Exercise 2 – Downloading Texts and Zipf's Law

2b)

Explanation of the steps in 1b):
In order to remove the preamble and appendix, I looked at the text as a whole. Since there was no markup indicating where the preamble and appendix started and ended, I had to decide that myself. For me, the actual text starts after the author's preface with the beginning of Chapter 1 and ends after the conclusion.
Then I searched the text for the strings indicating the start and end of the text and found out their position in the whole text string by using the `.finditer()`–method and adding the respective positions to a list. Then, by using the positions saved in the list, I was able to indicate that the actual text lies only between these two positions and saved this text in the variable `only_text`.

2c)

In order to only get the words and tokenize afterwards in ex.1c, I substituted all the punctuation marks in the text by using `re.sub()`, substituting them with an empty string. Moreover, I case-folded the text so that only non-capital letters appear. I think this makes sense because later we want to examine the frequency of certain words in the text and for that it doesn't make a difference whether the word e.g. starts with a capital letter or not – i.e. I do not distinguish between capitalized words and non-capitalized ones.

2d)

```
word     absolute frequency

------  --------------------

the               3702

and               3087

a              1829

to              1711

of              1434

he               1197

was               1168

it             1149

in              941

that              905

his              815
```

i	781
you	777
tom	688
with	647
but	580
they	558
for	525
had	512
him	434

2e)

frequency	number of words with this frequency
1	3767
2	1202
3	608
4	382
5	231
6	172
7	147
8	127
9	74
10	93
11-50	508
51-100	81
>100	104

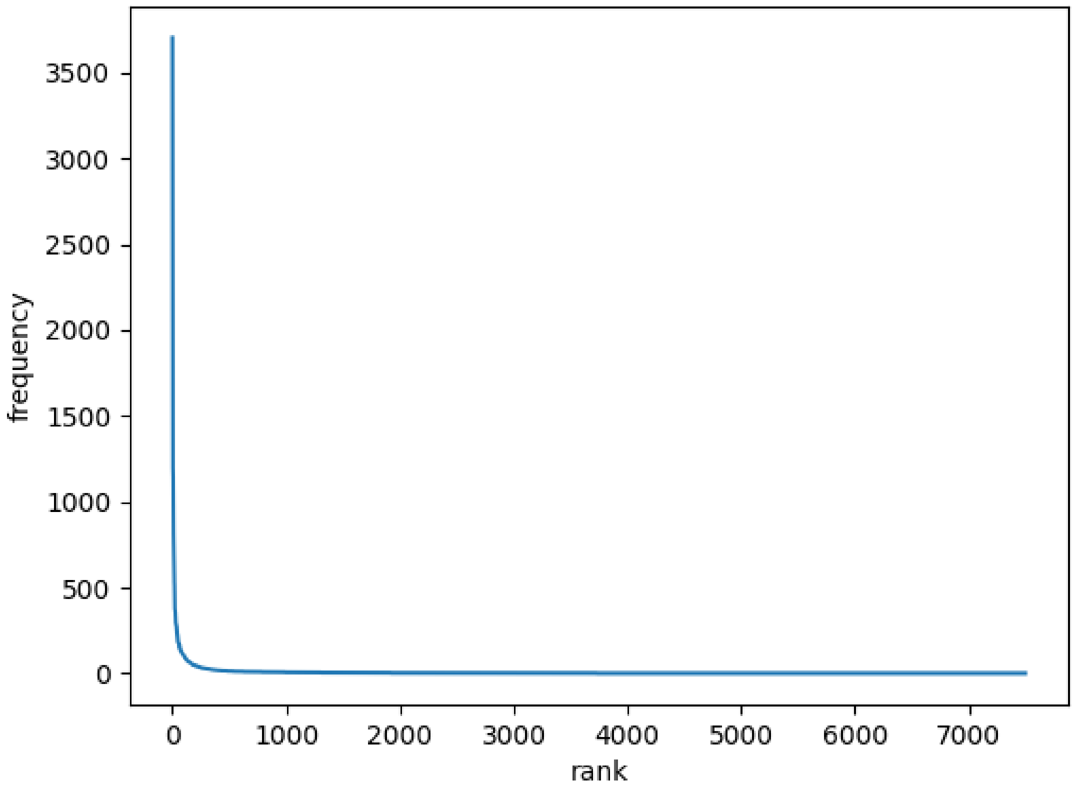
2f)

