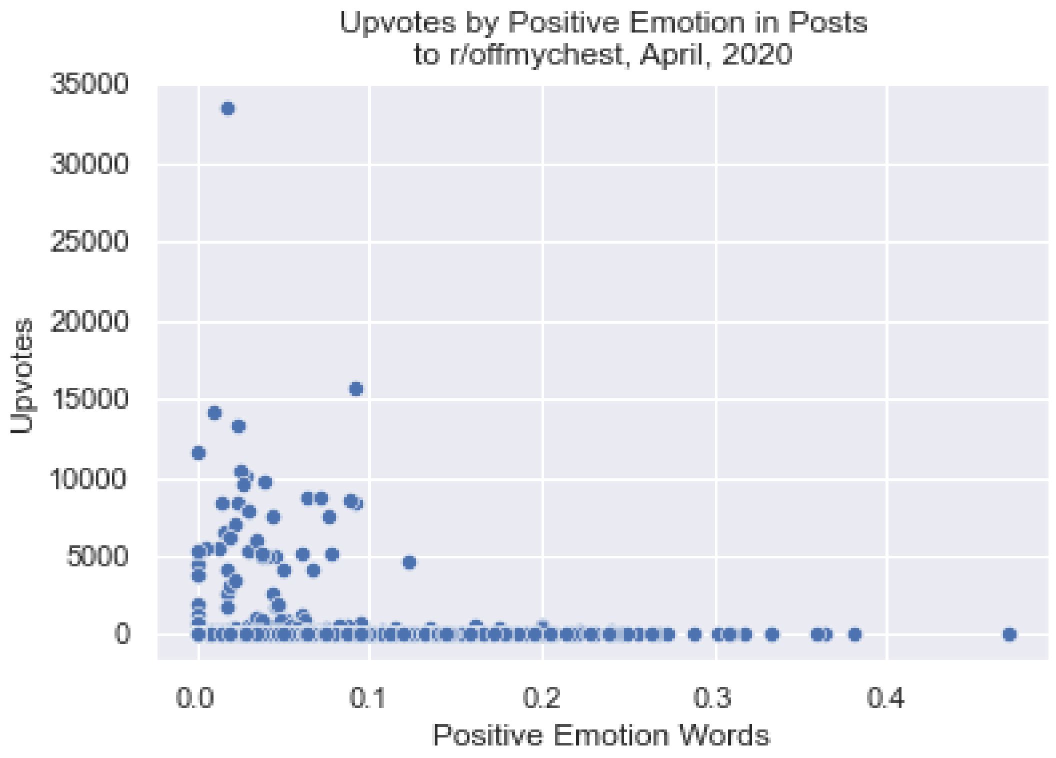

Negative Emotion Words by Word Count in Posts to /r/offmychest, April, 2020
Negative Emotion Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Word Count
0
200
400
600
800
1000
1200
1400
1600


Positive Emotion Words by Word Count in Posts to /r/offmychest, April, 2020
Positive Emotion Words
0.0
0.1
0.2
0.3
0.4
Word Count
0
200
400
600
800
1000
1200
1400
1600

Is the success of a post (in terms of it's karma/upvotes) associated with sentiment?

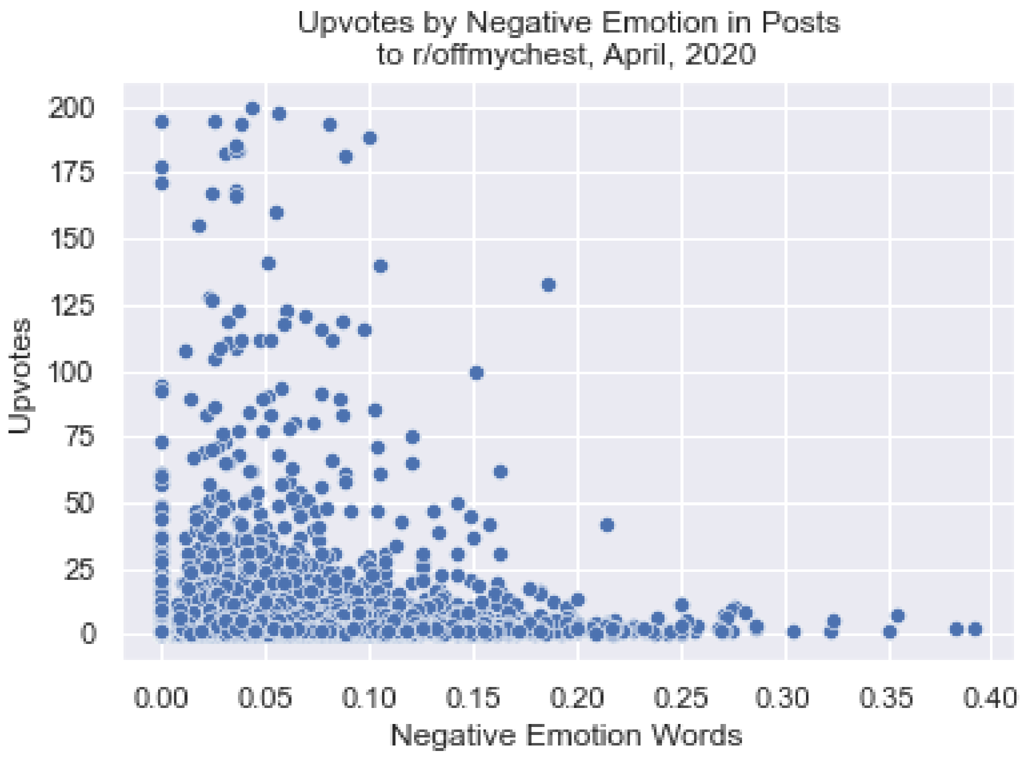
[40]:
```
sns.scatterplot(x="neg", y="score", data=df)
plt.xlabel("Negative Emotion Words")
plt.ylabel("Upvotes")
plt.title("Upvotes by Negative Emotion in Posts\nto r/offmychest, April, 2020")
plt.show()

sns.scatterplot(x="pos", y="score", data=df)
plt.xlabel("Positive Emotion Words")
plt.ylabel("Upvotes")
plt.title("Upvotes by Positive Emotion in Posts\nto r/offmychest, April, 2020")
plt.show()
```

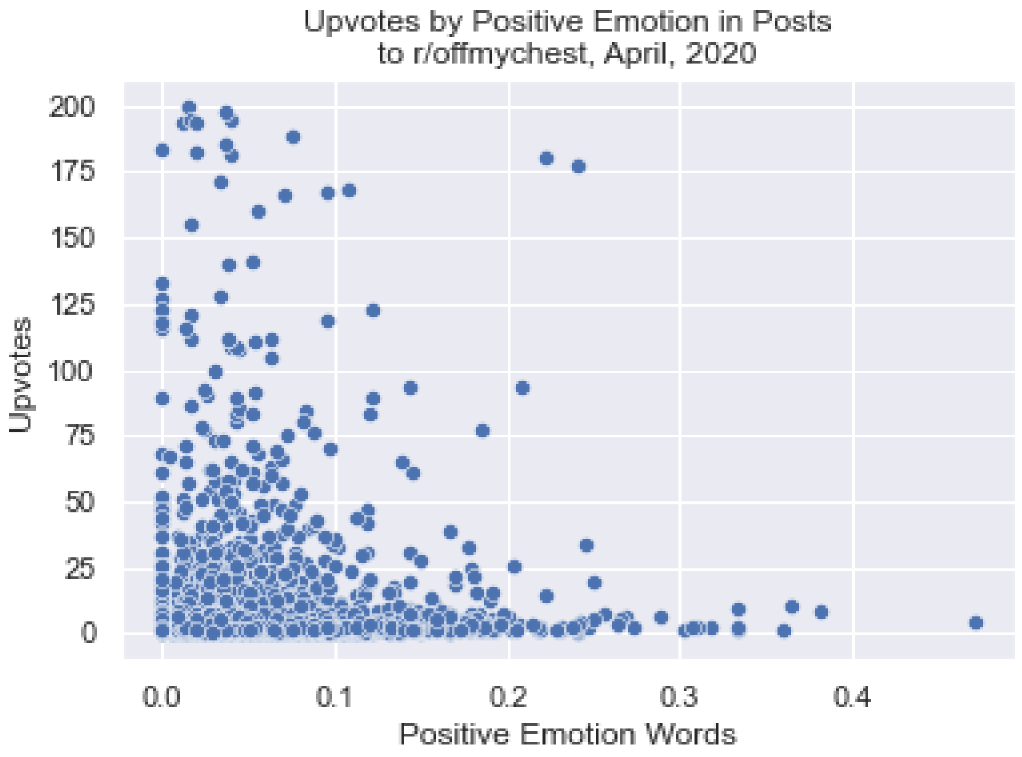




Notice that the y-axis ranges from zero to 35,000! Let's take a look at these relationships again, this time excluding posts with more than 200 upvotes.

[41]:
```
sns.scatterplot(x="neg", y="score", data=df[df["score"]<=200])
plt.xlabel("Negative Emotion Words")
plt.ylabel("Upvotes")
plt.title("Upvotes by Negative Emotion in Posts\nto r/offmychest, April, 2020")
plt.show()

sns.scatterplot(x="pos", y="score", data=df[df["score"]<=200])
plt.xlabel("Positive Emotion Words")
plt.ylabel("Upvotes")
plt.title("Upvotes by Positive Emotion in Posts\nto r/offmychest, April, 2020")
plt.show()
```

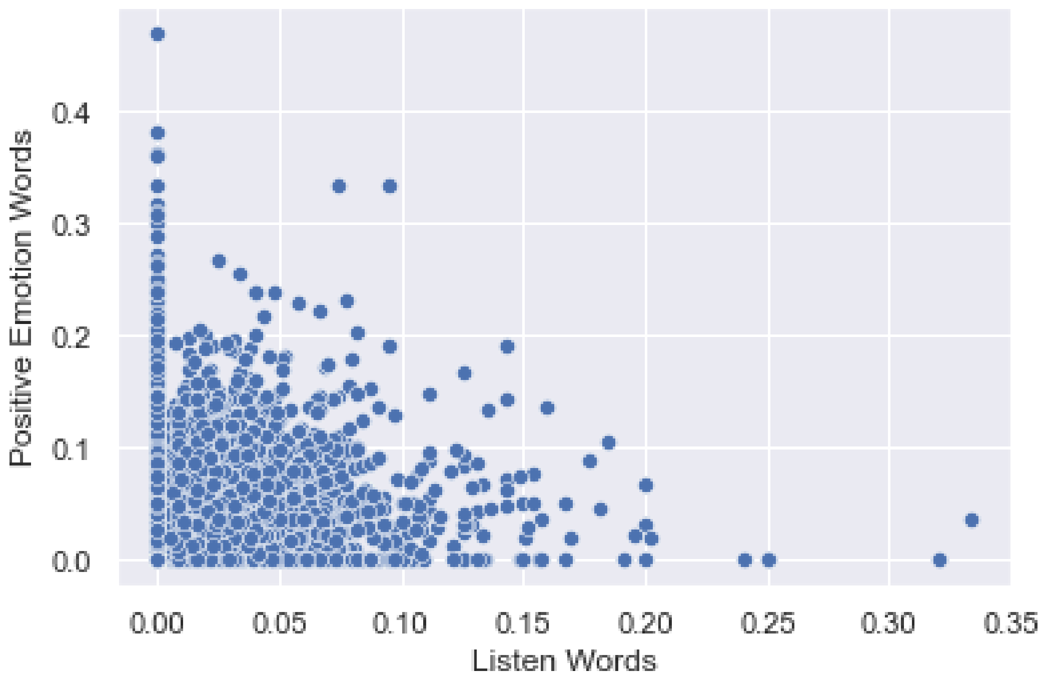

Upvotes by Negative Emotion in Posts
to r/offmychest, April, 2020
Upvotes
0
25
50
75
100
125
150
175
200
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Negative Emotion Words

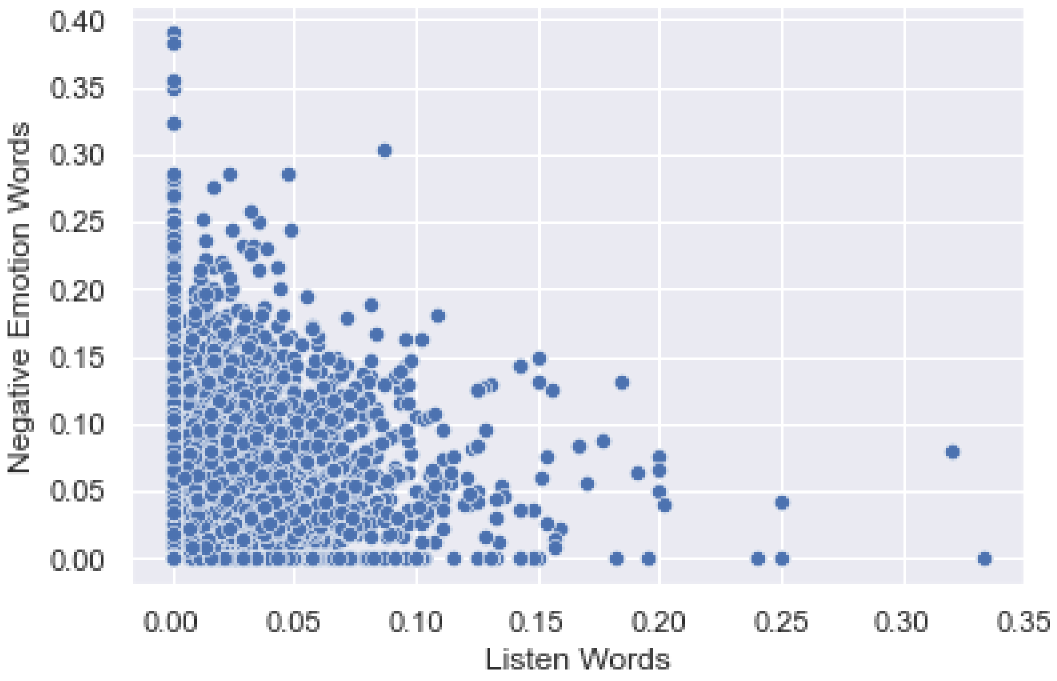

Upvotes by Positive Emotion in Posts
to r/offmychest, April, 2020
Upvotes
0
25
50
75
100
125
150
175
200
0.0
0.1
0.2
0.3
0.4
Positive Emotion Words

