Assignment 3 Report

1. Introduction to Zipf's Law

Zipf's Law states that in natural language the frequency f of a word is inversely proportional to its rank r in a sorted frequency table (f ~ 1/r). Thus a few words are used very often while most words are rare.

2. Verification on Natural Language Corpus

Using the supplied English corpus we counted word frequencies, produced frequencyrank plots and fitted a straight line in loglog space. The slope was approximately 1.0, confirming Zipf's Law.

3. Zipf's Law for Programming Languages

We repeated the experiment on the NumPy source tree. Tokens were obtained with the Python 'tokenize' module. The loglog slope was about 1.15, showing that programming languages also exhibit a Zipflike powerlaw, though with a slightly steeper decay than natural language.

4. Noise Removal via Zipf Slope

Each intercepted message consisted of an encoded file plus a shuffled file containing the real words. By computing the slope of the combined word list we separated valid messages (slope in [0.7,0.3]) from noise (slope near 0). Results were recorded in mask.txt.

5. Decryption Method

For valid messages we decrypted the remaining tokens. Frequencies of encrypted tokens in the encoded file were matched to word frequencies in the shuffled file (ties broken alphabetically). The bijective mapping allowed deterministic reconstruction. Decrypted texts were saved to 'decrypted_messages/'.

6. Q & A

Q1 Higherorder ngrams also follow Zipflike distributions but need larger corpora and have steeper slopes.
Q2 The highfrequency (head) region is most useful for retrieval term selection.
Q3 The mole preserved every plaintext token (bijective mapping) so frequencies still matched.
Q4 Noise messages (mask value 1) could not be decrypted.
Q5 Nonbijective encryption would break the frequency alignment, so the Zipf method would fail.