CSE 476/598 Intro to Natural Language Processing

Assignment 1

Rohan Jain

rjjain1@asu.edu

Class Mate worked with- Pradeep Chaudhari

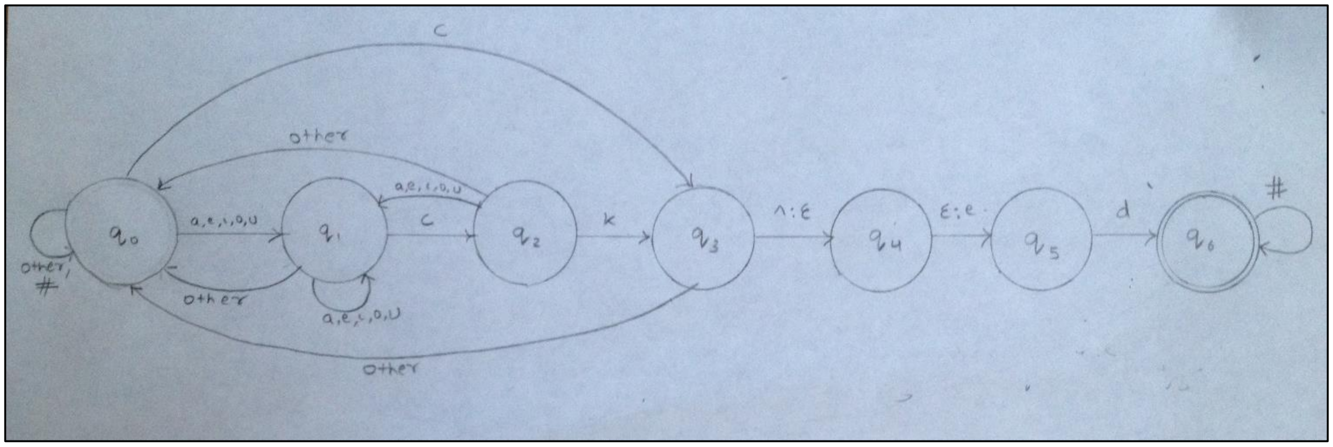
Solution 1

(a) The set of all alphanumeric strings using Python notation:
^[a-zA-Z0-9]*$

(b) The set of all strings with two consecutive repeated words, such as \the the using Python notation:
^[\w\s\W]*the\sthe[\s\w\W]*$

(c) A set of 10 or more face emoticons (e.g., ;-) =(:-p) using Python notation:
^\:[(\))(\()(\D)(*)(\P)(\\)(\()(\|)(\$)(\O)]$
What emoticons does your regular expression fail to match?
;)   :'(>:)   >:   XD   =)   >.<