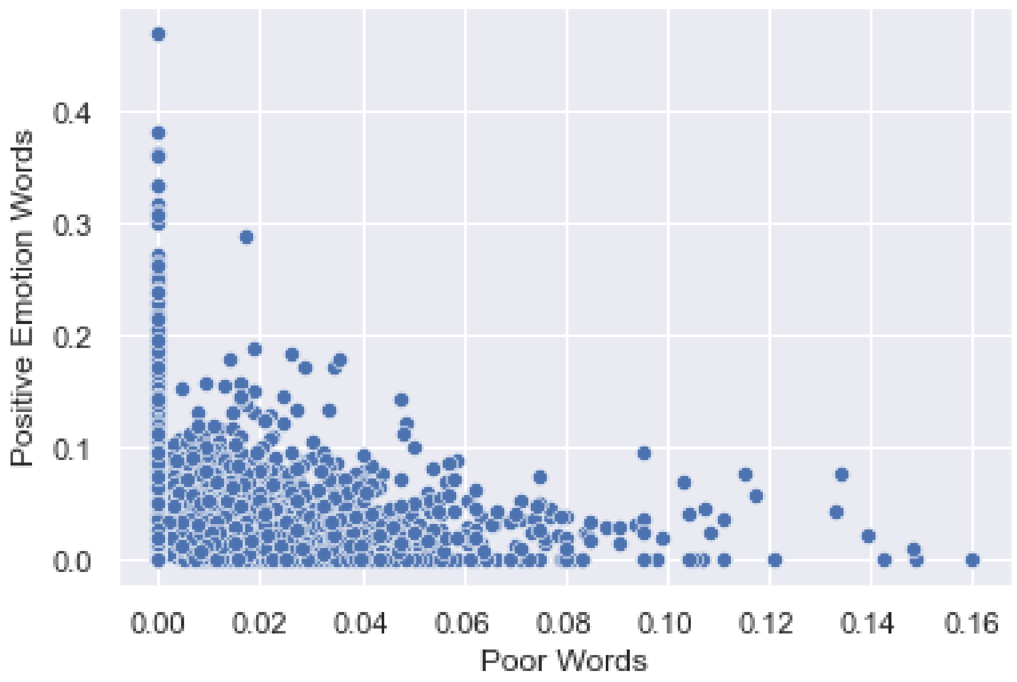
Let's check a few more categories. We'll try `listen`, `poor`, and `politics` first.

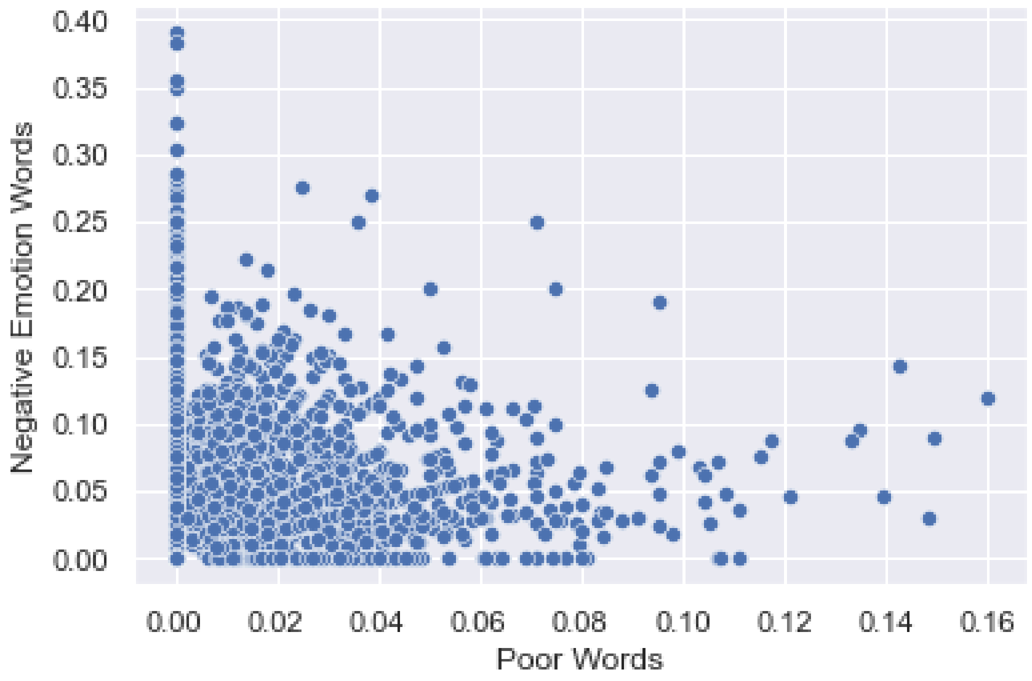
[42]:
```
df["listen"] = [score_category(s, "listen", normalize=True) for s in df["text"].
↪values]
df["poor"] = [score_category(s, "poor", normalize=True) for s in df["text"].
↪values]
df["politics"] = [score_category(s, "politics",  normalize=True) for s in␣
↪df["text"].values]
```

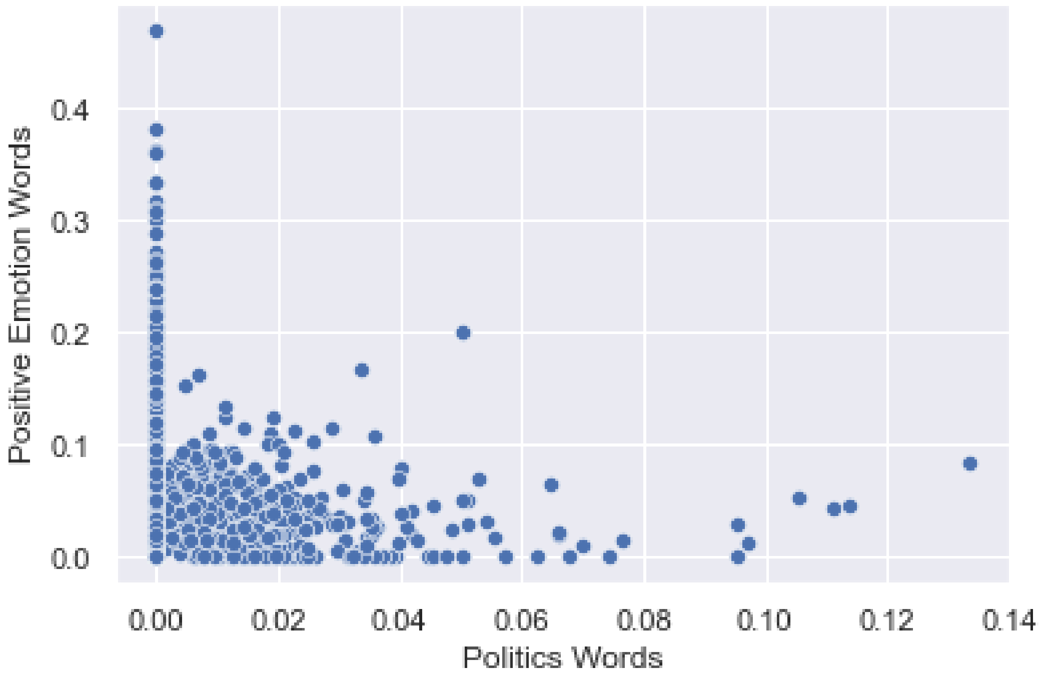
[43]:
```
sns.scatterplot(x="listen", y="pos", data=df)
plt.xlabel("Listen Words")
plt.ylabel("Positive Emotion Words")
plt.show()

sns.scatterplot(x="listen", y="neg", data=df)
plt.xlabel("Listen Words")
plt.ylabel("Negative Emotion Words")
plt.show()

sns.scatterplot(x="poor", y="pos", data=df)
plt.xlabel("Poor Words")
plt.ylabel("Positive Emotion Words")
plt.show()

sns.scatterplot(x="poor", y="neg", data=df)
plt.xlabel("Poor Words")
plt.ylabel("Negative Emotion Words")
plt.show()

sns.scatterplot(x="politics", y="pos", data=df)
plt.xlabel("Politics Words")
plt.ylabel("Positive Emotion Words")
plt.show()

sns.scatterplot(x="politics", y="neg", data=df)
plt.xlabel("Politics Words")
plt.ylabel("Negative Emotion Words")
plt.show()
```

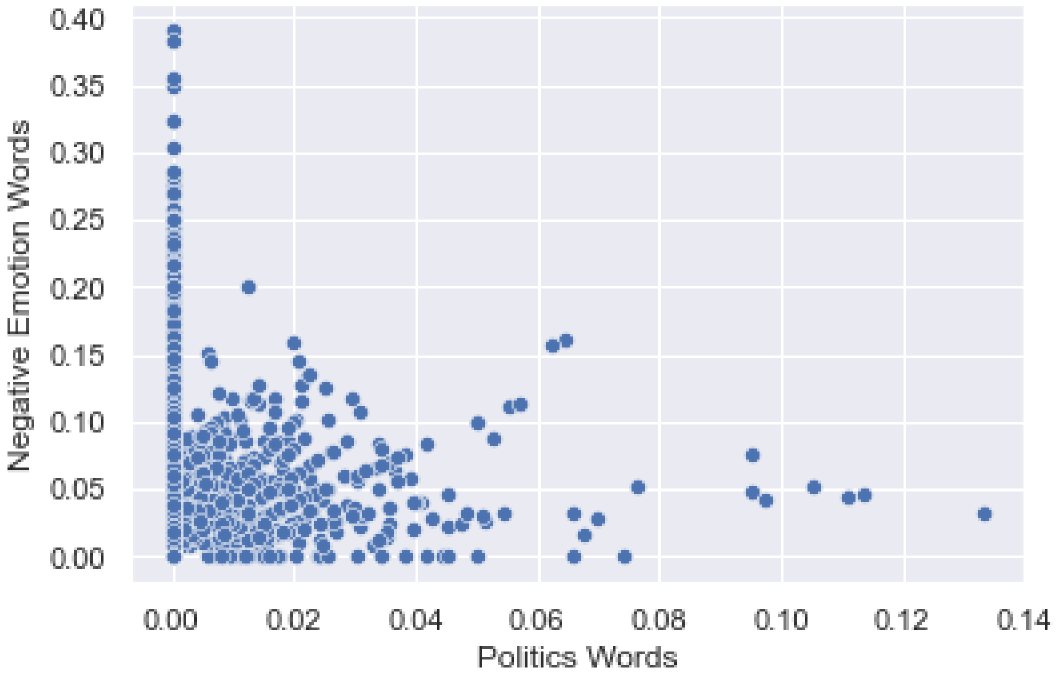

Positive Emotion Words
0.0
0.1
0.2
0.3
0.4
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Listen Words


Negative Emotion Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Listen Words


Positive Emotion Words
0.0
0.1
0.2
0.3
0.4
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Poor Words


Negative Emotion Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Poor Words





`pandas` has some incredibly powerful methods for manipulating data structures. For the sake of learning via exploratory analysis, we're going to apply each of the 194 dictionaries to each of the posts. The code below should run pretty quickly, despite applying 194 dictionaries to over 8,000 documents! After running that, we'll again use the .head() method to view a few rows of the dataframe. We can see that we now have over 200 variables, and we can scroll to the side to see many of them. These represent each post's score for each lexicon.

