

Report for Assignment 1

In this assignment, I do the main part of the programming task in Perl (for the step 1 to 3), and some extra experimental coding tasks in Java (Using the Princeton WordNet 1.6 and the JWNL WordNet Interface).

For the step 1 to 3 in this assignment, the pseudocode for the algorithm is listed below (for the 791a-assg1.pl) :

- Open the corpus file (here it should be a text file in plain txt format);
- Read all the lines of the text into an array;
- For each item of the array (it is a line in the corpus):
 - Pickup the all possible WORD TYPES in the line;
 - Exclude the forms like 's, 'd, and 'll
 - Put the extracted word type into a hash, and increase its current frequency
- Sort the result into an array (as word types' frequencies in descending order)
- Print the required information according to the step 1 to 3

To run the program, you need perl, the code 791a-assg1.pl, and a plain text of the corpus, then type:
perl 791a-assg1.pl ~/corpus.txt > result.txt. (you should provide the complete path for the corpus file!)
Then the result will be in the result.txt file.

I also make an extra experiment to test whether the distribution of WORD TOKEN will confirm the Zipf's law. I use the Princeton WordNet 1.6 as the stemming tool to morph the WORD TYPE in the corpus into their original WORD TOKEN and count its frequency.

For this extra experiment, I use MorphWord.java and convrt_stem_file.pl. To run the code, you need to install Princeton WordNet 1.6 first, and you also have to install JWNL WordNet Interface. (you can find it in www.sourceforge.net, please get the newest patched version) I will not submit these two components because two reasons: The first, it is only an extra experiment for interest. Moreover, they are so large, over 40Mb, that I can't upload it.

After installing WordNet 1.6 and JWNL WordNet correctly, you can type:

```
perl extra_expr.pl d:\corpus.txt d:\output.txt (corpus.txt can be any corpus file)
(assume that you install JWNL in d:\jwnl)
javac -classpath d:\jwnl MorphWord.java
java MorphWord d:\output.txt d:\tmp.txt
perl convrt_stem_file.pl d:\tmp.txt d:\stem_output.txt ( stem_output.txt can be any file name you want)
```

In this assignment I use the following corpus:

1. "Around the World in Eighty Days" by Jules Verne, a science fiction novel about 63,000 words;
2. "Chatelar" by D. H. Lawrence, a romantic novel about 116,000 words;
3. "Sign of Four" by Sir. Conan Doyle, a detective novel about 4,000 words;

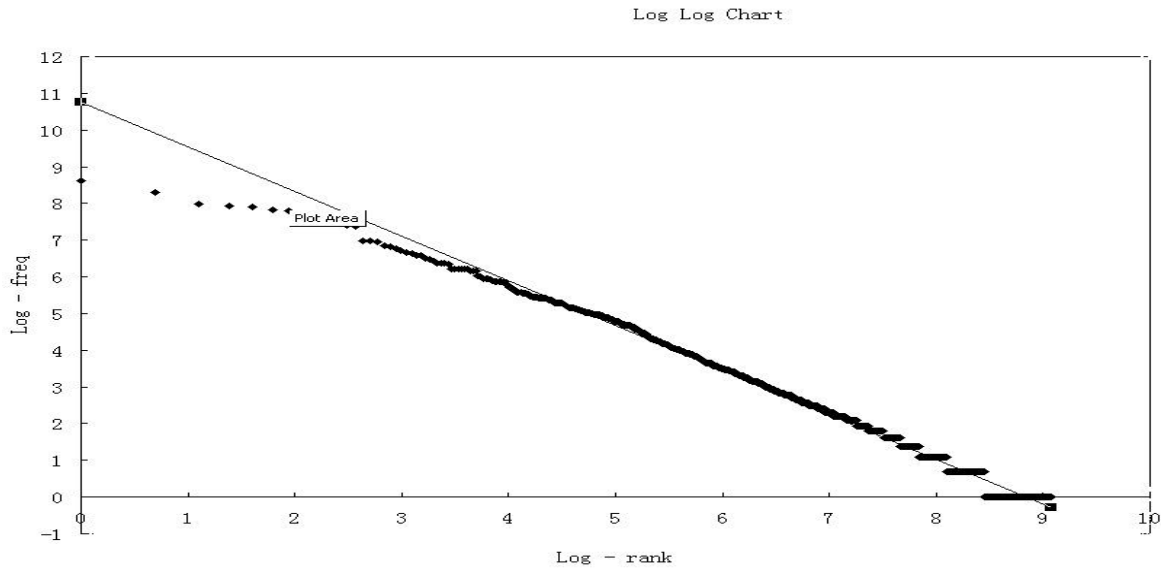
4. A combined text with the above three novels, about 223,000 words;
5. One randomly chosen DUC testing document about 675 words, from newspaper reports;
6. Four randomly chosen DUC testing documents, about 2,300 words, from newspaper reports;
7. Five randomly chosen DUC testing documents 2,900 words, from newspaper reports;
8. a combined text from about 600 DUC testing documents, about 351,000 words.