Solution 2

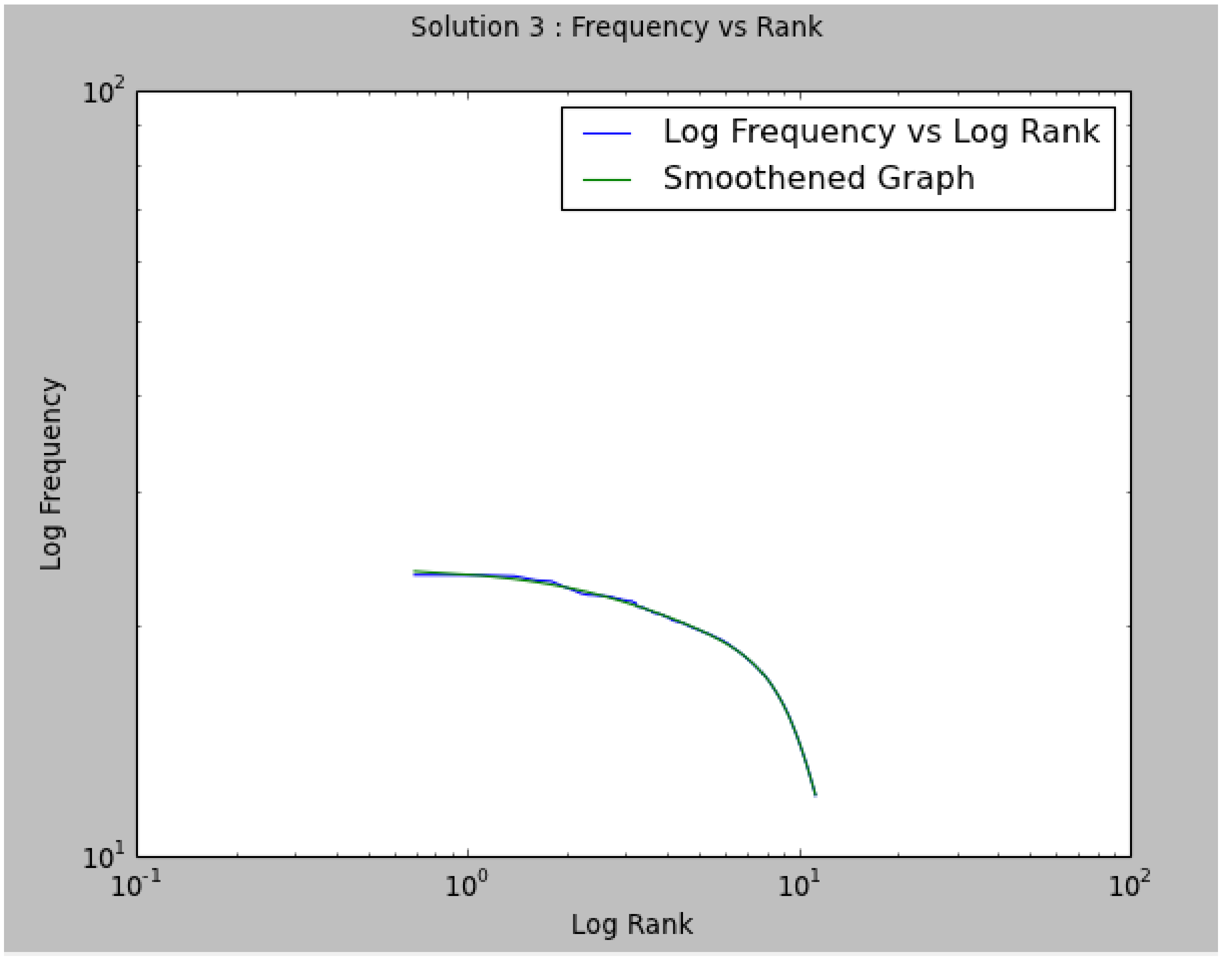


State Sequences

Surface level	Intermediate Level	State Sequences	
Panicked	Panick^ed#	q_0-q_0-q_1-q_0-q_1-q_2-q_3-q_4-q_5-q_6-q_6	Accept
Paniced	Panic^ed#	q_0-q_0-q_1-q_0-q_1-q_2-q_0-q_0-q_0-q_0	Reject
Synced	Sync^ed#	q_0-q_0-q_0-q_0-q_3-q_4-q_5-q_6-q_6	Accept

<u>Solution 3</u>

The graph when the frequency of each word type is plotted against rank of the word type on a log-log scale is shown below-



Interpretation:

By looking at the graph we can make out that at high ranks the frequencies of words and similarly at low ranks the frequencies of words are high which is also very obvious.

We can also make out that the product of the log values of rank and frequencies are nearly constant at any time.

The Zipf's law states that when word types are ranked by frequency , then frequency(f)*rank(r) is roughly equal to some constant (k):

$f\ X\ r = k$

Therefore the multiplication of rank and frequency of the different word types has to be a constant to satisfy Zipf's law.

If,

f X r = k

Then,

Log f + Log r = Log k

We therefore take different points (x,y) and their values (log r, log f) on the graph-

x(log rank)	y(log frequency)	log x+log y(log rank+log frequency)
11.2249	12.011	23.2359
10.7363	12.859	23.5958
8.80998	16.012	24.8220
4.66309	19.977	24.6409
2.04835	22.332	24.3809
0.73501	23.305	24.0400

As we can see from the above table that the values of log rank+log frequency are nearly the same.

