Information Retrieval and Extraction

Assignment 3

Due: November 3, 2024

You are allowed to use any python libraries for the purpose of this assignment. But make sure to understand the logic behind any of the library methods you use, since they can be a part of your evaluation.

You have been provided with a small corpus for this assignment. Please post any queries you have on the course portal.

Introduction to Zipf's Law

Zipf's Law is a statistical principle that describes the frequency distribution of words in natural language and other datasets. Formulated by linguist George Kingsley Zipf in the 1930s, it posits that the frequency f of any word is inversely proportional to its rank r in a frequency table. Specifically, the most common word appears approximately $\frac{1}{r}$ times as often as the top-ranked word, leading to a predictable pattern where a few words are used very frequently while many others are rarely used.

For example, in English, the word *the* is the most frequent, appearing about one-tenth of the time in typical texts, while *of* follows as the second most common word at about one-twentieth of the time. This creates a logarithmic distribution when plotted, illustrating that a small number of items (or words) dominate usage.

Zipf's Law has been observed not only in language but also across various fields such as urban studies (population sizes of cities), economics (income distributions), and even internet traffic patterns.

Its implications suggest that human communication tends to optimize for efficiency, reflecting what is known as the "principle of least effort," where speakers and listeners aim to minimize cognitive load during communication.

1 Analysis [10]

Making use of the dataset you have been using for previous assignments, verify Zipf's Law. To do so, plot the frequencies. Also find a best fit plot for the frequencies, using linear regression on $ln(f(r))$ vs $ln(r)$.

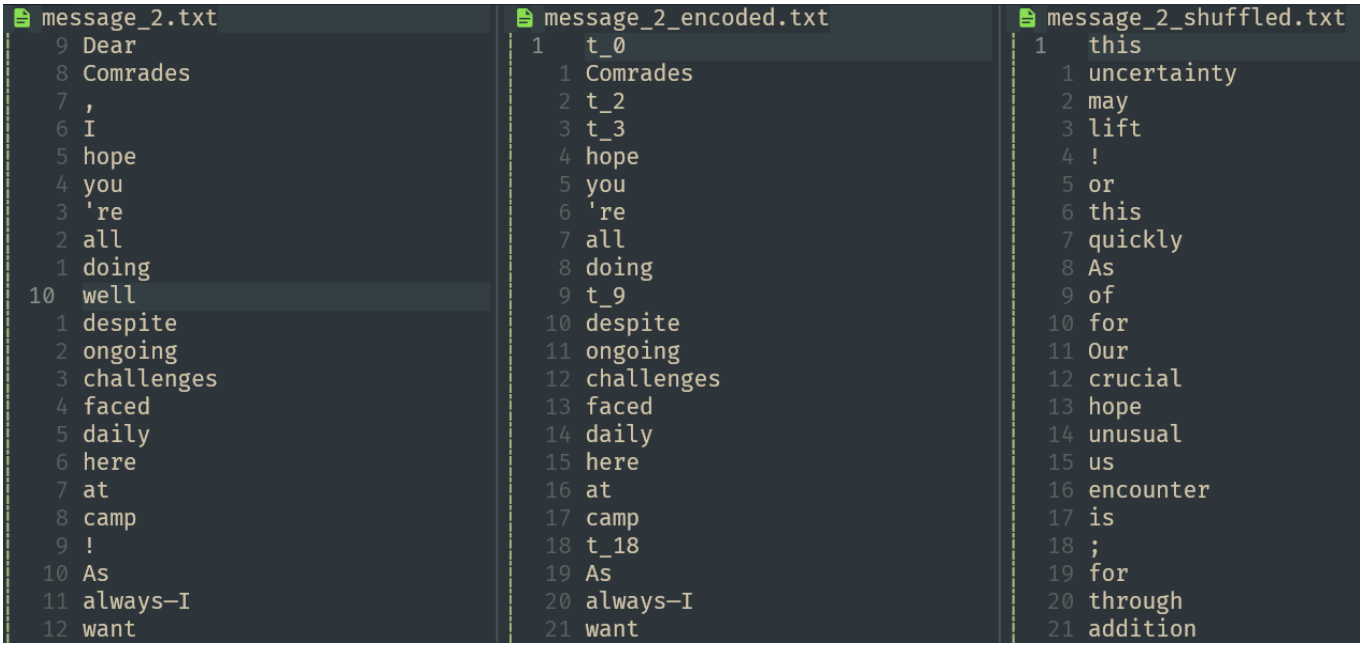
2 Decrypt Messages [60]