Experiment 1:

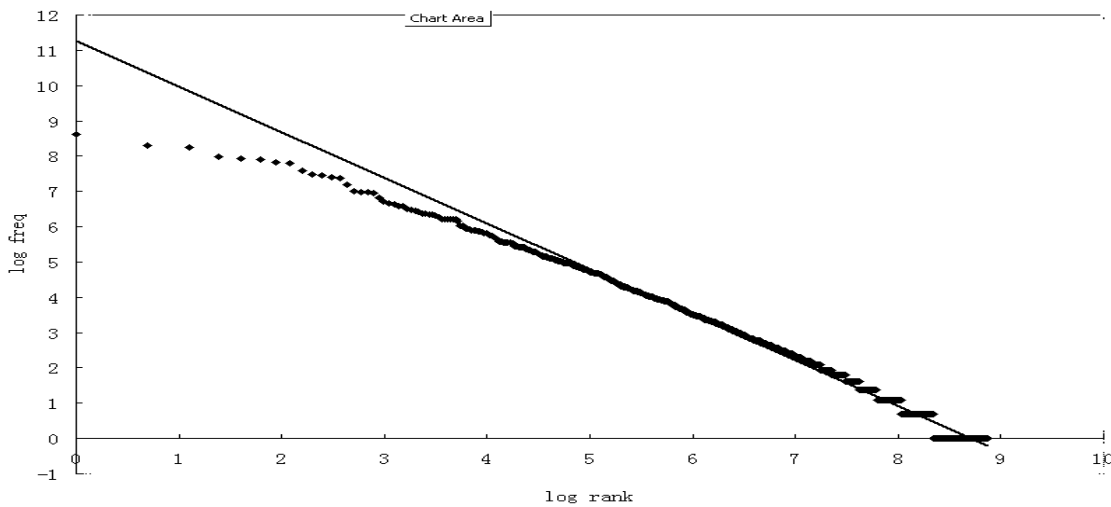
Does the WORD TYPE distribution confirm Zipf's law? What about WORD TOKEN distribution?

Here I use corpus 1 and 2. I draw the Log-Log chart for the WORD TYPE distribution. I also draw the Log-Log chart for the WORD TOKEN distribution to verify if Zipf's law can be applied to the WORD TOKEN distribution.

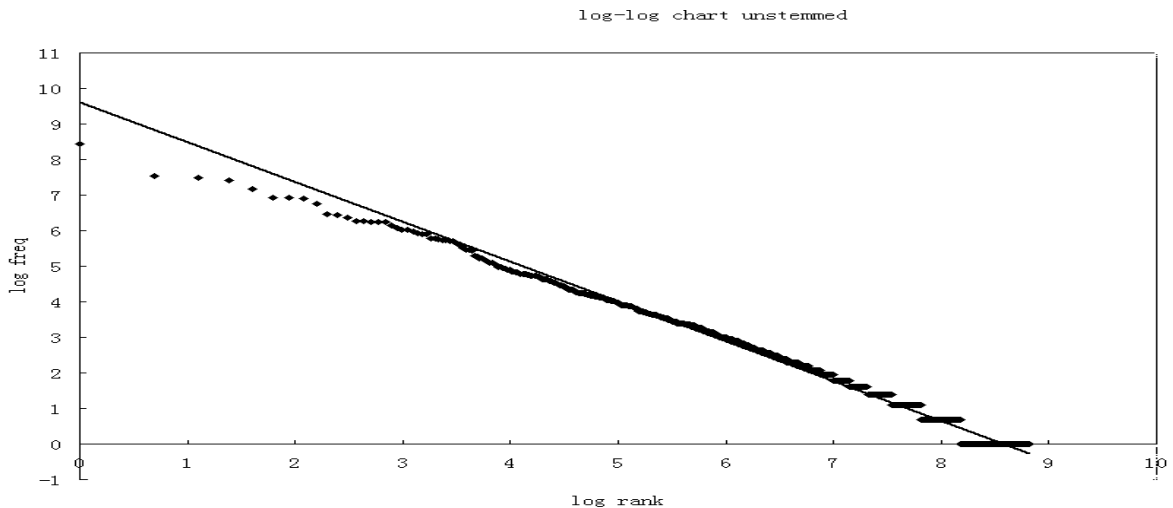
The reason I choose corpus 1 and 2 is that they are in different category. Furthermore, Jules Verne is a French. It may indicate that whether text from non-native author will confirm Zipf's law.