Thus the values form a constant and thereby the above graph satisfies the Zipf's law.

<u>Solution 4:</u>

(a) For tokenizing I have used the following Regular Expression in Python : \w+-\w+|\w+\'s|\w+

I have considered tokens to be of alphanumeric characters and have also considered words with special characters between them like – and ‘. For example words like broad-minded and children's are also accepted by the regular expression. It further rejects words which are syntactically incorrect like broad-. As the tokens are attained I append the tokens to a list which will store all the tokens of a particular part.

For tokenization the system reads the text file and then processes each character against the Regular expression. If the set of characters is accepted it will be appended in a list. The list in this manner in the end consists of all the set of tokens.

(b) Number of word tokens in part1: 1023339
Number of word types in part1: 35295
(c) Number of word types that appear exactly once in part1: 14984
(d) Probability that the next word sampled will be a new word type in part1: 0.0146422641959
(e) Comparison between actual percentage of new word types in part 2 compare to Good Turing estimate from part 1:

Percentage of new word type addition in part 2 : 1.4060762899
Percentage of new word type addition in part 1 : 1.46422641959
Difference in percentage between part 2 and part 1 : -0.0581501296887

(f) No of word tokens in part 1 and part 2 2038860
No of word types in part 1 and part 2 : 69306

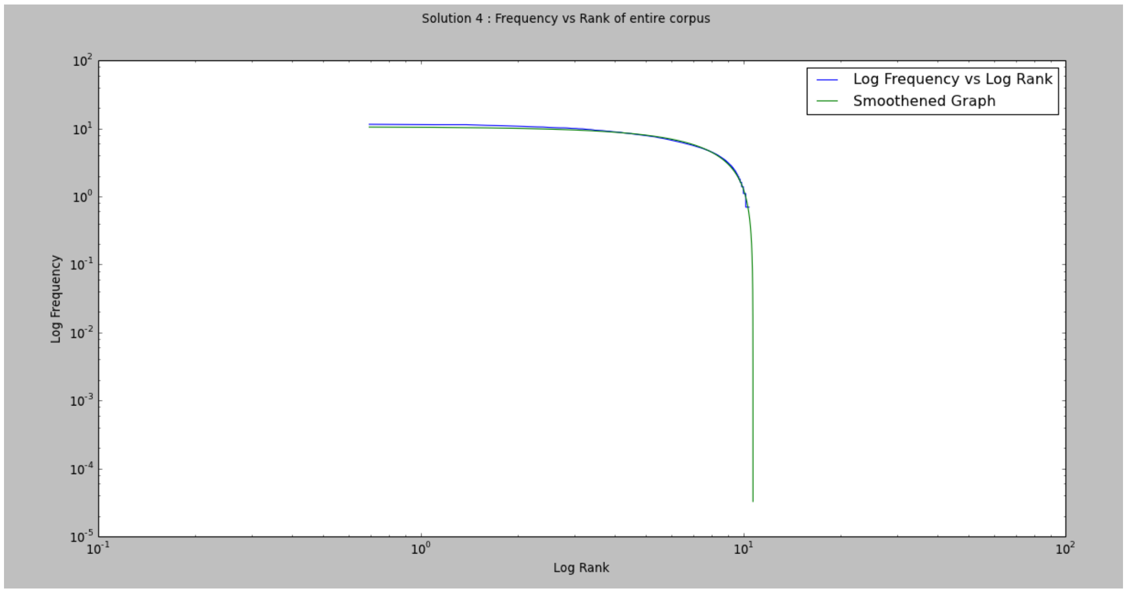
(g) Words with one frequency in part 1 and part 2 : 29263
(h) Probability of new word type of first 2/3rd of the book : 0.0143526284296
(i) Comparison between actual percentage of new word types in remaining section i.e. last 1/3rd of the book compare to Good Turing estimate from part h:
Percentage of new word type of last 1/3rd of the book : 1.3838512164
Percentage of new word type of first 2/3rd of the book : 1.43526284296
Difference between last 1/3rd part and first 2/3rd part : -0.0514116265572

(j) On comparing we can see that when more corpus is considered the Good Turing estimate values decreases.