[44]:
```
df[list(lexicon.cats)] = df.text.apply(lambda x: pd.Series(lexicon.analyze(x,␣
↪normalize=True)))
```

[45]:
```
df.head()
```

[45]:
```
                  date  score  \
1  2020-04-01 00:01:27      2
9  2020-04-01 00:21:43      2
10 2020-04-01 00:22:54      1
11 2020-04-01 00:24:17      3
12 2020-04-01 00:28:02      2

                                               title  \
1                         i am so unhappy with my life
9   How many people are considering major life cha…
10                          a loop of paradoxical envy
11                          It's 2am and I can't sleep
12                                Disturbing thoughts?

                                            selftext  \
1   this rant isn't really going to be about anyth…
9   Being trapped in a home I hate in a situation …
10  i would say the reason as to why i mess everyt…
11  Again it's late at night and I'm laying in my …
12  Alright, I already made another post on this s…

                                                text  day  hour  minute  \
1   unhappy life rant go specific pretty messy try…    1     0       1
9   people consider major life change current situ…    1     0      21
10  loop paradoxical envy reason mess talk fact je…    1     0      22
11  sleep late night lay bed wide awake stay awake…    1     0      24
12  disturb thought alright post subreddit probabl…    1     0      28

    dayofweek  user_id  … monster  ocean    giving  contentment  writing  \
1   Wednesday      856  …     0.0    0.0  0.000000     0.015789      0.0
9   Wednesday     6680  …     0.0    0.0  0.000000     0.000000      0.0
10  Wednesday     7292  …     0.0    0.0  0.044444     0.000000      0.0
11  Wednesday     4636  …     0.0    0.0  0.000000     0.019608      0.0
12  Wednesday     3679  …     0.0    0.0  0.000000     0.014706      0.0

       rural  positive_emotion  musical  racism  stigma
1   0.005263          0.042105      0.0     0.0     0.0
```

```
9   0.000000          0.045455      0.0      0.0      0.0
10  0.000000          0.022222      0.0      0.0      0.0
11  0.000000          0.058824      0.0      0.0      0.0
12  0.000000          0.044118      0.0      0.0      0.0

[5 rows x 210 columns]
```

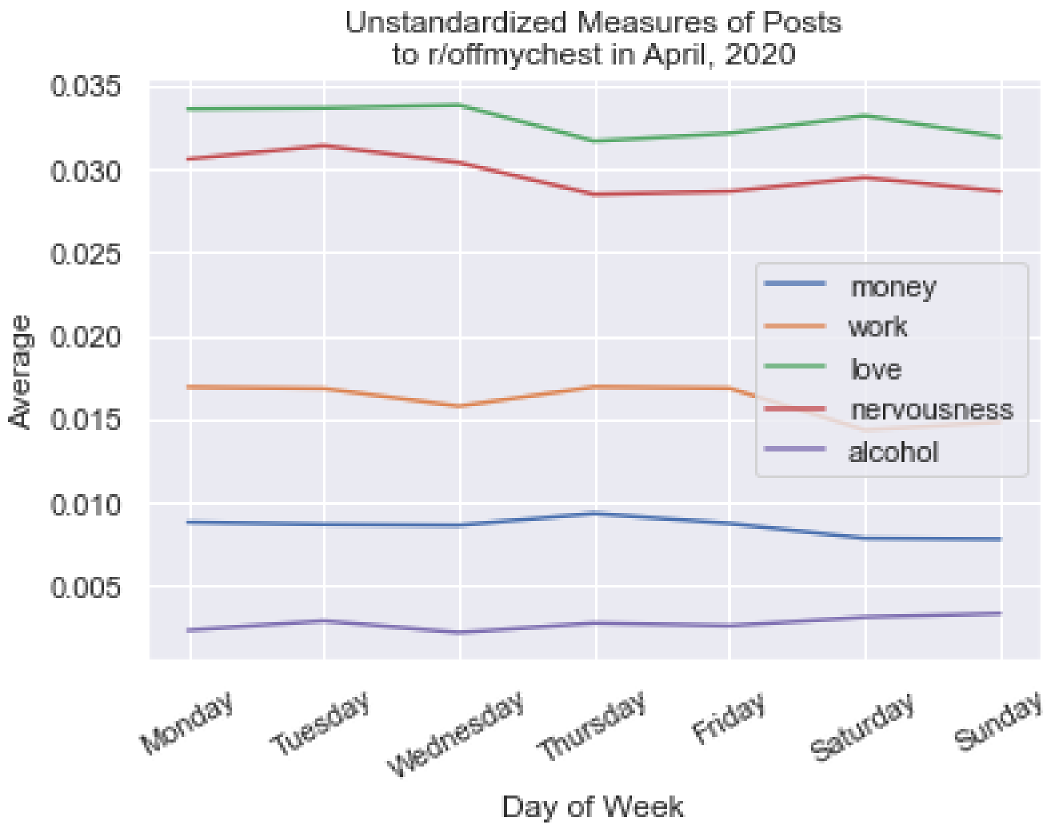
0.2.1 Standardizing Variables

Before we dive into the rest of our analyses, we are going to "standardize" the variables. If you have some background in statistics, this may be familiar: we subtract the mean of a variable from each value and divide by the standard deviation, which is the same as calculating a z-score. This sets the mean of the variable to zero and the standard deviation to one.

Standardizing variables offers two major advantages that are relevant to this class. First, sometimes variables are just difficult to interpret. Have you ever been asked whether you agree or disagree with a statement on a scale of 1-7 (or 1-5, or 0-1000)? What would it mean to say that the difference between two groups is 0.24 on that 1-7 scale? Often, psychologists and others standardize variables so that differences have a more straightforward interpretation. This interpretation is that a difference is some fraction of a standard deviation. When analyzing the results of an experiment, a psychologist might say that the treatment has an effect on the outcome of 0.4 standard deviations. That's pretty opaque–but many people find it more intuitive than saying the effect is 0.24 out of seven.

The second and most immediately obvious advantage is that we can plot standardized variables at the same time. Examine the two plots below. The first plot includes unstandardized variables. If we were to look at any one of these variables, we may see more ups and downs, but the additional variables change the scale of the y-axis. This means we have zoomed out and the ups and downs are more difficult to see. (Whether the ups and downs matter is another question, and we'll talk more about this!)

[46]:
```
df[["dayofweek", "money", "work", "love", "nervousness", "alcohol"]].
 ↪groupby("dayofweek").mean(["money", "work", "love", "nervousness",␣
 ↪"alcohol"]).reindex(daysofweek).plot(kind="line")
plt.xlabel("Day of Week")
plt.xticks(list(range(7)), labels=daysofweek, rotation=30)
plt.ylabel("Average")
plt.title("Unstandardized Measures of Posts\nto r/offmychest in April, 2020")
plt.show()
```

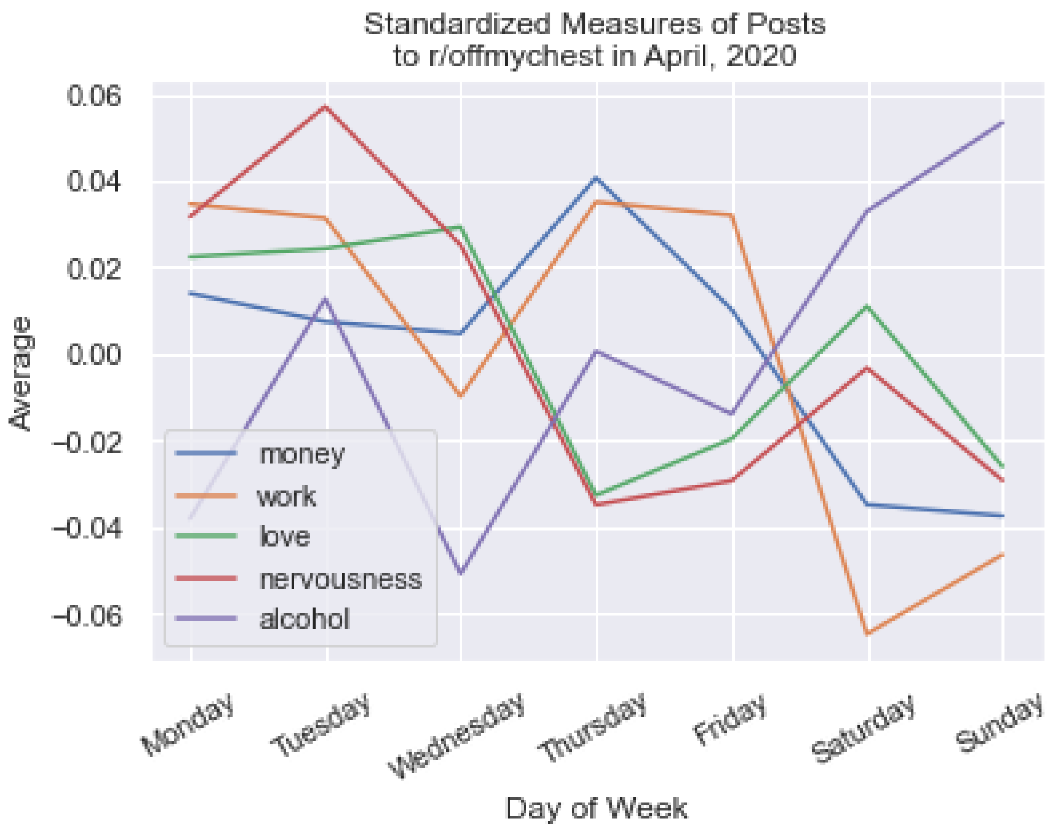


These look pretty flat. Now let's standardize them and see how things change.

[47]:
```
def standardize(series):
    """
    Mean-centers a variable and divides by the standard deviation,
    resulting in a mean of zero and a standard deviation of one.
    This puts variables on the same scale and can make comparisons
    a bit easier to see.
    """
    mu = np.mean(series.values)
    std = np.std(series.values, ddof=1)
    series = [(val-mu)/std for val in series.values]
    return series
```

[48]:
```
for cat in lexicon.cats.keys():
    df[cat] = standardize(df[cat])
```

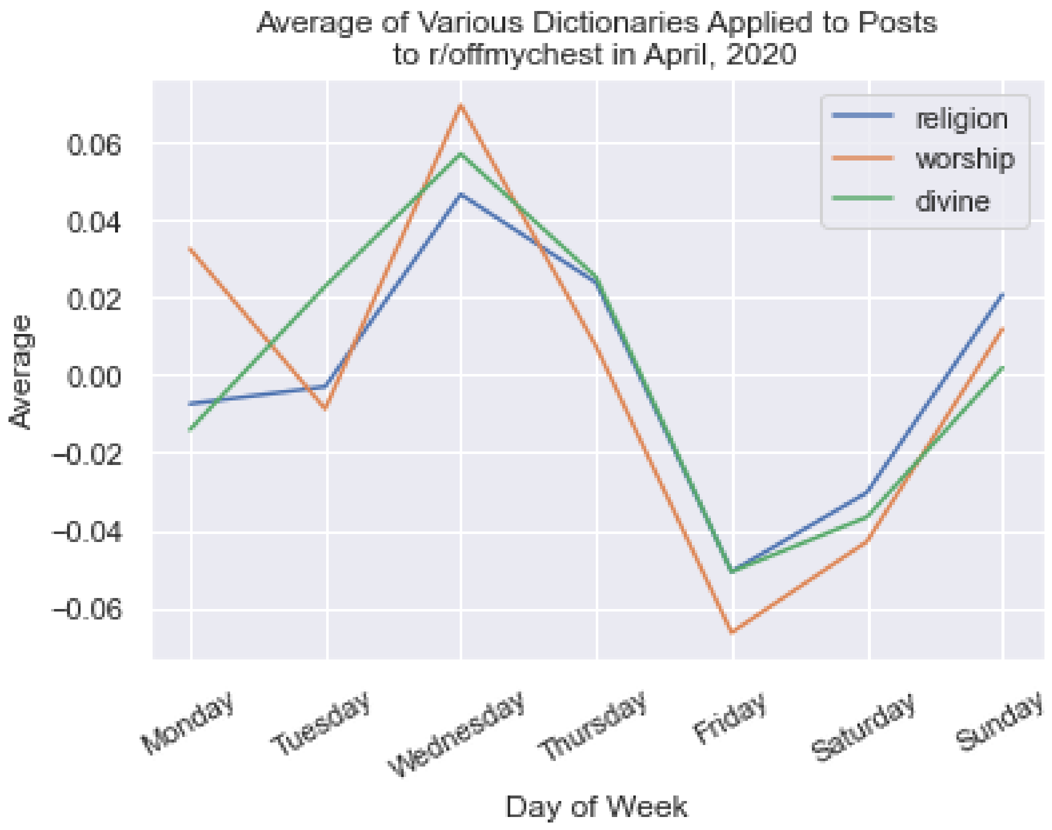
[49]:
```
df[["dayofweek", "money", "work", "love", "nervousness", "alcohol"]].
 ↪groupby("dayofweek").mean(["money", "work", "love", "nervousness",␣
 ↪"alcohol"]).reindex(daysofweek).plot(kind="line")
plt.xlabel("Day of Week")
plt.xticks(list(range(7)), labels=daysofweek, rotation=30)
plt.ylabel("Average")
plt.title("Standardized Measures of Posts\nto r/offmychest in April, 2020")
plt.show()
```



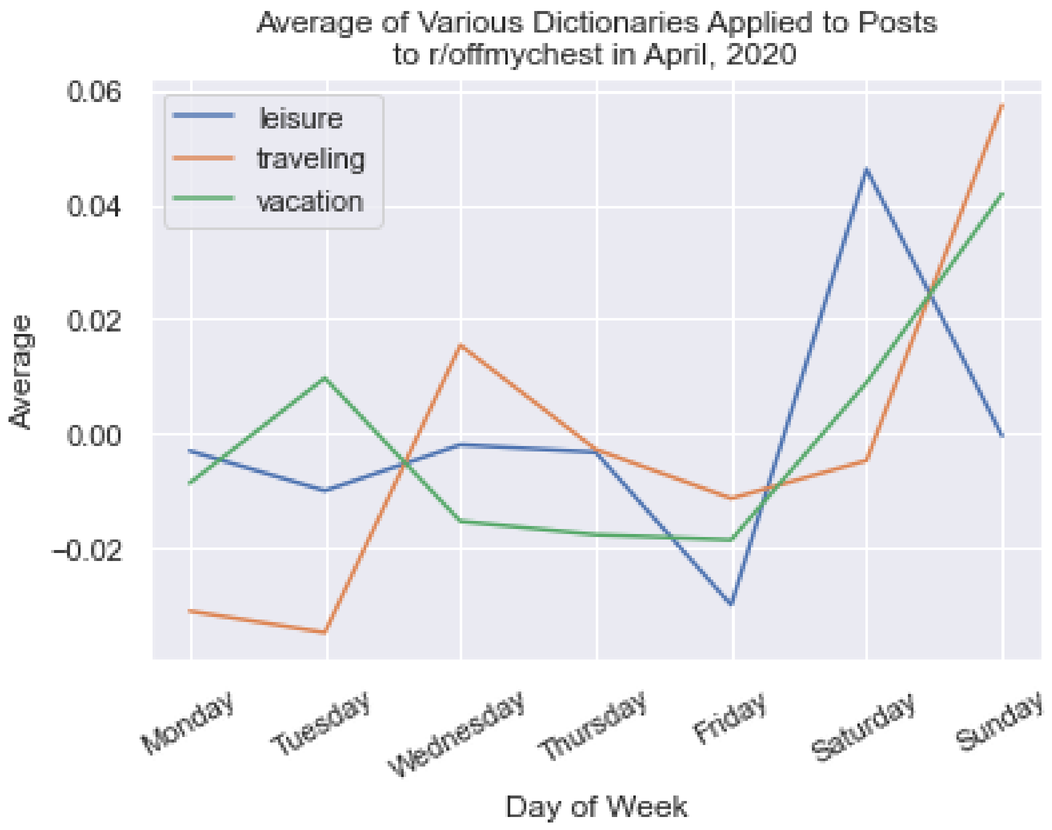
This plot is a mess, but we see a lot of change from day to day!