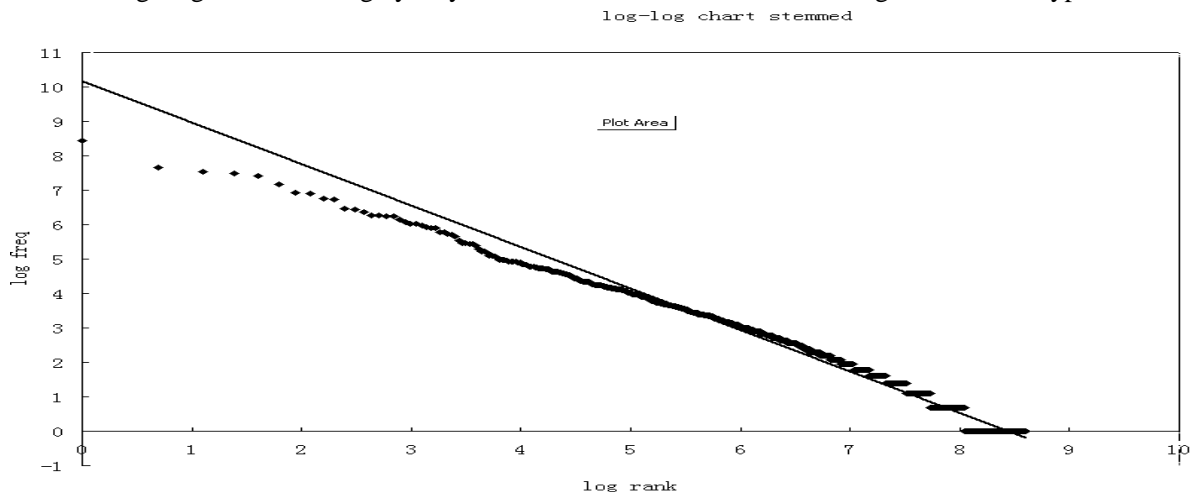
Log-Log Chart for Chatelar before stemming , 8736 word types (with the linear regression trend line)
log-log chart stemmed



Log-Log Chart for Chatelar after stemming , 7088 word tokens (with the linear regression trend line)



Log-Log Chart for “Eighty Days around the World” before stemming , 6789 word types



Log-Log Chart for “Eighty Days around the World” after stemming , 5418 word tokens

Here we can see that the distribution for word types does confirm the Zipf’s law well.

However, we can note that the word token distribution for “Eighty Days around the World” barely confirms the law, say, about half of the word tokens go off the linear regression trend line. Can we still state that Zipf’s law applies to the word token distribution?

My statement is that Zipf’s law can also apply for the word token distribution. My explanation for the exception of the “Eighty Days around the world” is that with stemming, some words that previously are not highly frequent, for example, is, are, was and were, are changed into highly frequent word, for example, be. Therefore, they don’t not abide by the law any more. But most word tokens with medium frequency still confirm the law. We can verify this fact by the chart for “Chatelar”. Since “Eighty Days around the World” has a smaller vocabulary, after stemming it has less remained words with medium frequency which still confirm the law, thus the new word token distribution looks breaking down the law.

