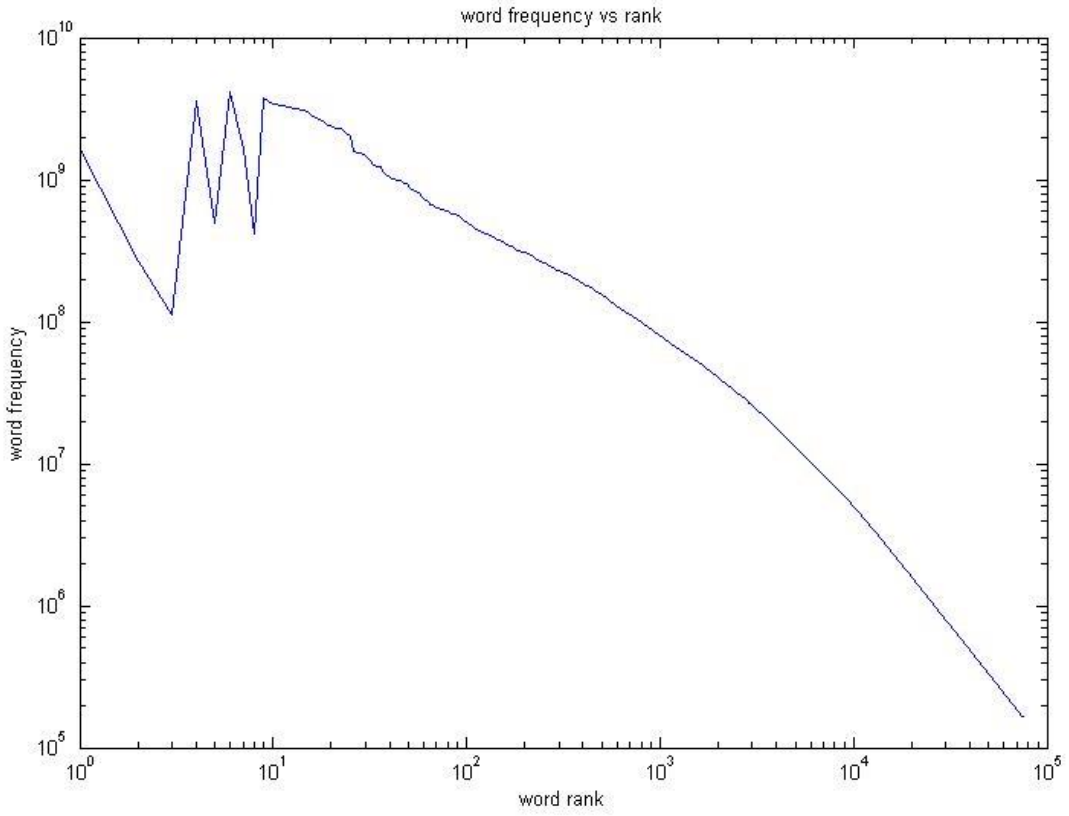
Grace S Yoo
CS341PR SP'15
HW#1 (2/2)

1. Regex
    a. All alphanumeric strings: **'A-Za-z'**
    b. All strings with 2 consecutive repeated words: I tried a few things, but if I grep –E on alice.txt using these RE, I often get repeated phrases ie "so he did so he did" along with consecutive repeated words such as "beautiful beautiful"
    here are some of the REs I tried, which are not entirely correct:
    '\b([A-Za-z].+)\s\1{1}'
    '(\b[A-Za-z].+\b)\b\1{1}\b'
    '(\b[A-Za-z].+\b)\s\b\1{1}\b'

2. Rot13 decryption
    a. Map alphanumeric characters to rot13 translation
    echo "uryyb jbeyq" | tr 'a-zA-Z' 'n-za-mN-ZA-M'
    hello world
    b. unix command w regex rot13 encrypt/decrypt:
    echo "uryyb jbeyq" | tr 'a-zA-Z' 'n-za-mN-ZA-M' | tr 'n-za-mN-ZA-M' 'a-zA-Z'

3. bigwordlist75k.txt
    a. Plot frequency as a function of rank:



    b. Zipf's law states that a word's frequency is inversely proportional to its rank. This graph seems to illustrate this type of relationship.

