Bloom filters are probabilistic data structures that allow a user to search for an element within a large set in a time and space efficient manner. Bloom filters have a wide variety of applications such as searching, predictive text, and more. As it is a probabilistic data structure, Bloom filters can occasionally return false positives, something our code will explore. However, Bloom filters will never return a false negative. If an element is not in a set, the Bloom filter will always be accurate in telling us so.

The basic design of a Bloom filter is an array where every value is set to 0. We use a given number of hash functions to hash the data we are trying to store, then set bits at the indices corresponding to the value of our hashes to 1. When we want to check if something is stored in the Bloom filter, we simply check if the proper indices have been set to 1.

We can calculate the probability of a false positive as

$$P = (1 - [1 - \frac{1}{m}]^{kn})^{k}$$

Where m is the size of our bloom filter, k is the number of hash functions we use, and n is the number of elements we will store.

We can calculate the size of our bloom filter as

$$m = -\frac{n*ln(P)}{(ln(2)^{2})}$$

Where n is the number of elements we will store and P is the false positive rate we are aiming for.

We can also calculate the optimal number of hash functions to use as

$$k = \frac{m}{n}ln(2)$$

Where k is the number of hash functions we will use, m is the size of our bloom filter, and n is the number of elements we will store.

In our problem we are attempting to demonstrate the extent to which Bloom filters display false positives. To do this we are first manually compiling the amount of common words between two large datasets into a list. Then, we will take one dataset, and encode it into our bloom filter. We will then find matching elements from our second dataset, and record how many matches were made in total. Lastly, we will compare this to the size of our manually compiled list to see just how many false positives our Bloom filter encountered.