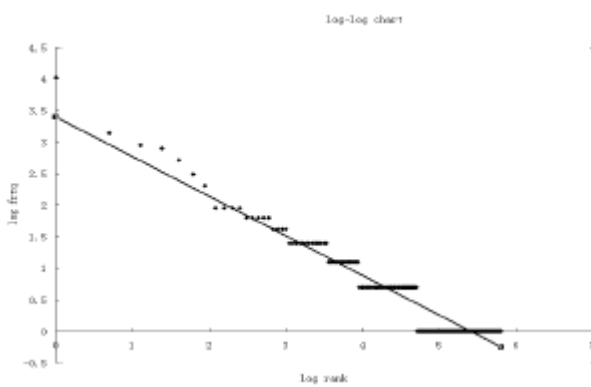
(Please note that all the following experiments are about word type distribution)

Experiment 2:

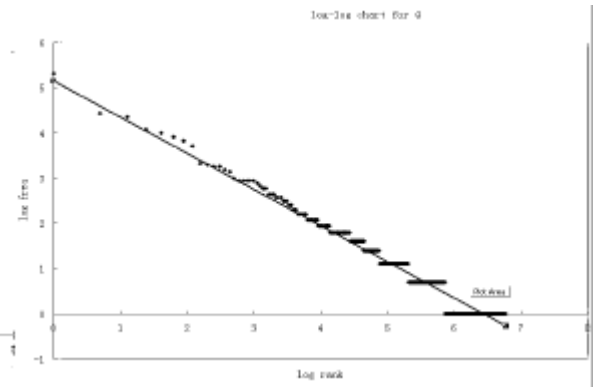
What is the Threshold for the scale of the corpus where Zipf’s law will be effective?

Since Zipf’s law is obviously a statistical statement, it surely is to be effective only if the scale of the text

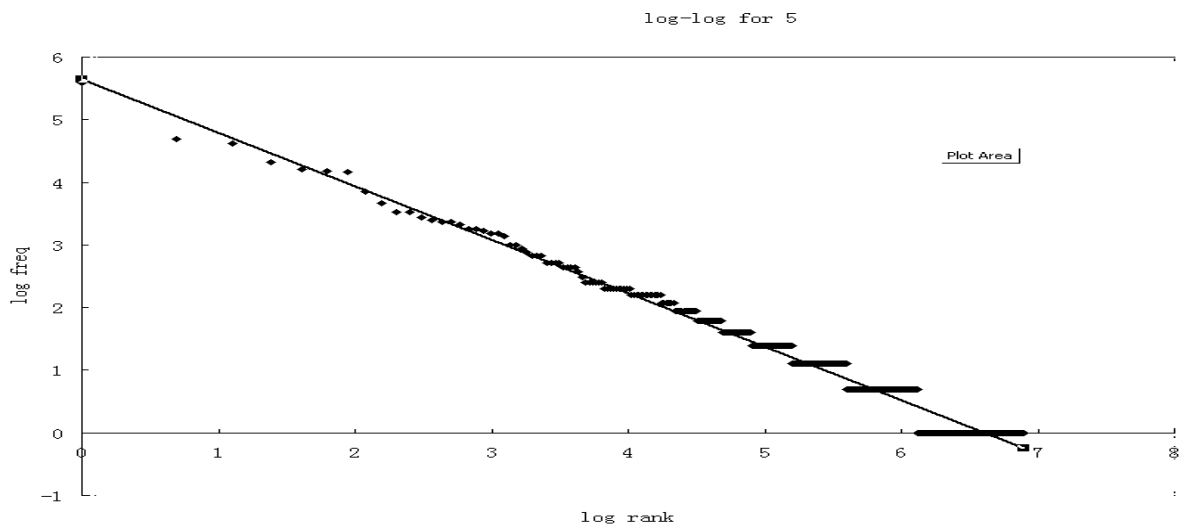
gets over some threshold. Here I try to test it by corpus 5,6, and 7.



Log-log chart for one DUC document about 250 word types



Log-log chart for 4 DUC documents about 850 word types



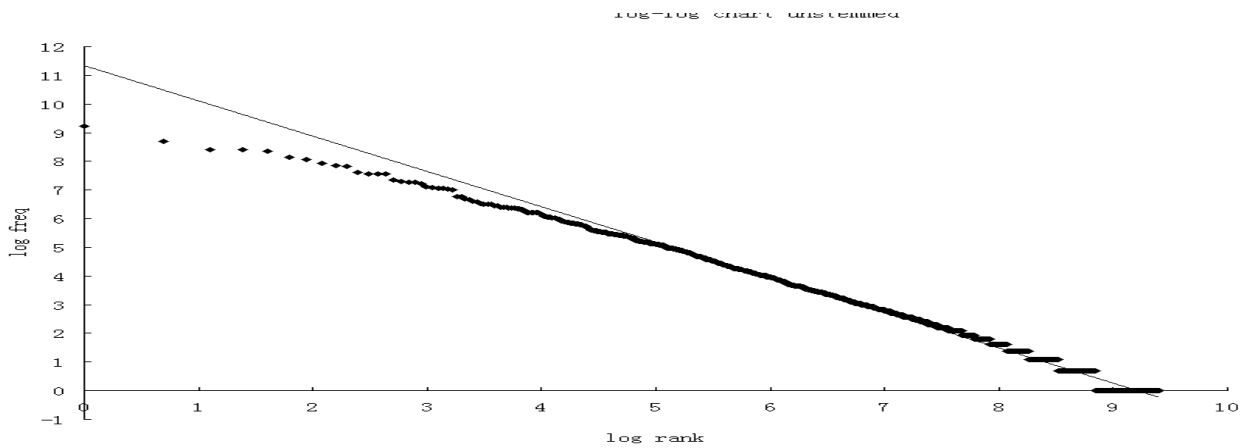
Log-log chart for 5 DUC documents, about 1,100 word types