0.2.2 Do people talk about different things at different points in the week?

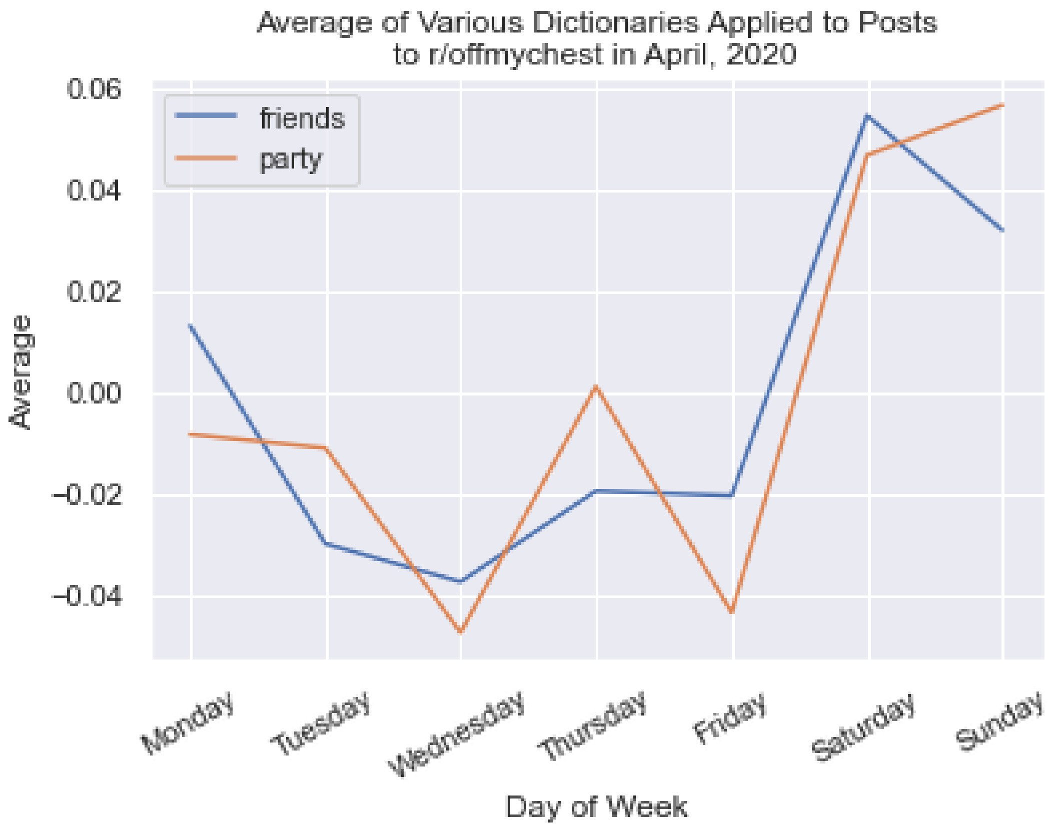
```
[71]: df[["dayofweek"] + ["religion", "worship", "divine"]].groupby("dayofweek").
       ↪mean(["religion", "worship", "divine"]).reindex(daysofweek).plot(kind="line")
      plt.xlabel("Day of Week")
      plt.xticks(list(range(7)), labels=daysofweek, rotation=30)
      plt.ylabel("Average")
      plt.title("Average of Various Dictionaries Applied to Posts\nto r/offmychest in␣
       ↪April, 2020")
      plt.show()
```



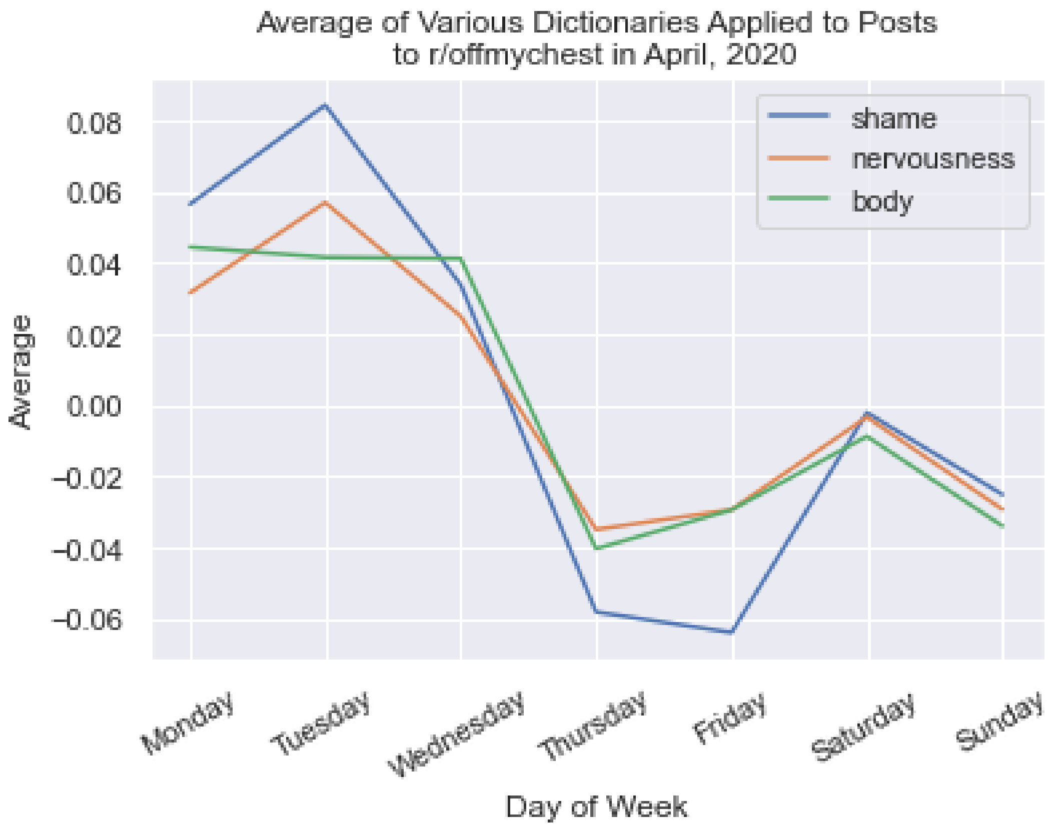
```
[51]: df[["dayofweek"] + ["leisure", "traveling", "vacation"]].groupby("dayofweek").
       ↪mean(["leisure", "traveling", "vacation"]).reindex(daysofweek).
       ↪plot(kind="line")
      plt.xlabel("Day of Week")
      plt.xticks(list(range(7)), labels=daysofweek, rotation=30)
      plt.ylabel("Average")
      plt.title("Average of Various Dictionaries Applied to Posts\nto r/offmychest in␣
       ↪April, 2020")
      plt.show()
```



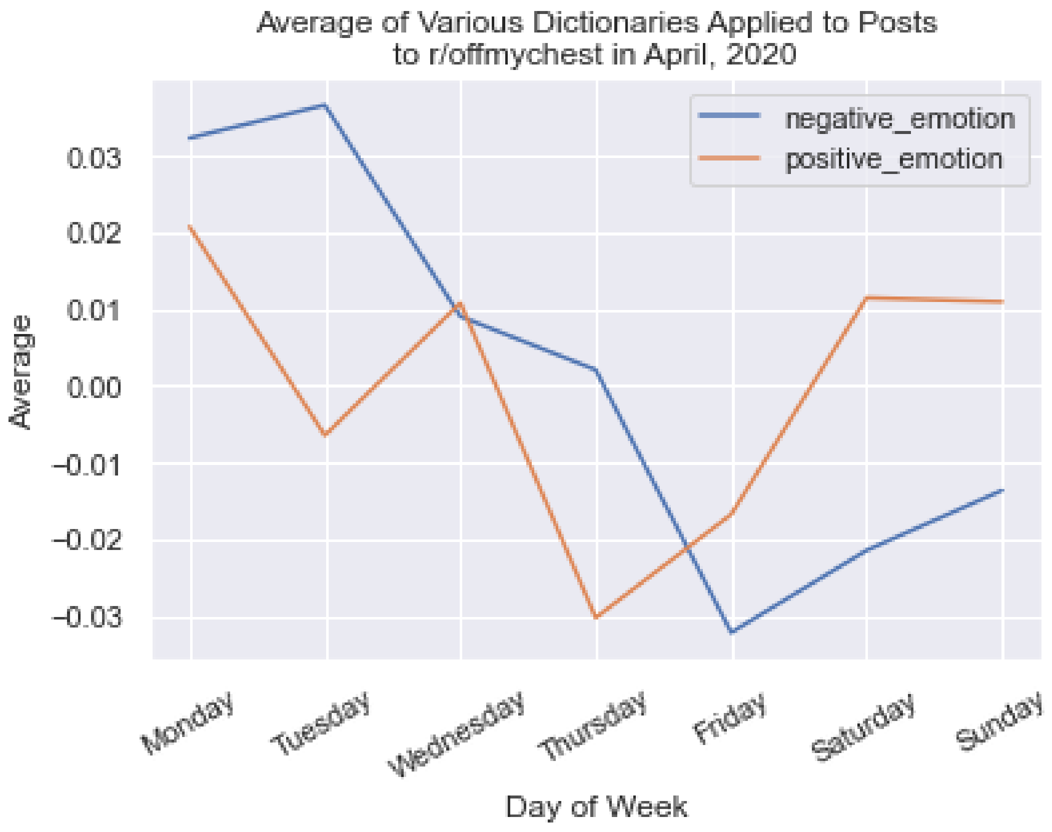
[52]:
```
df[["dayofweek"] + ["friends", "party"]].groupby("dayofweek").mean(["friends",␣
 ↪"party"]).reindex(daysofweek).plot(kind="line")
plt.xlabel("Day of Week")
plt.xticks(list(range(7)), labels=daysofweek, rotation=30)
plt.ylabel("Average")
plt.title("Average of Various Dictionaries Applied to Posts\nto r/offmychest in␣
 ↪April, 2020")
plt.show()
```



```
[53]: df[["dayofweek"] + ["shame", "nervousness", "body"]].groupby("dayofweek").
       ↪mean(["shame", "nervousness", "body"]).reindex(daysofweek).plot(kind="line")
      plt.xlabel("Day of Week")
      plt.xticks(list(range(7)), labels=daysofweek, rotation=30)
      plt.ylabel("Average")
      plt.title("Average of Various Dictionaries Applied to Posts\nto r/offmychest in␣
       ↪April, 2020")
      plt.show()
```

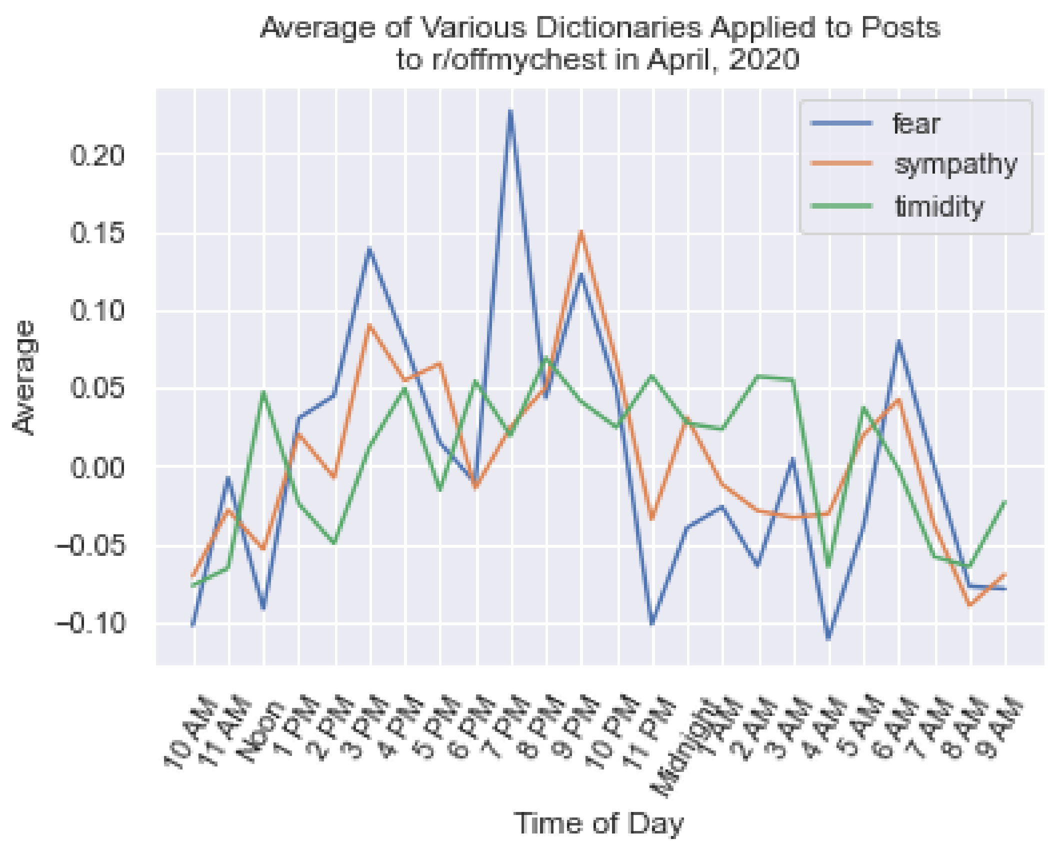


[54]:
```
df[["dayofweek"] + ["negative_emotion", "positive_emotion"]].
 ↪groupby("dayofweek").mean(["negative_emotion", "positive_emotion"]).
 ↪reindex(daysofweek).plot(kind="line")
plt.xlabel("Day of Week")
plt.xticks(list(range(7)), labels=daysofweek, rotation=30)
plt.ylabel("Average")
plt.title("Average of Various Dictionaries Applied to Posts\nto r/offmychest in␣
 ↪April, 2020")
plt.show()
```