This can be explained by the fact that when more and more corpus words are considered the GT estimate goes down and thereby the accuracy decreases.

The GT Estimate Probability of finding unseen words decreases when the corpus is increased therefore when we consider the English language it would not be a wise decision to consider the Good Turing technique as the English Language has an extremely wide corpus. Thus with such a big corpus the efficiency reached would be vey less.

(k)



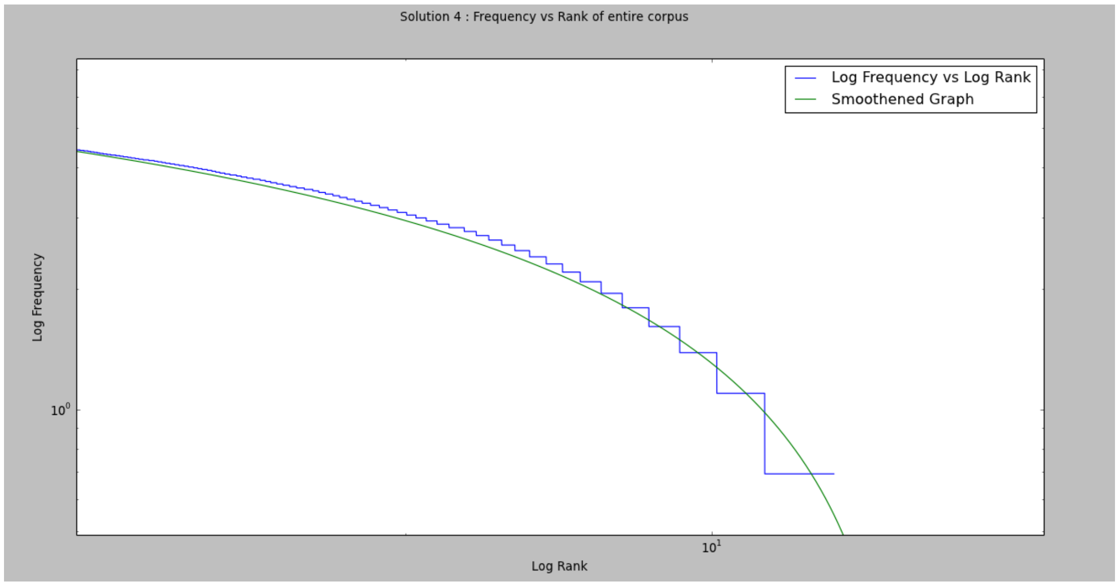
(l) By looking at the graph we can make out that at high ranks the frequencies of words and similarly at low ranks the frequencies of words are high which is also very obvious.
We can also make out that the product of the log values of rank and frequencies are nearly constant at any time.

The Zipf's law states that when word types are ranked by frequency, then frequency(f)*rank(r) is roughly equal to some constant (k):
f X r = k
Therefore the multiplication of rank and frequency of the different word types has to be a constant to satisfy Zipf's law.
If,
f X r = k
Then,
Log f + Log r = Log k
We therefore take different points (x,y) and their values (log r, log f) on the graph-

x(log rank)	y(log frequency)	log x+log y(log rank+log frequency)
10.1804	1.09691	11.27
8.78749	3.40033	12.18
6.40691	6.23488	12.64
2.98726	9.97211	12.95
4.89032	7.8885	12.77
1.35073	11.3122	12.66
0.694215	11.4647	12.15

As we can see from the above table that the values of log rank+log frequency are nearly the same.
Thus the values form a constant and thereby the above graph satisfies the Zipf's law.

High noise is observed in the graph at a high rank and a low frequency. This can be shown by zooming over the above graph as shown in the below figure:



Answer 5:

It took me about 3 days to complete this problem.