Russell Ronalds Comp. Ling.

2/7/19 HW 1

1. I added a few extra lines, just to get rid of a few more insignificant, but common, occurrences (numbers, single letters, etc.)
2. a = no top words removed, guess at top five unique words; b = with 100 top-words removed; c = with 200 top-words removed:

Corpus 1: a. food, breakfast, dinner, cook, lunch [within the top 66]

b. food, eat, eating, breakfast, dinner

c. food, eat, eating, breakfast, dinner

Corpus 2: a. speech, language, speak, native, language [within top 28]

b. speech, language, speak, didnt, let

c. speech, language, speak, teach, native, parents

Seemingly most common RT, which skews the data: “@nnekaiko: All of us who’s parents didn’t teach us to speak our native language, let us gather here and cry”

Corpus 3: a. cat, watch, doggy, found, crying [within top 13]

b. cat, watch, doggy, found, does, crying

c. cat, dog, found, crying, pls, youtube

Seemingly most common RT, which skews the data: “@courtdezy: i found this youtube that does doggy makeovers... and i'm crying pls watch this”

Corpus 4: a. trump, wall, build, crime, government [with top 26]

b. trump, wall, build, crime, government

c. wall, build, crime government fall

Seemingly most common RT, which skews the data: “@realDonaldTrump: BUILD A WALL & CRIME WILL FALL!”

Corpus 5: a. game, football, bowl, nfl, brady [within top 64]

b. game, team, football, bowl, de, live

c. team, football, bowl, nfl, play

Class Choice: a. film, actor, director, jussie, smollet, crime [within top 24]

b. film, actor, de, director, hate

c. film, actor, director, hate, jussie, smollet

Seemingly most common RT, which skews the data: “@\_SJPeace\_: Actor Jussie Smollett was a victim of a racially motivated and homophobic attack. Chemicals were thrown at him and they trie…”

Mystery: a. food, eat, eating, dinner, taste, breakfast [within top 40]

b. food, eat, eating, dinner, taste, breakfast

c. food, eat, eating, dinner, taste, breakfast

1. I also added a Rank column on the far left. Yes, the graphed data seems to follow Zipf’s law. See scatter plot below.
2. a. Something between 100-200 seems pretty good. It seems that if the original search terms were words that might easily appear in other contexts (eats), if the N goes too high, it might get removed.

b. The topics are definitely clearer after removing the top words, but sifting through the most common words that appear in all tweets wasn’t that hard to begin with, mostly because their length was very small compared to words like ‘breakfast’ and ‘football’.

c.  Yes, we needed to use the ‘all’ corpus to get rid of the top words, because otherwise we would have been getting rid of important words in the smaller corpra by the time we got to word rank 10 (“doggy”, “speech”), rank 8 (“food”) rank 6 (“trump”), etc. or we wouldn’t have been getting rid of all the noise of “to”, “of”, “my”, etc. which are pretty clearly not keywords. If we had removed the top 50-100 words from each corpus without referencing the ‘all’ corpus, we might have gotten an idea of the topics, but it would have been hard, e.g.:

Words 100-104, corpus 1: chicken, did, there, while, need

c2: no, de, great, actual, should

c3: life, bac, shouldve, todo, complain

c4: years, than, clinton, de, support

c5: best, basketball, title, been, go

cc: words, like, first, member, best

mys: see, order, better, had, than

Never mind, we wouldn’t have known at all the keywords… Stop words are too intermingled, and the top keywords are especially intermingled with the most frequent stop words.

d.  To automatically determine which topics a mystery file came from:

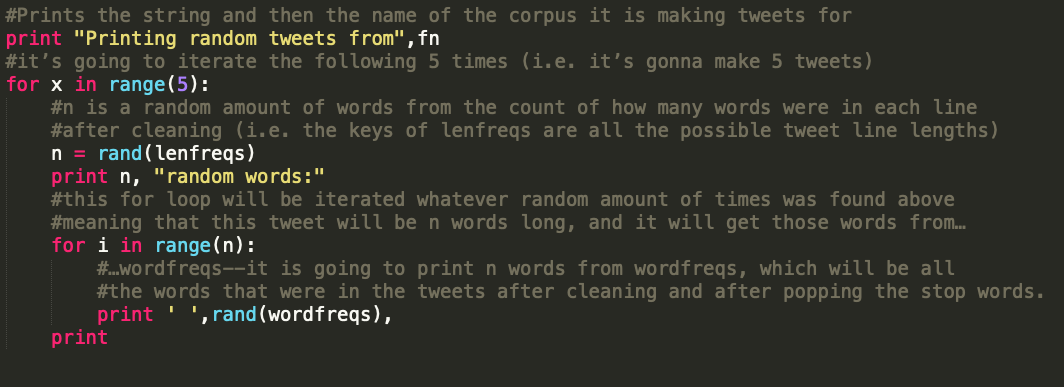
After removing the stop words, I would take the top ten words of a mystery corpus and compare them to the top ten or twenty words of a known corpus. For each of the words I am looking for, if it exists in the wordfreqs of the known corpus, take the value of that word in the *known corpus’ wordfreqs*. Sum these values. Repeat this for each known corpus. The corpus with the highest sum is likely to have the same topic words.