Here you can find out from the chart for one document that it barely confirms the law, When the size of the word types set increases to about 1000 (see the charts for 4 and 5 documents), the law seems to get effective. Since all these DUC documents are randomly chosen and are written by average English reporters, I conclude that the Zipf's law will get effective only if the size of the word type set of the corpus gets over 1,000~1,200.

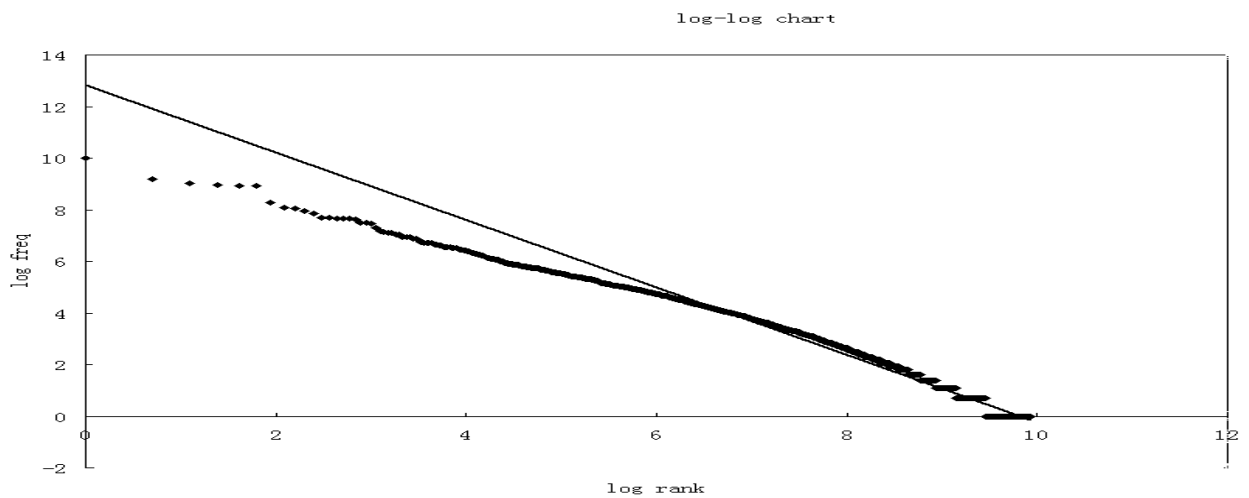
Experiment 3:

While according to Zipf's law, rank timing frequency equals to a constant, is this constant determined by document size factor or by the factor of the author?

In this experiment I use corpus 4 and 8. They are also created by combining different documents. If the constant is determined by the document size, the results for these two combined texts should also confirm Zipf's law in a certain degree. But if the constant is determined by the factor of author, for example, his vocabulary and writing style, the results may break the law.



Log-log chart for corpus 4, a combination of 3 novels



Log-log chart for corpus 8, a combination of 600 short new reports

Here we can see that the more diverse the original resources, the more its result breaks the law. (please note that though the chart of the combination of 3 novels seems acceptable for the law, its result is noticeably worse than that of single novel, especially for those word types with high frequency.

If we assume that the constant is determined by the size of the text (no matter it is determined by a hidden linear function or not), we can also assume that the combined text is written by different authors one by one, thus it should also confirm the Zipf's law. However here we get a contradict. Therefore I can make my conclusion that the factor is not determined by the size, likely by the author.

Another interesting experiment:

Whether the Zipf's law can be applied to other similar languages, for example, French and Latin?

I would like to take this experiment but I have to give up because of my insufficient knowledge on these languages.