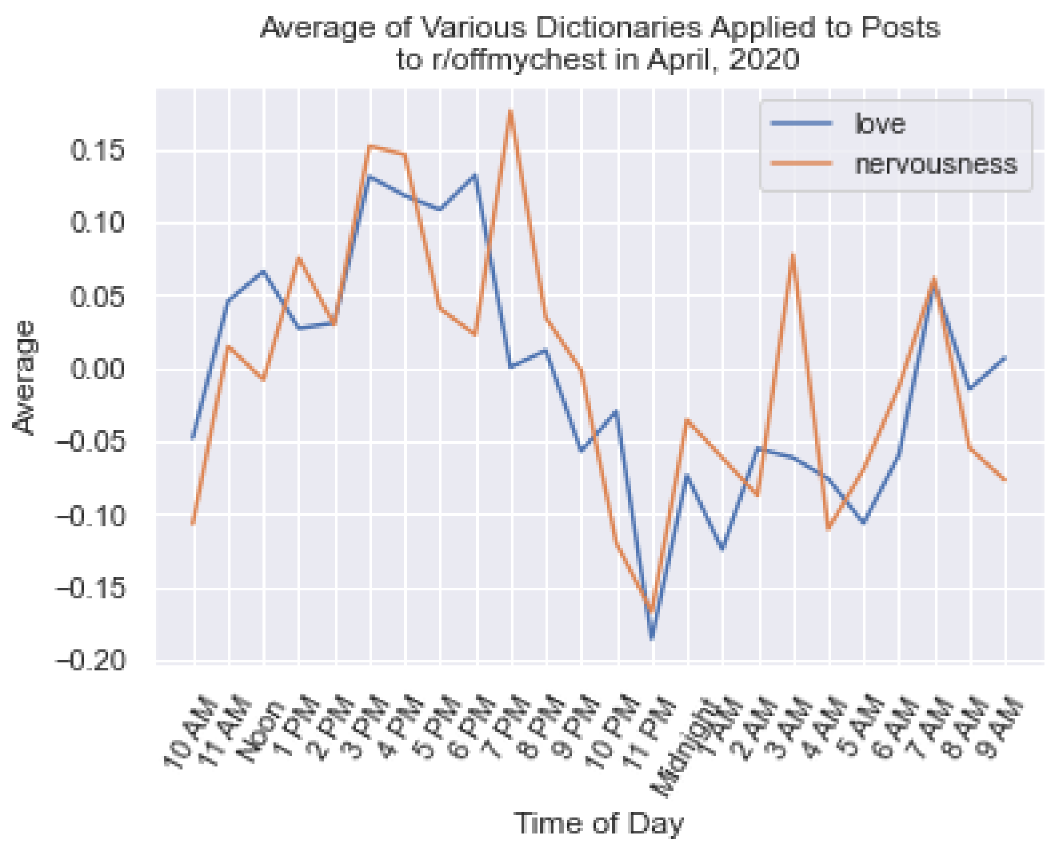


0.2.3 Do people talk about different things at different points in the day?

```
[55]: cats = ["fear", "sympathy", "timidity"]
      df[["hour"] + cats].groupby("hour").mean(cats).plot(kind="line")
      plt.xlabel("Time of Day")
      plt.xticks(list(range(24)), labels=hours, rotation=60)
      plt.ylabel("Average")
      plt.title("Average of Various Dictionaries Applied to Posts\nto r/offmychest in␣
       ↪April, 2020")
      plt.show()
```



[56]:
```
cats = ["love", "nervousness"]
df[["hour"] + cats].groupby("hour").mean(cats).plot(kind="line")
plt.xlabel("Time of Day")
plt.xticks(list(range(24)), labels=hours, rotation=60)
plt.ylabel("Average")
plt.title("Average of Various Dictionaries Applied to Posts\nto r/offmychest in␣
 ↪April, 2020")
plt.show()
```



0.2.4 What about posts early versus late in the day?

Let's create a variable to distinguish posts submitted early in the day from posts submitted late in the day.

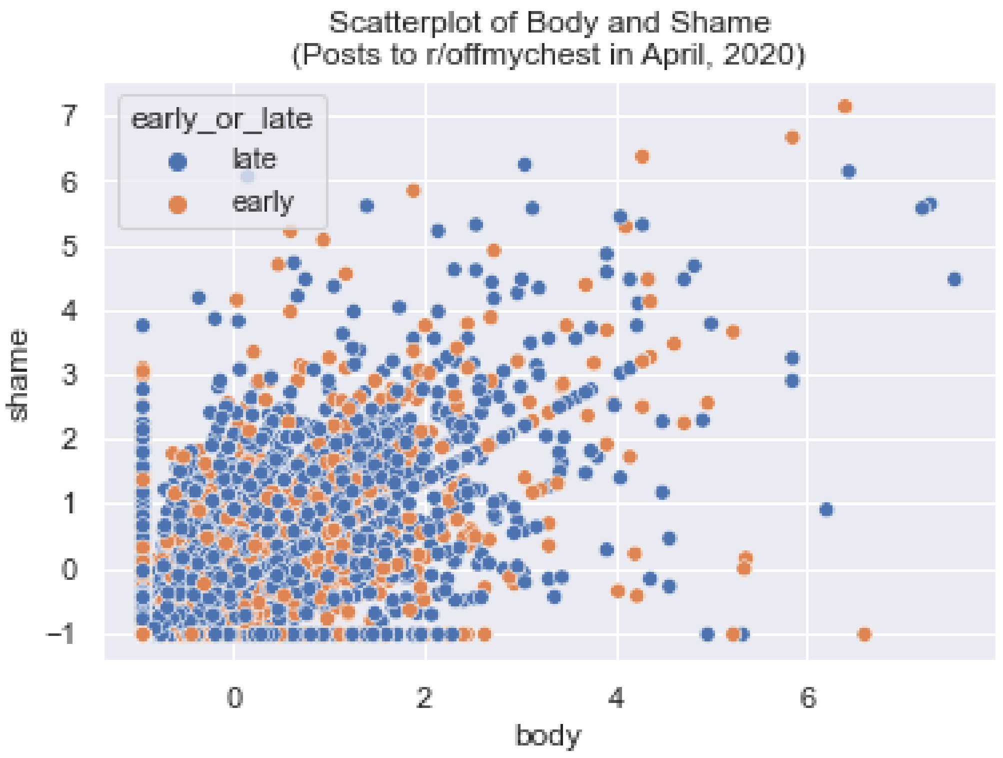
[57]:
```python
def time_of_day(t):
    if t in list(range(10,20)):
        return "early"
    elif t in list(range(22,24)) + list(range(0,8)):
        return "late"
    else:
        return "NA"

df["early_or_late"] = [time_of_day(t) for t in df["hour"].values]
```

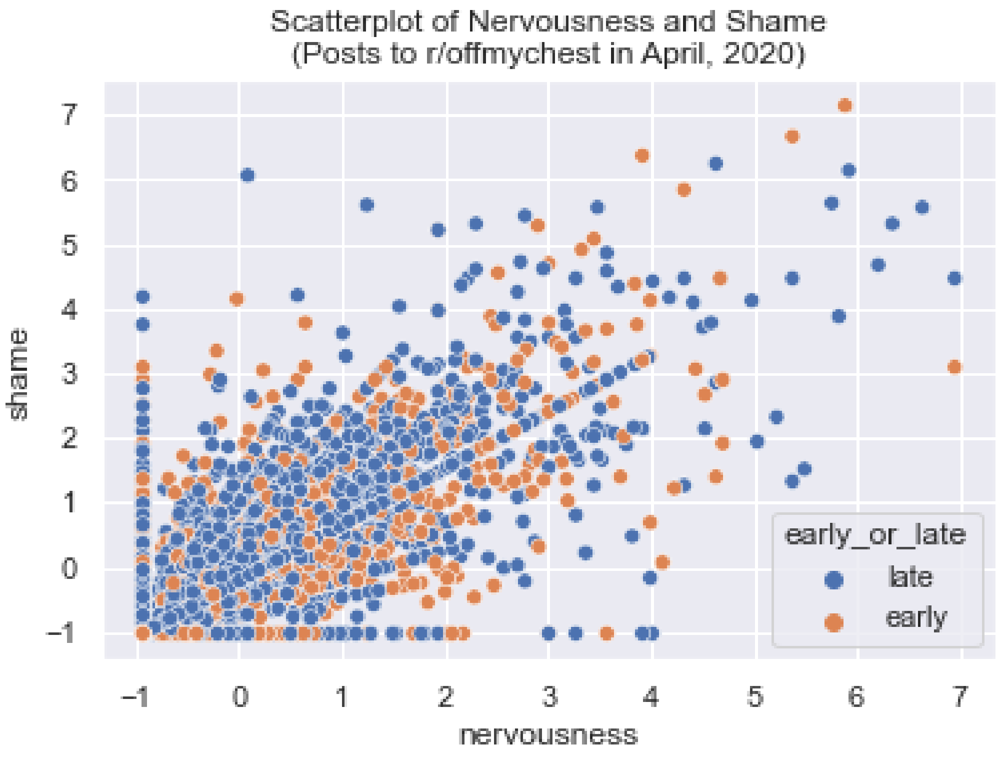
We can use this variable to directly compare the two groups, but we can also combine this with other plots.

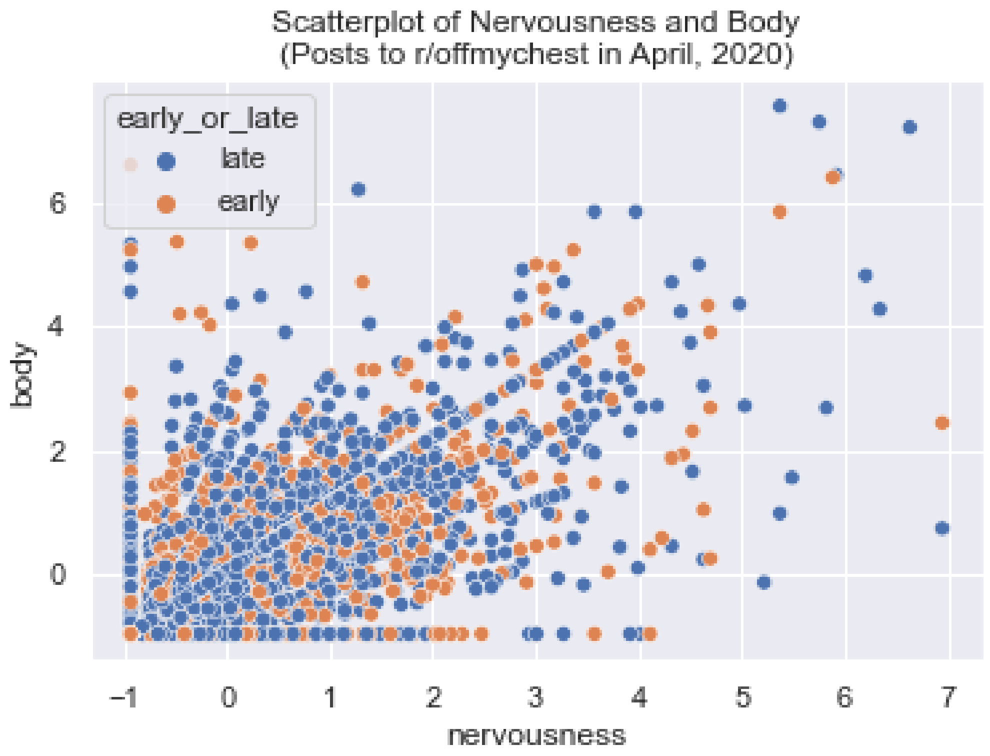
[58]:
```python
cat1 = "body"
cat2 = "shame"
```