In the second world war, Axis powers used a powerful encryption system to communicate. Most of the communications were intercepted by Allied forces, but they were not able to decipher them for a long time. Alan Turing was one of the brilliant minds behind building a robust decryption system for Britain, which proved pivotal, not just in putting an end to the war, but in the area of information theory, encryption and in many other fields.

In this part of the assignment, you will be performing a similar but much simpler task. You have a mole in the opposition camp, who is trying to help you break the encryption system. The mole sends you partially decrypted message corresponding to each encrypted message that you receive. Each message is given to you in a file, where the *message is already tokenized*. Note that the encryption mapping for each message can be different, but it will always be bijective.

The opposition camps use punchcards for the actual message to be encrypted and throws these punchcards after the encryption is done. Each punchcard contains a single token. The mole is able to access the thrown cards in garbage for each message, but the order of punchcards is gone, since they are thrown in randomly. The mole also sends you the punchcards that he finds.

Sample Input



message_2.txt

```
 9 Dear
 8 Comrades
 7 ,
 6 I
 5 hope
 4 you
 3 're
 2 all
 1 doing
10 well
 1 despite
 2 ongoing
 3 challenges
 4 faced
 5 daily
 6 here
 7 at
 8 camp
 9 !
10 As
11 always—I
12 want
```

message_2_encoded.txt

```
1  t_0
 1 Comrades
 2 t_2
 3 t_3
 4 hope
 5 you
 6 're
 7 all
 8 doing
 9 t_9
10 despite
11 ongoing
12 challenges
13 faced
14 daily
15 here
16 at
17 camp
18 t_18
19 As
20 always—I
21 want
```

message_2_shuffled.txt

```
1  this
 1 uncertainty
 2 may
 3 lift
 4 !
 5 or
 6 this
 7 quickly
 8 As
 9 of
10 for
11 Our
12 crucial
13 hope
14 unusual
15 us
16 encounter
17 is
18 ;
19 for
20 through
21 addition
```

You will be provided with the last two files. Your task is to get back the first file. The `message_2_encoded.txt` here has some tokens encrypted. These are the tokens that the mole could not decipher. The `message_2_shuffled.txt` contains the shuffled text that the mole sends from the punchcards he finds.

2.1 Removing the Noise [15]

To suppress any automated attempts at breaking the code, the opposition camp is sending random noise messages in-between meaningful messages. This makes the task a little difficult for you. Remove any messages that you can classify as noise.

Submit a `mask.txt` file that contains for each message (in a separate line), the number 1 if it is a noisy message and 0 if it is not.

2.2 Decryption [30]

Now that you have removed noisy messages, you want to decrypt the messages you intercepted. Using the two messages that the mole is able to send you, intercept the messages to the best of your abilities. You might just end up winning the war for your side!

2.3 QnA [15]

1. You have made use of Zipf's Law for unigrams in this task. Would Zipf's Law for higher order n-grams give you any more information ?
2. What region of tokens from the distribution would be helpful in retrieval systems ?
3. How did the mole help you in this task ? In particular, what was the property of the partial decryption he used that was key in your success ?
4. Was there any message that you were not able to decrypt ?
5. Would Zipf's Law work the way you have used it, if the given encryption scheme was not bijective ?

3 Zipf's Law for Programming Languages [30]

In the introduction section, you discovered that the implications of Zipf's Law suggest some fundamental optimization in human communication. What impact does it have on programming languages ? More specifically, would tokenization of programming languages result in token distribution as predicted by Zipf's Law ?

Take any one corpus (Github repository, library code) and perform tokenization, get the probability distribution and comment on what you get. Note that your tokenization of the language should take care of the specific nuances of the language itself. For example, *tabspaces* are an important part of `python` indentation.