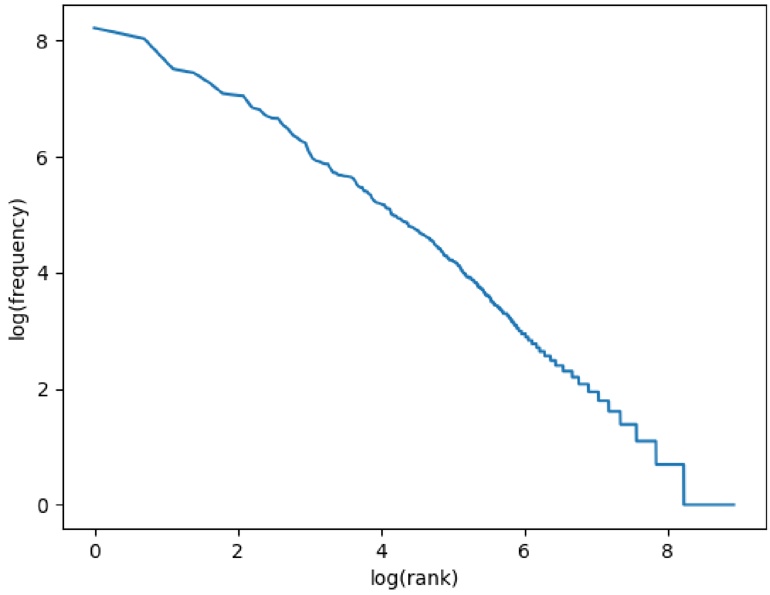
rank r	frequency n	r*n
1	3702	3702
2	3087	6174
3	1829	5487
4	1711	6844
5	1434	7170
6	1197	7182
7	1168	8176
8	1149	9192
9	941	8469
10	905	9050
11	815	8965
12	781	9372
13	777	10101
14	688	9632
15	647	9705
16	580	9280
17	558	9486
18	525	9450
19	512	9728
20	434	8680

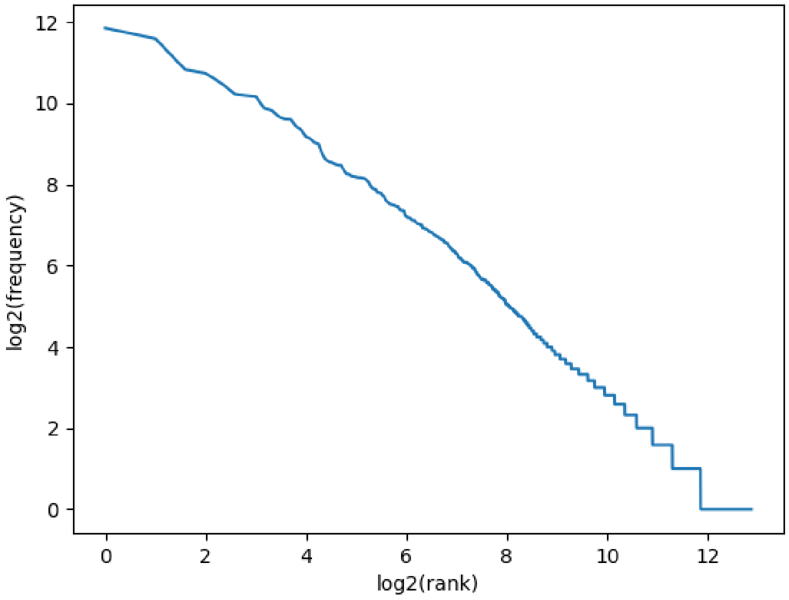
When calculating r*n, it becomes apparent in this example that r*n is not a constant here in this example. The values vary between 3072 at the lowest (word at rank 1) and 10101 at the highest (word at rank 13).

However, when we look closely, most r*n calculations are situated somewhere between 7000 and 9999, which is not really a constant either, but there is not a high variance between the values here. Only five out of the 20 words do not comply to that, which are the first four words and word 13. For the first 4 words, the multiplying factor – the rank – is not high enough to come closer to the other values, for 13 it is a little bit too high.

2g)


3500
3000
2500
2000
1500
1000
500
0
frequency
0
1000
2000
3000
4000
5000
6000
7000
rank


log(frequency)
8
6
4
2
0
0
2
4
6
8
log(rank)


log2(frequency)
12
10
8
6
4
2
0
0
2
4
6
8
10
12
log2(rank)