4. Literary works from 19th and 20th centuries

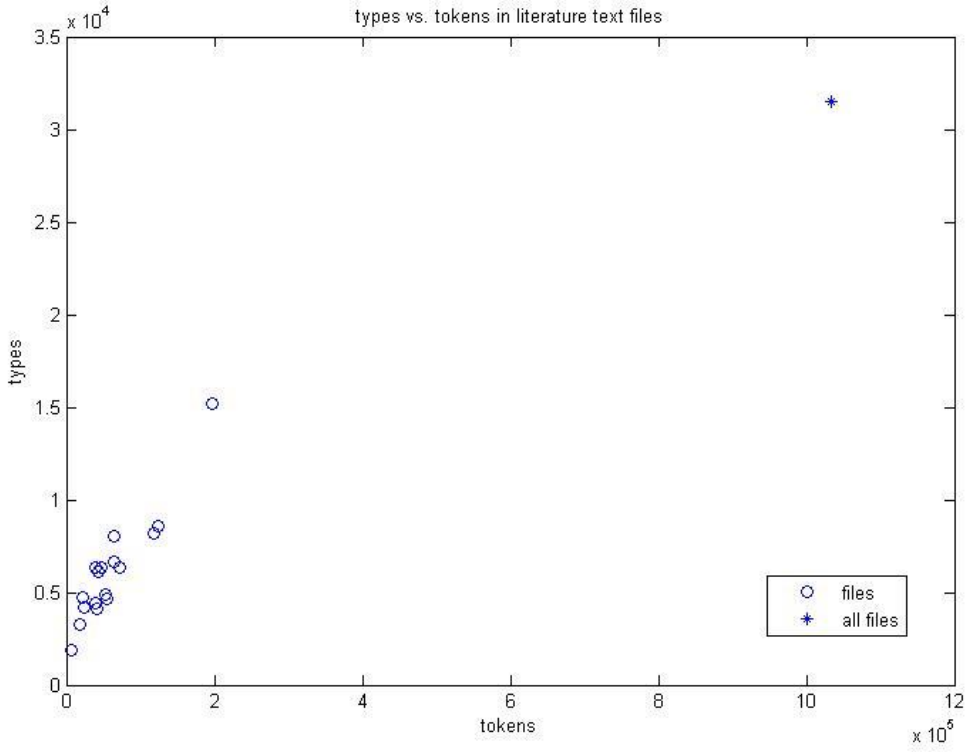
a. A-Za-z all alphanumeric characters
\n newline
this command takes all non-alphanumeric characters and replaces with newline, squeezing consecutive newlines → a single newline, then saves the tokenized texts in a new text file:
tr –sc 'A-Za-z' '\n' < filename.txt > tokenized_filename.txt

b. The web address at the very beginning of the text was also poorly handled, as it really should be 1 token but is separated into new tokens at every period. Words that are comprised of 2 or more words and joined by dashes are also poorly handled (ie "good-will") for the same reasons.
Because the possessives are a separate token from the word they modify, those words are handled poorly by this tokenization.
If we were uniquely tokenizing words (not tokenizing a word already tokenized), our convention would have failed to tokenize words that were the same but had different capitalizations.