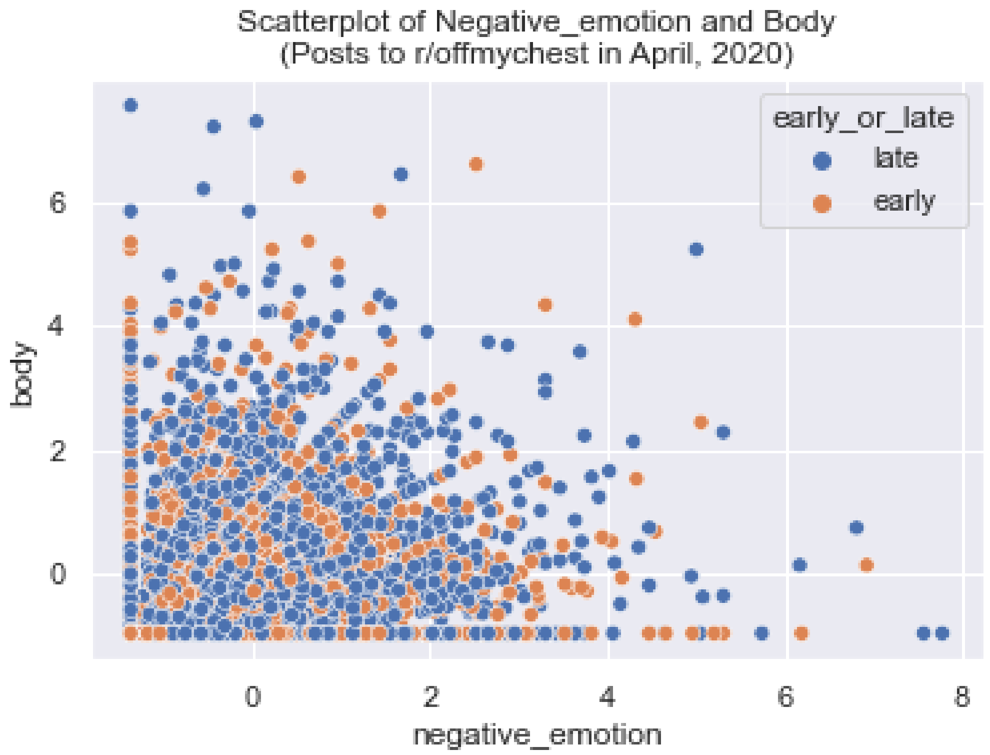
```
sns.scatterplot(x=cat1, y=cat2, data=df[df["early_or_late"]!="NA"],␣
↪hue="early_or_late")
plt.title(f"Scatterplot of {cat1.capitalize()} and {cat2.capitalize()}\n(Posts␣
↪to r/offmychest in April, 2020)")
plt.show()

cat1 = "nervousness"
cat2 = "shame"
sns.scatterplot(x=cat1, y=cat2, data=df[df["early_or_late"]!="NA"],␣
↪hue="early_or_late")
plt.title(f"Scatterplot of {cat1.capitalize()} and {cat2.capitalize()}\n(Posts␣
↪to r/offmychest in April, 2020)")
plt.show()

cat1 = "nervousness"
cat2 = "body"
sns.scatterplot(x=cat1, y=cat2, data=df[df["early_or_late"]!="NA"],␣
↪hue="early_or_late")
plt.title(f"Scatterplot of {cat1.capitalize()} and {cat2.capitalize()}\n(Posts␣
↪to r/offmychest in April, 2020)")
plt.show()

cat1 = "negative_emotion"
cat2 = "body"
sns.scatterplot(x=cat1, y=cat2, data=df[df["early_or_late"]!="NA"],␣
↪hue="early_or_late")
plt.title(f"Scatterplot of {cat1.capitalize()} and {cat2.capitalize()}\n(Posts␣
↪to r/offmychest in April, 2020)")
plt.show()

cat1 = "friends"
cat2 = "positive_emotion"
sns.scatterplot(x=cat1, y=cat2, data=df[df["early_or_late"]!="NA"],␣
↪hue="early_or_late")
plt.title(f"Scatterplot of {cat1.capitalize()} and {cat2.capitalize()}\n(Posts␣
↪to r/offmychest in April, 2020)")
plt.show()

cat1 = "optimism"
cat2 = "positive_emotion"
sns.scatterplot(x=cat1, y=cat2, data=df[df["early_or_late"]!="NA"],␣
↪hue="early_or_late")
plt.title(f"Scatterplot of {cat1.capitalize()} and {cat2.capitalize()}\n(Posts␣
↪to r/offmychest in April, 2020)")
plt.show()
```

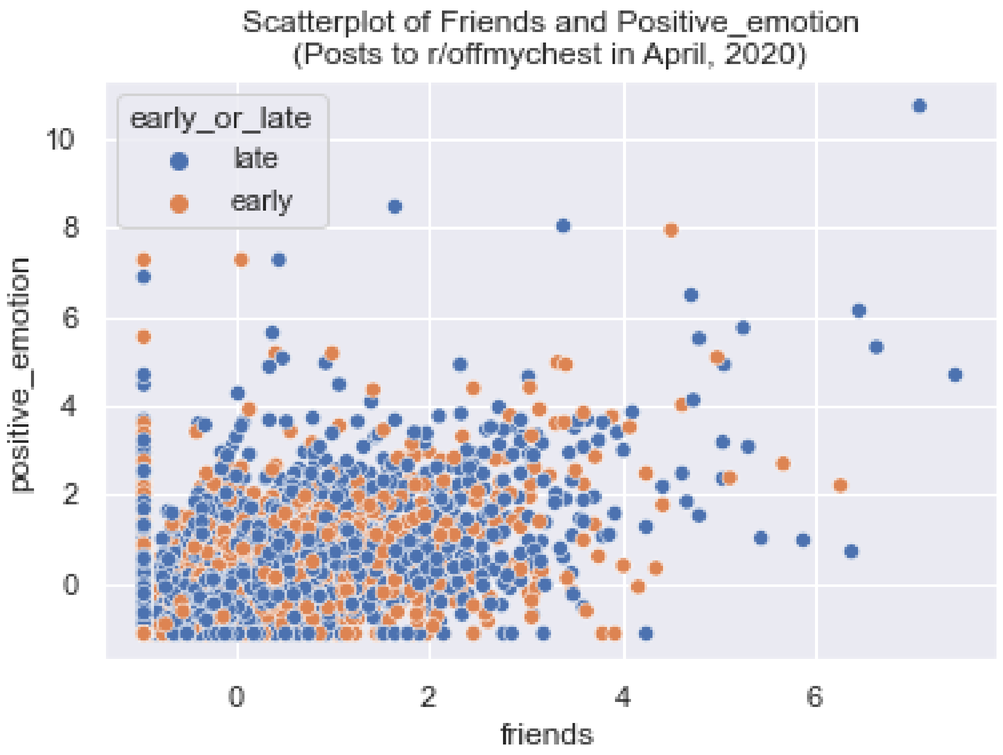
```
cat1 = "nervousness"
cat2 = "love"
sns.scatterplot(x=cat1, y=cat2, data=df[df["early_or_late"]!="NA"],␣
 ↪hue="early_or_late")
plt.title(f"Scatterplot of {cat1.capitalize()} and {cat2.capitalize()}\n(Posts␣
 ↪to r/offmychest in April, 2020)")
plt.show()

cat1 = "blue_collar_job"
cat2 = "poor"
sns.scatterplot(x=cat1, y=cat2, data=df[df["early_or_late"]!="NA"],␣
 ↪hue="early_or_late")
plt.title(f"Scatterplot of {cat1.capitalize()} and {cat2.capitalize()}\n(Posts␣
 ↪to r/offmychest in April, 2020)")
plt.show()

cat1 = "domestic_work"
cat2 = "negative_emotion"
sns.scatterplot(x=cat1, y=cat2, data=df[df["early_or_late"]!="NA"],␣
 ↪hue="early_or_late")
plt.title(f"Scatterplot of {cat1.capitalize()} and {cat2.capitalize()}\n(Posts␣
 ↪to r/offmychest in April, 2020)")
plt.show()

cat1 = "white_collar_job"
cat2 = "occupation"
sns.scatterplot(x=cat1, y=cat2, data=df[df["early_or_late"]!="NA"],␣
 ↪hue="early_or_late")
plt.title(f"Scatterplot of {cat1.capitalize()} and {cat2.capitalize()}\n(Posts␣
 ↪to r/offmychest in April, 2020)")
plt.show()
```

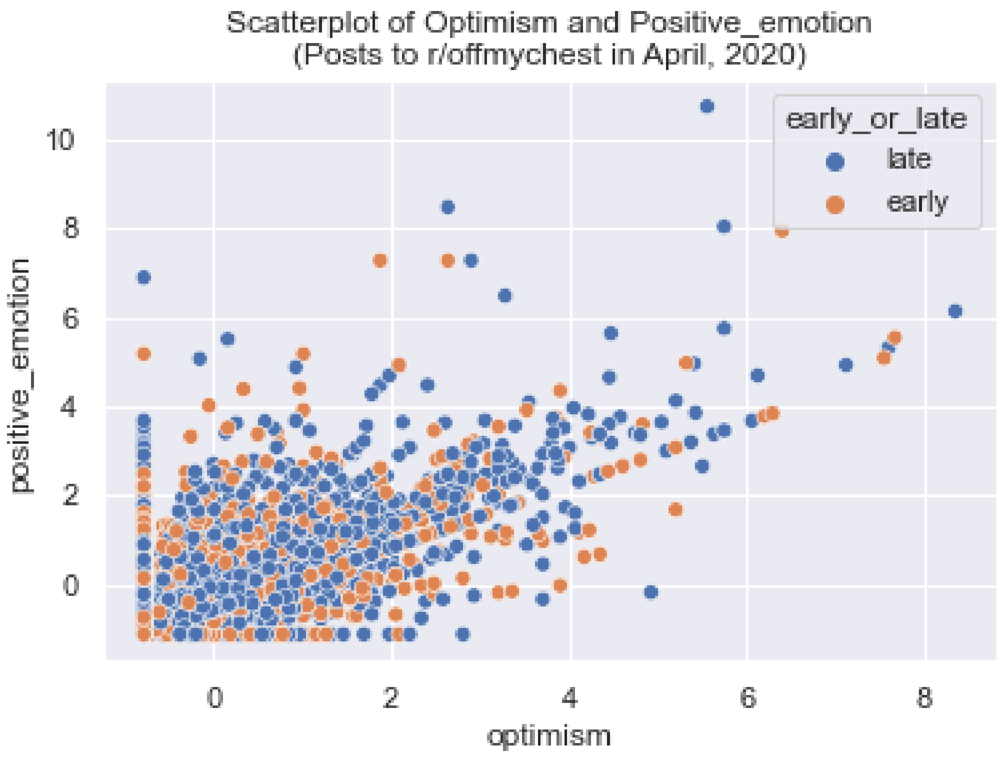

Scatterplot of Body and Shame
(Posts to r/offmychest in April, 2020)
early_or_late
late
early
shame
body
7
6
5
4
3
2
1
0
−1
0
2
4
6

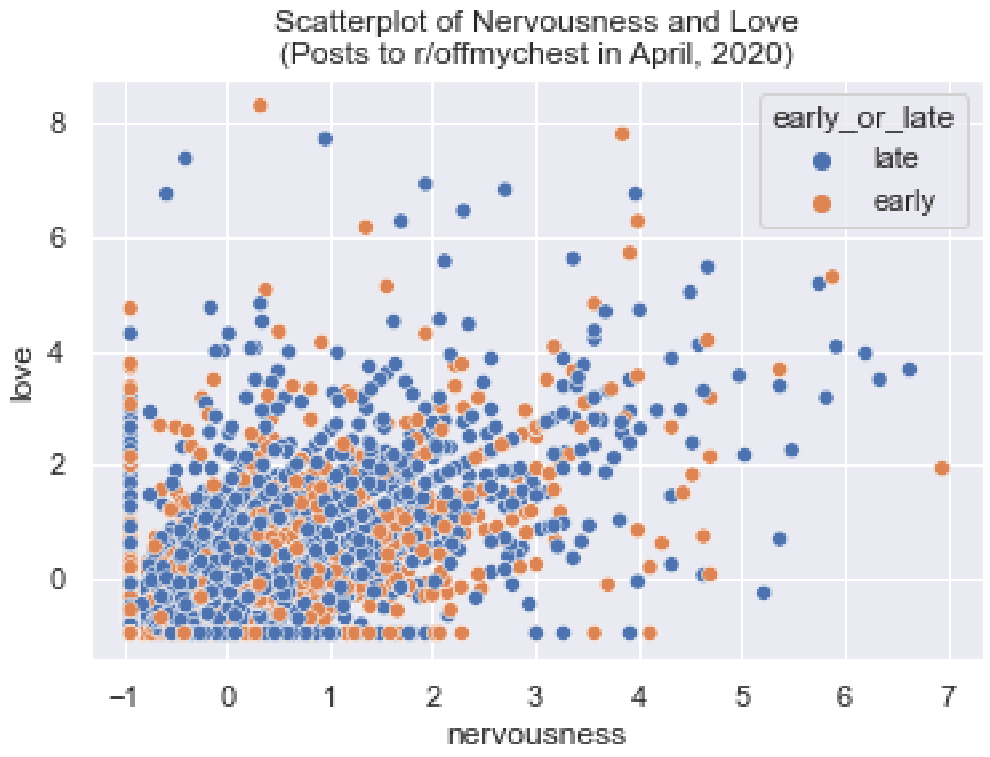

Scatterplot of Nervousness and Shame
(Posts to r/offmychest in April, 2020)
early_or_late
late
early
shame
nervousness
7
6
5
4
3
2
1
0
−1
−1
0
1
2
3
4
5
6
7