**EXTRA CREDIT**

The tweet generator lines of code will produce 5 random tweets for each file (argument) provided on the command line after the first file (argument). I attached a screenshot of the commented upon tweet generator. Since the random tweet generator is pulling only from the keywords, and not from the filler words that are, in fact, the most common words of all tweets, they are totally unintelligible. Also, since it pulls from *any* words that are not in the stop words, it is often choosing from the super infrequent obscure things like “venom”, “despotricado”, and “schiff”. Pulling from only the top 50/100 might work better?

With such a simple version of the random tweets, it might be better to create ones that are smaller, so that their ungrammaticalities are not so obvious. Maybe pulling words that are followed by an exclamation point might be one way of creating one-word tweets that are sensical?



c1 random tweet: “venom jks wanna start selling wrote accurate copies bathroom trouble forget whole fried continually cheshire grand la chicfila”

c2: “calling another claim notice sound familys teach mindshiftkqed language language tr country blanks testing telling speech valley administration keynote parents”

c3: “en pas supremo mess pls honest crase”

etc.

From that, I changed the random tweets to:

c1: “cant hole more wrong go photosynthetic”

“good brexit she lychee his groupie”

“only kidz today music see cars”

“their ate think hot now cooking”

“she service then females had wine”

c4: “all military one mouth her locked”

“would example its mueller new music”

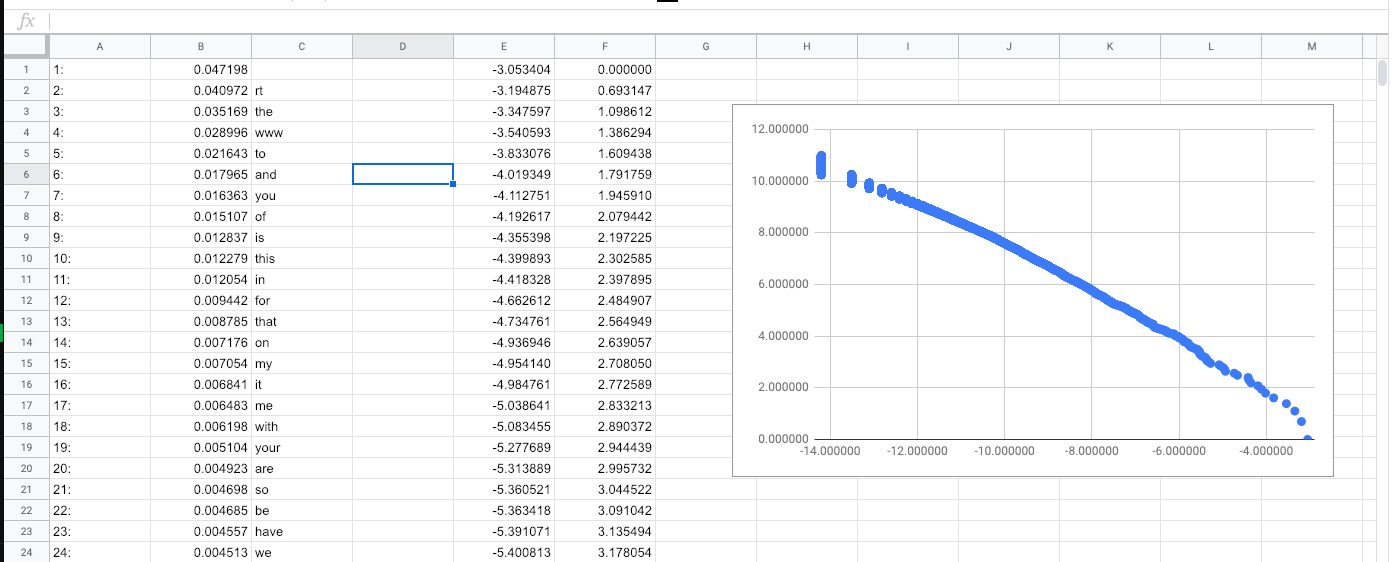
“got wrong why schools were example”

“all financial has mueller even feeling”

“cant second more muellers had mueller”

Anyway, still terrible, but getting somewhere. Maybe if I got better with regex I could turn it back into a sentence with capitalization and such….

Remember: Can’t second more Muellers had Mueller.

Scatter plot:

The lowest point on the plot is the ‘’ empty set that was super common after using the clean function (maybe before cleaning, too, actually). We could get rid of it by only running get\_freqs on lines that have data (run it under an “If lines:” or something), but it disappears as a stop word anyway, and we can just ignore it in this graph—with so many other points, it wouldn’t do much to a regression. Also, maybe there is a sag in the line from x > -8 onwards? I don’t exactly know what that is…