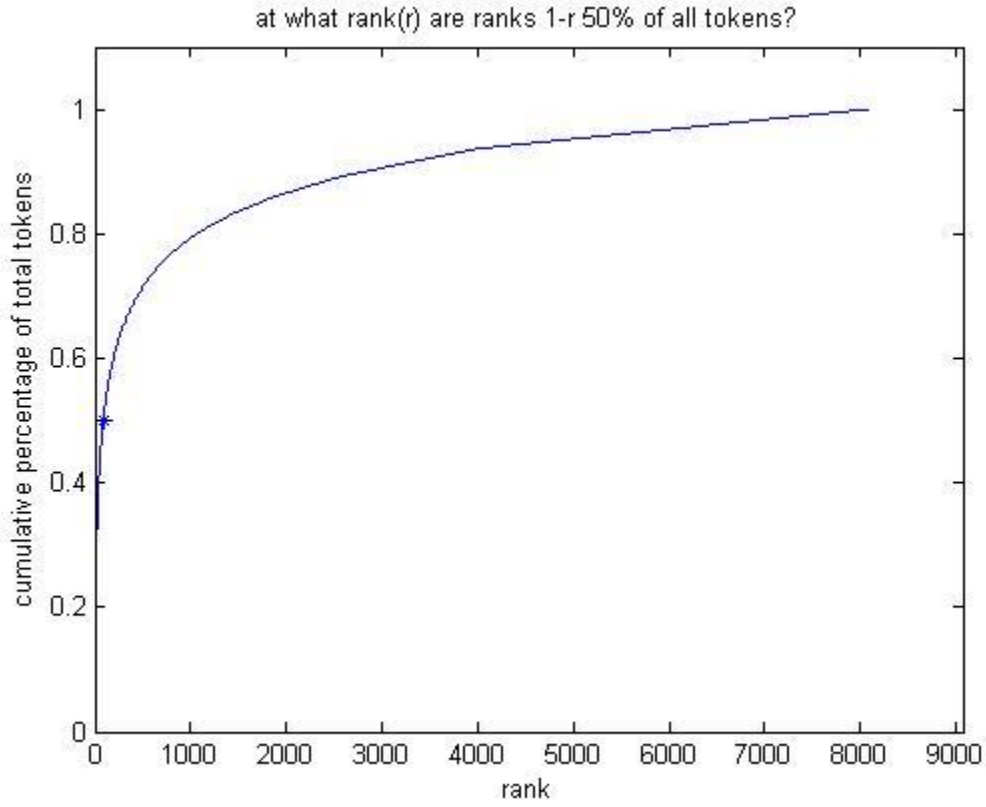
c. Command to find tokens:
tr –sc 'A-Za-z' '\n' <filename.txt | wc
Command to find types:
tr –sc 'A-Za-z' '\n' <filename.txt | sort | uniq | wc

d

Lit	Tokens	Types	Lit	Tokens	Types
Austen- Emma	55176	4655	Hugo- Miserables	196808	15174
Austen- Mansfield	53359	4915	Melville- Bartleby	6271	1848
Austen- Pride	40917	4149	Natsume- Botchan	18615	3300
Austen- Sense	40296	4429	Shelley- Frankenstein	24731	4177
Bronte- Jane Eyre	64295	8069	Thoreau- Walden	39696	6328
Dickens- 2 Cities	46835	6323	Tolstoy- Anna Karenina	118321	8188
Dickens- Expectations	64179	6652	Twain- Yankee	43491	6096
Dostoyevsky- Crime	72157	6331	Twain- Prince and Pauper	23059	4731
Dostoyevsky- Brothers	124560	8592	ALL FILES	1032766	31524



d. To answer this question, I decided to do a mini-research approach.
I chose 4 works of literature to analyze (Emma, Anna Karenina, Jane Eyre, Les Miserables)
I did command: tr –sc 'A-Za-z' '\n' < filename.txt | sort |uniq –c| sort –nr > new_filename.txt
which gave me word frequencies and the words they were associated with, starting with the highest rank (rank = 1) at the top of the list.
Using Matlab, I imported the word frequency text files and plotted the (CUMULATIVE frequencies/total types) on the Y axis, and plotted the ranks (1- #types) on the X axis. Since the text files are ordered with the highest frequencies on top, those numbers were aligned with the highest rank (1).
This gave me a graph like this: (this is for Jane Eyre)



In Jane Eyre, there were 8,069 types/ranks of words. I then used a special function to locate at which point along this plot the cumulative percentage = 0.5. This point (50% of all word TOKENS) is indicated by a star. I found the rank corresponding to cumulative percentage = 0.5 (column 2 in the table below). For files of different sizes, this value isn't very telling. So, I took the rank at 0.5 / total ranks, giving column 4 in the table below.