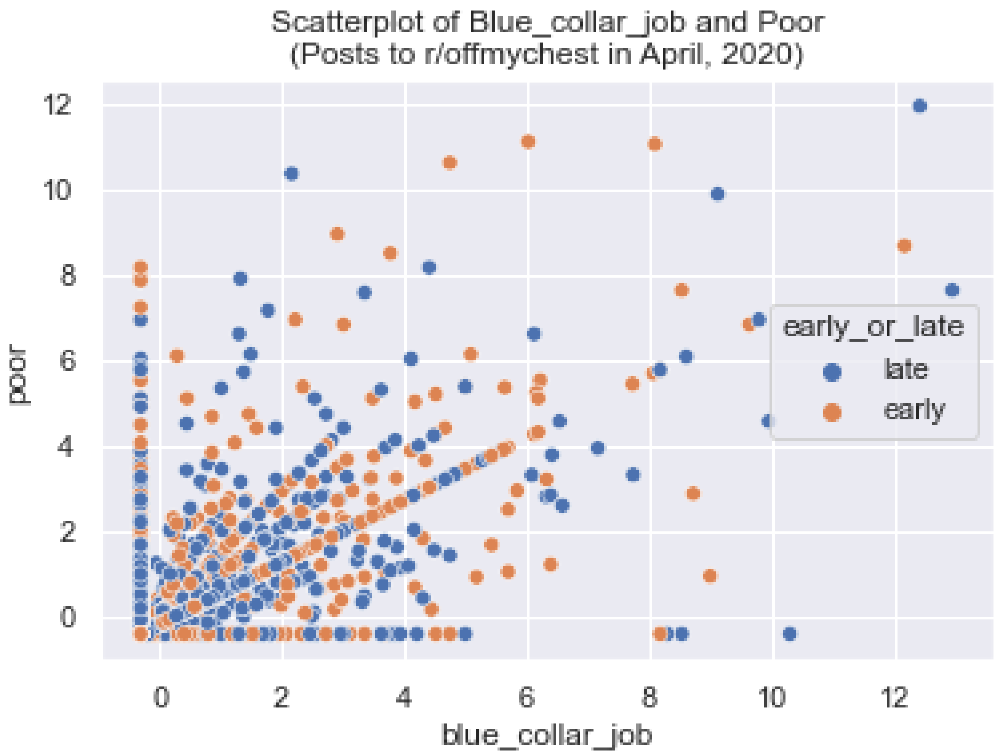

Scatterplot of Nervousness and Body
(Posts to r/offmychest in April, 2020)
early_or_late
late
early
body
nervousness
-1
0
1
2
3
4
5
6
7

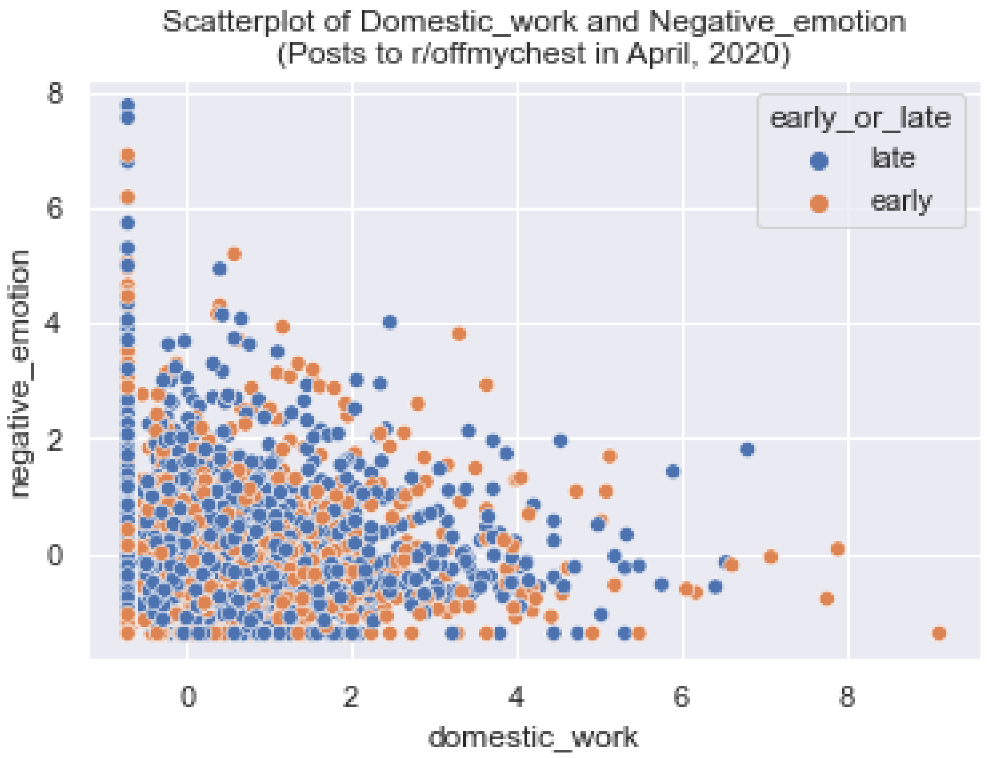

Scatterplot of Negative_emotion and Body
(Posts to r/offmychest in April, 2020)
early_or_late
late
early
body
negative_emotion
0
2
4
6
8

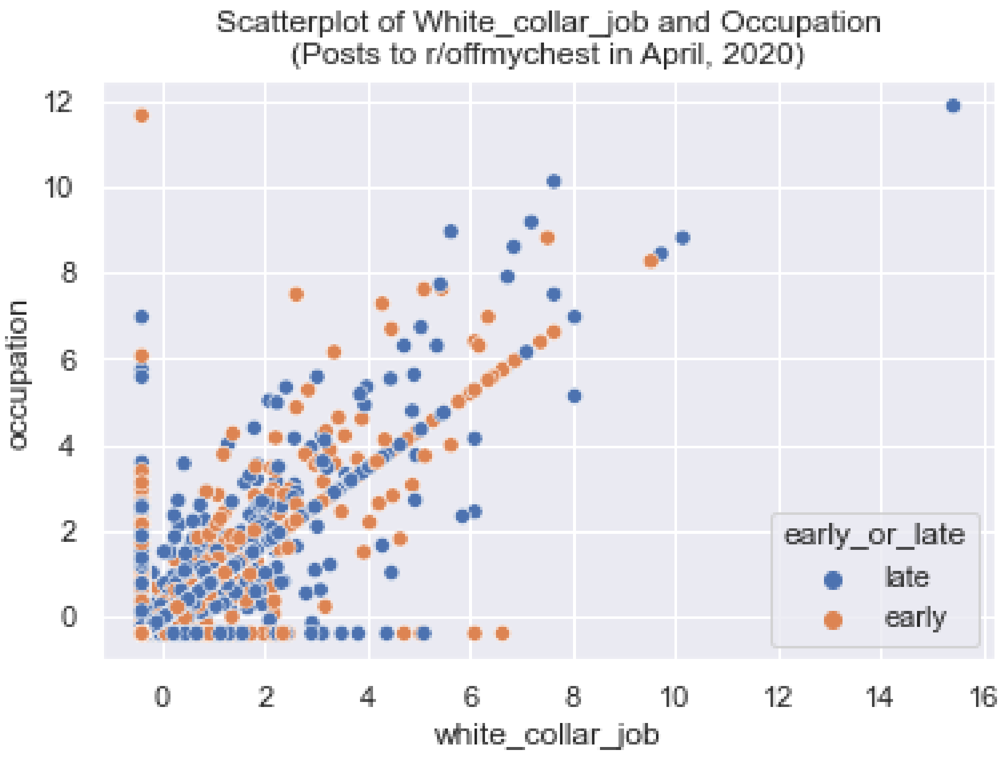

Scatterplot of Friends and Positive_emotion
(Posts to r/offmychest in April, 2020)
early_or_late
late
early
positive_emotion
friends
10
8
6
4
2
0
0
2
4
6


Scatterplot of Optimism and Positive_emotion
(Posts to r/offmychest in April, 2020)
early_or_late
late
early
positive_emotion
optimism
10
8
6
4
2
0
0
2
4
6
8


Scatterplot of Nervousness and Love
(Posts to r/offmychest in April, 2020)
early_or_late
late
early
love
nervousness
8
6
4
2
0
−1
0
1
2
3
4
5
6
7


Scatterplot of Blue_collar_job and Poor
(Posts to r/offmychest in April, 2020)
poor
blue_collar_job
early_or_late
late
early
0
2
4
6
8
10
12


Scatterplot of Domestic_work and Negative_emotion
(Posts to r/offmychest in April, 2020)
negative_emotion
domestic_work
early_or_late
late
early
0
2
4
6
8



Notice that some of the scatterplots have pretty defined straight lines! The lexicons provided for using dictionary methods are not necessarily mutually exclusive. In the last plot, which shows the relationship between `white_collar_job` and `occupation`, we see one of these really defined lines. We might expect a strong association between the two, anyway, right? Let's check whether there's anything strange going on.

Remember that each dictionary or lexicon is just a list of words. The line of code below looks at the lists of words for `white_collar_job` and `occupation` and shows the words they have in common.

As it turns out, they have quite a few words in common! Each time a word like "job" or "lawyer" appears in a document, taking this dictionary approach means we make the assumption that document is a bit about each of these categories.

```
[59]: set(lexicon.cats["occupation"]).intersection(lexicon.cats["white_collar_job"])
```

```
[59]: {'accountant',
       'actor',
       'analyst',
       'attorney',
       'businesswoman',
       'chemist',
 'colleague',
 'consultant',
 'dentist',
 'detective',
 'doctor',
 'employment',
 'entrepreneur',
 'executive',
 'gynecologist',
 'hire',
 'inspector',
 'internship',
 'job',
 'lawyer',
 'manager',
 'neurologist',
 'nurse',
 'office',
 'pediatric',
 'pediatrician',
 'pharmacist',
 'photographer',
 'physician',
 'politician',
 'profession',
 'professional',
 'psychiatrist',
 'psychologist',
 'qualified',
 'retire',
 'retired',
 'salary',
 'senator',
 'specialist',
 'supervisor',
 'surgeon',
 'therapist',
 'veterinarian',
 'workaholic'}
```

Some of these words appear elsewhere. For example, the word "job" appears in lexicons for categories like `blue_collar_job` and `poor`.

```
[60]: set(lexicon.cats["blue_collar_job"]).intersection(lexicon.cats["poor"])
```

```
[60]: {'job'}
```

0.3 Exercises

First, pick a social problem you would like to explore using this corpus. In Exercise 1, you will create a lexicon for that topic in only three lines of code. You will use your lexicon for the remaining exercises.

`empath` makes creating lexicons fairly straightforward on the surface. Beneath the surface, it uses some pretty powerful tools. We'll learn about these tools (such as word embeddings) later in the class. For now, all we need to know is that we can provide a small list of words related to our chosen topic and `empath` will give us a longer list of related words. We can use that list as our lexicon.

For example, let's say we want to create a lexicon related to stigma. We can seed (start) our lexicon using a handful of words. Let's try these:

[111]:
```
stigma_words = ["stigma", "taboo", "avoid", "aversion", "stereotype", "trash"]
```

Now we can create our lexicon using the following line of code. The first part ("stigma") is what we are naming the category. The second part (stigma_words) is the list we created in the line above). The model="reddit" argument means we will find words that are related based on how language is (was) used in Reddit data when `empath` was created. The alternatives are based on the New York Times and a collection of fiction.

Note: we will talk a LOT more about word embeddings and vector space models in the future, and you will learn how they are used to generate collections of words like the following. The TL;DR is that these tools assume words are "similar" if they are used in similar contexts. This assumption can make it easy to find synonyms, but sometimes tools based on this assumption will think *antonyms* are highly "similar" in this specific sense. These tools can also reflect social prejudices, which can make them useful for studying those issues (as we will later in the course) but may be a bit of a surprise. Debiasing word embeddings and other tools is also an active area of research, and we will discuss that more.

In other words, whatever shows up in the lexicon you create, it shouldn't be seen as a normative claim. You may see examples of prejudice inherent to the language used on Reddit (or in fiction or the New York Times), but that doesn't mean that a word that shows up should be seen as part of the reality of your topic.

[112]:
```
lexicon.create_category("stigma", stigma_words, model="reddit", size=300)
```

```
["stigma", "negative_thing", "social_stigma", "taboo", "negative_stigma",
"aversion", "stigmas", "negative_connotations", "weird_idea",
"negative_connotation", "stereotype", "negative_association", "positive_thing",
"taboo", "stigmatized", "social_pressure", "bad_thing", "stigmatization",
"negative_associations", "whole_culture", "negative_attitudes", "fetishization",
"stereotypes", "negative_view", "social_taboo", "cultural_norm", "taboo",
"huge_stigma", "fetishizing", "stigmatize", "irrational_fear", "ironically",
"normal_thing", "certain_people", "stupid_stereotypes", "stigma",
"negative_stereotypes", "natural_thing", "obsession", "preconception",
"fetishize", "shunning", "negative_things", "serious_problem", "social_norm",
"skewed_view", "prudishness", "negative_aspects", "real_problem",
"ridiculous_notion", "stereotyping", "demeaning", "double_standard",
"real_issue", "negative_stereotype", "gay_culture", "disdain", "social_stigmas",
"typical_thing", "negative_perception", "good_thing", "phobia",
"knee_jerk_reaction", "inherently_bad_thing", "demonization", "perpetuates",
"serious_issue", "connotations", "trendy_thing", "implication", "distaste",
"victim_mentality", "associate", "double-standard", "just_a_thing", "dislike",
"common_thing", "shun", "insinuation", "phobic", "fetishism", "stigmatised",
"legitimate_thing", "legitimate_problem", "prejudice", "negative_context",
"taboo_subject", "common_response", "oversensitivity", "common_perception",
"negative_sense", "social_pressures", "mentality", "slut_shaming",
"cultural_stigma", "fixation", "automatic_assumption", "undesirable",
"legitimate_concern", "social_standard", "pervasive", "general_attitude",
"very_common_thing", "emasculation", "strong_aversion", "extreme_way",
"shameful", "social_taboos", "warped_view", "underlying_reason",
"bad_connotation", "negative_aspect", "stigmatizing", "oversensitive",
"shaming", "fetishizing", "fetishized", "many_people", "abhor",
"negative_trait", "general", "negative_feeling", "stigmatized", "misconception",
"social_acceptance", "growing_trend", "negative_implications", "snobbery",
"shaming", "harmful_stereotypes", "knee-jerk_reaction", "weird_ideas",
"whole_other_thing", "subculture", "current_culture", "bad_kind",
"common_theme", "gayness", "stereo_type", "denigration", "detest",
"stereotyped", "mock_people", "cultural_stereotypes", "extreme_degree", "norm",
"fetishizes", "superiority_complex", "just_an_excuse", "just_women",
"special_snowflake_syndrome", "reinforces", "unmanly", "sterotype", "elitism",
"euphemisms", "taboo_thing", "asshole_behavior", "cultural_pressure",
"taboo_topic", "stupid_stereotype", "cultural_taboo", "mainstream_society",
"homophobia", "bigger_problem", "just_ignorance", "insist", "negative_views",
"sexual_repression", "excuse", "sexual_promiscuity", "romanticizing",
"internalized_homophobia", "promiscuity", "mindset", "degrading",
"western_culture", "misguided_notion", "perpetuating", "far_too_many_people",
"associating", "whole_stigma", "attitude", "double_standards", "predilection",
"demonizing", "certain_stigma", "superficiality", "ridiculous_idea",
"just_a_sign", "socially_acceptable_way", "common_sentiment",
"defense_mechanism", "todays_society", "common_idea", "disdain",
"insulting_way", "common_stereotype", "percieved", "unpleasant_things",
"certain_stereotypes", "demeans", "huge_taboo", "general_sense", "ostracizing",
"common_reaction", "vilification", "unfortunate_reality", "misogyny",
"male_culture", "glamorizing", "general_perception", "internalized_misogyny",
"western_cultures", "misguided_idea", "transexuals", "stereotypically",
"other_reason", "societal_pressures", "whole_notion", "much_bigger_problem",
"same_mentality", "such_extremes", "visceral_reaction", "pure_ignorance",
"big_problem", "flippantly", "huge_double_standard", "political_incorrectness",
"problematic", "convenient_excuse", "bad_connotations", "pretty_common_thing",
"perpetuate", "unhealthy_obsession", "&gt;A_lot", "so_much_stigma",
"general_mindset", "less_stigma", "connotation", "general_society",
"social_perception", "accepted", "misguidedly", "masculinity", "good_reason",
"misogynistic", "deeper_issue", "taboos", "demonizing", "terrible_thing",
"very_real_problem", "hatred", "transgenders", "wrong_kind", "certain_group",
"othering", "shitty_behaviour", "very_real_issue", "ignorantly",
"social_consequences", "Tumblrinas", "insinuate", "societal_norm",
"societal_pressure", "political_correctness", "bad_stereotypes", "bad_history",
"unmanly", "notion", "nasty_thing", "serious_thing", "sexual_frustration",
"bad_stigma", "mainstream_culture", "perpetuated", "insecurity",
"just_a_stereotype", "denigrating", "peer_pressure", "gay_community",
"hypersensitive", "old_stereotype", "shunned", "PC_culture", "stems",
"weird_obsession", "character_flaw"]
```

Also notice the n-grams connected by underscores (for example, "cultural_norm") and compound words (for example, "slut-shaming"). The corpus of posts from r/offmychest only has unigrams and has no punctuation. We'll cover those issues more later on in the course. For now, that just means there are words in the lexicon that won't appear in any of the documents in this corpus.

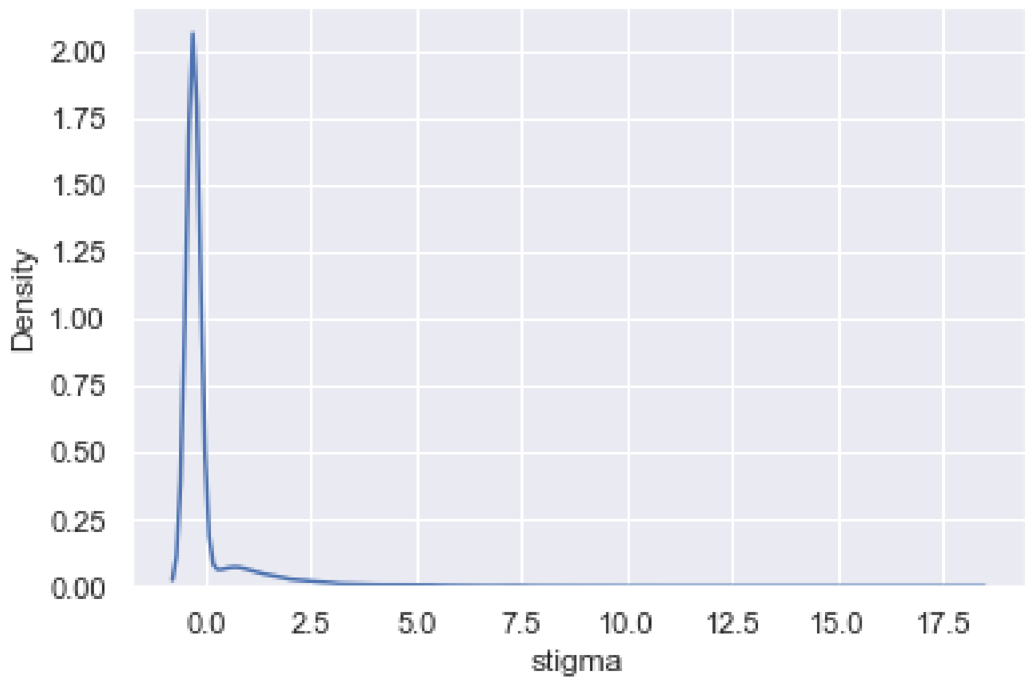
That aside, this custom lexicon can now be used just like those we've already used. The line of code below will add a variable to our dataframe. Be sure to include the normalize=True argument when you create your own in Exercise 1! Alternatively, if you'd prefer to use the raw counts, you should use the raw counts for the other lexical categories as well, as we initially did toward the beginning of this notebook. If you opt to use raw counts, please provide your rationale.

[119]:
```
df["stigma"] = [score_category(s, "stigma", normalize=True) for s in df["text"].
 ↪values]
df["stigma"] = standardize(df["stigma"])
```

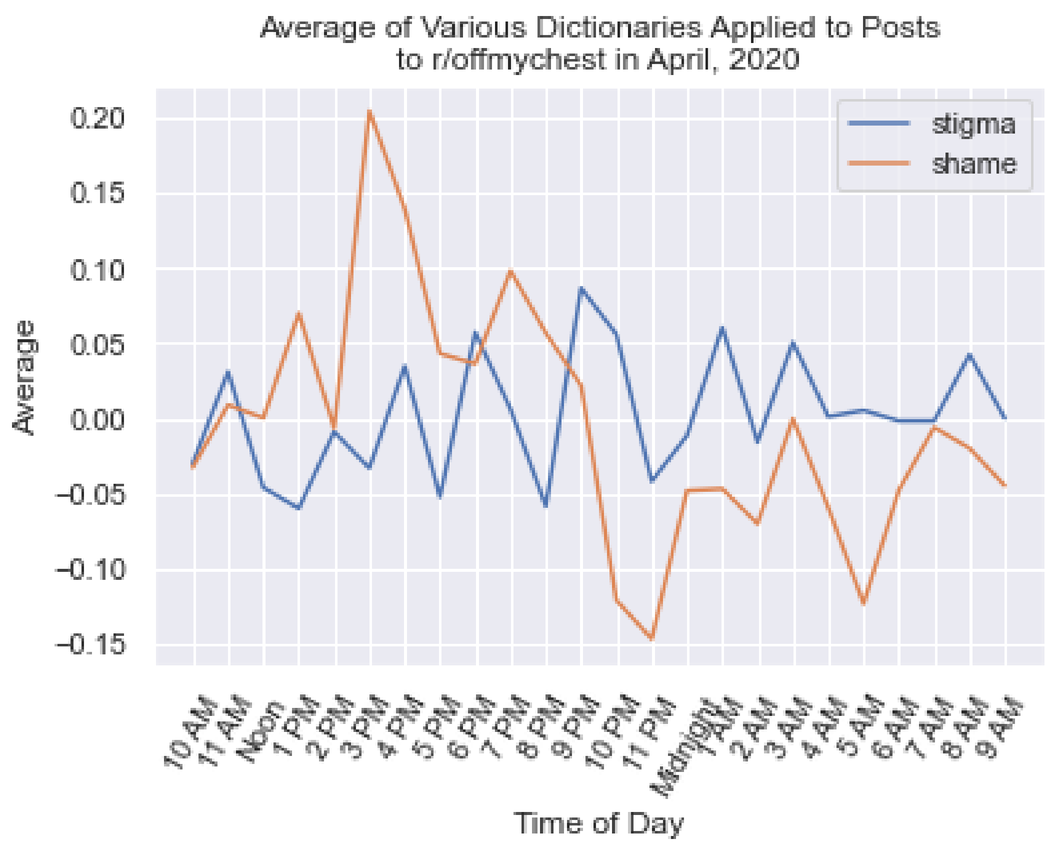
[126]:
```
df[["stigma", "racism", "shame"]].rcorr()
```

[126]:
```
        stigma  racism shame
stigma       -     ***
racism   0.457       -
shame   -0.006  -0.016     -
```

[121]:
```
sns.kdeplot("stigma", data=df)
plt.show()
```



```
[125]: cats = ["stigma", "shame"]
       df[["hour"] + cats].groupby("hour").mean(cats).plot(kind="line")
       plt.xlabel("Time of Day")
       plt.xticks(list(range(24)), labels=hours, rotation=60)
       plt.ylabel("Average")
       plt.title("Average of Various Dictionaries Applied to Posts\nto r/offmychest in␣
        ↪April, 2020")
       plt.show()
```



There you have it! We've created our own lexicon for the lexical category "stigma" in only three lines of code using `empath`!

Now you will do the same.

0.3.1 Exercise 1

1. Create your custom lexicon by modifying the code below.

[1495]:
```
# 1.1 Replace "LISTOFWORDS" with whatever you like, but create a list of words␣
 ↪to seed (start) your lexicon
LISTOFWORDS = []

# 1.2 Now pick a name for your lexicon (replacing "YOURCATEGORYNAMEHERE") and␣
 ↪replace LISTOFWORDS with your list.
# If you'd prefer, you can use the alternatives to Reddit-based word embeddings␣
 ↪discussed above. You can also
# modify the size argument.
lexicon.create_category("YOURCATEGORYNAMEHERE", LISTOFWORDS, model="reddit",␣
 ↪size=300)
```

```
# 1.3 Replace the name with your chosen name and run this cell
df["YOURCATEGORYNAMEHERE"] = [score_category(s, "YOURCATEGORYNAMEHERE",␣
 ↪normalize=True) for s in df["text"].values]
```

Exercise 2

Now you will use your lexicon to conduct a few analyses of your own. First, describe the topic you chose and a few hypotheses.

2.1 What topic did you choose, and what could it tell us about social processes and social institutions?

Your answer here

2.2 Is text data useful for studying your topic? (It's okay if you don't think it is!)

Your answer here

Exercise 3

Now let's think about how your topic might relate to social institutions like work, schooling, or others that might structure time use (that is, what people do at what times).

3.1 Come up with a hypothesis about when during the week redditors may be more or less likely to post about your topic to r/offmychest. If you believe there's no reason to expect a relationship between the day of the week and writing about your topic, you may explain why instead. Clarify whether your hypothesis is causal or correlational (although we will not adequately *test* causal claims at this point). You may also describe potential confounding variables.

Your answer here

3.2 Create a plot showing variation in the frequency of posts about your topic to r/offmychest over the course of the week. You may modify code from earlier in the notebook.

```
[1489]: # YOUR CODE HERE
```

3.3 Does this plot support your hypothesis? If you have a background in statistics or want to try something new, you may conduct and refer to formal hypothesis tests. If you believe there may be confounding variables, please describe them.

Your answer here

Exercise 4

4.1 Come up with a hypothesis about when during the *day* redditors may be more or less likely to post about your topic to r/offmychest. If you believe there's no reason to expect a relationship between time of day and writing about your topic, you may explain why instead. Clarify whether your hypothesis is causal or correlational (although we will not adequately *test* causal claims at this point). You may also describe potential confounding variables.

Your answer here

4.2 Create a plot showing variation in the frequency of posts about your topic to r/offmychest over the course of the day. You may modify code from earlier in the notebook.

```
[1490]: # YOUR CODE HERE
```

4.3 Does this plot support your hypothesis? If you have a background in statistics or want to try something new, you may conduct and refer to formal hypothesis tests. If you believe there may be confounding variables, please describe them.

Your answer here

Exercise 5

5.1 Come up with a hypothesis about whether a post's karma (upvotes, the "score" field in our dataset) would be related to the extent your topic is discussed. If you believe there is no reason to expect such a relationship, you may explain that instead. Clarify whether your hypothesis is causal or correlational (although we will not adequately *test* causal claims at this point). You may also describe potential confounding variables.

Your answer here

5.2 Create a plot showing the relationship between a post's score (df["score"]) and your topic. You may modify code from earlier in the notebook.

[1492]:
```
# YOUR CODE HERE
```

5.3 Does this plot support your hypothesis? If you have a background in statistics or want to try something new, you may conduct and refer to formal hypothesis tests. If you believe there may be confounding variables, please describe them.

Your answer here

Exercise 6

6.1 Come up with a hypothesis about another lexical category that should be associated (positively or negatively) with your own category. Be sure to think about social practices. What is it about how people live that makes it likely they would (or would not) write about both your topic and this other topic in the same posts to r/offmychest? Clarify whether your hypothesis is causal or correlational (although we will not adequately *test* causal claims at this point). You may also describe potential confounding variables.

Your answer here

6.2 Create a plot showing the relationship between your category and the category you chose. You may modify code from earlier in the notebook.

[1496]:
```
# YOUR CODE HERE
```

6.3 Does this plot support your hypothesis? If you have a background in statistics or want to try something new, you may conduct and refer to formal hypothesis tests. If you believe there may be confounding variables, please describe them.

Your answer here