Literature	Rank(r) where (1-r) = 50% tokens	Total ranks (word types)	r as a percentage of all ranks	CHECKING
Les Miserables	90	15175	0.0059	0.4981
Anna Karenina	72.5	8189	0.0089	0.5070
Jane Eyre	86.5	8070	0.0107	0.5262
Emma	64.5	4656	0.0139	0.5258

Taking the average of "r as a percentage of all ranks" we get r = 0.00985. Thus, I conclude that the first 0.985% ranking words in a text account for 50% of the tokens.
To check my findings, I plugged in the values of r and R that I had gotten into a formula for Zipf's law that I found here: http://mathworld.wolfram.com/ZipfsLaw.html

$P(r) \approx \frac{1}{r \ln(1.78\,R)}$, This formula states that the probability of a word at rank r can be found with this formula, were R is the total number of ranks/types.

Using a for loop in Matlab, I summed up all the probabilities of words from 1 – r for each value of r I found. The result, P(r), should be 50%, if I am answering the question properly. The column CHECKING in the above table holds all the values of P(r) I got from plugging in rank(r) and total ranks. You can see that they are all roughly 50%.

5. I spent about 6 hours on this assignment. Everything was pretty easy, but I really struggled with 1B (grepping 2 consecutive duplicates). Cumulatively, I spent about 3 hours on that one question. I am not counting the time I spent on the bonuses.

Extra credit from the Unix/Regex Lab

In alice.txt

1. How many 4 letter words? (types)
   command: tr –sc 'A-Za-z' '\n' <alice.txt | sort | grep –E '\b[A-Za-z]{4}\b' | uniq |wc
   answer: 411

2. How many different words are there with no vowels? (types)
   command: tr –sc 'A-Za-z' '\n' <alice.txt | sort | grep –v '[aeiou]' | uniq | wc
   answer: 22
   entire set = {**by, c, cry, d, dry, fly, hjckrrh, hm, ll, m, my, s, sh, shy, shyly, sky, t, try, v, w, why, x**}

   What subtypes do they belong to?
   ^^ I am not exactly sure what this means.
   If you meant what kinds of words are there without vowels in this text,
   a. words that have "y" instead of a vowel
      (ie **cry, try, shyly, sky, why, fly**)
   b. word fragments that were separated from the entire word, because of an apostrophe or something
      (ie **ll** from I'll, **m** from I'm, **t** from don't won't can't etc,)
   c. single letter consonants that were used as is
      (ie "…why is it you hate **c** and **d** she added in a whisper", "…brass plate with the name **w**")
   d. onomatopoeias or nonsense words
      (ie **hjckrrh, sh, hm**)
   e. roman numerals
      (ie **v** from chapter v, **x** from chapter x)

3. How many "1 syllable" words are there (words with exactly 1 vowel)? (types)
   command: tr –sc 'A-Za-z' '\n' <alice.txt | sort | grep –I '\b[^aeiou]*[aeiou][^aeiou]*\b' | uniq | wc
   answer: 585

   if I also include y as a vowel (many of the words in the previous set were 2 syllable, thanks to a y)
   command: tr –sc 'A-Za-z' '\n' <alice.txt | sort | grep –I '\b[^aeiouy]*[aeiouy][^aeiouy]*\b' | uniq | wc
   answer: 489

   How to improve this: many 1 syllable words like "steam" or "dream" would be left out. If I could figure out how to count clusters of consecutive vowels as equivalent to a single vowel within a word, it might improve precision.