Bloom Filter Implementation

- An application to flag text that contains obscene words

A bloom filter is a probabilistic data structure to answer yes/no questions about the membership of an element in a set. I choose to implement a profanity checker, using the bloom filter to check for the presence of obscene words. In my view, tagging a tweet or review for moderation is better than letting it slip by altogether. Hence the number of false positives generated by the bloom filter is allowable as they can be moderated and reviewed by an administrator.

Files included in the implementation:

bloomfilter.py: This file contains the implementation of the bloom filter along with functions to add words to the filter, generate hash functions and check if a word is in the filter. Removing elements from the bloom filter leads to inconsistencies as it may cause bits that are related to other elements being set to 0.

data : This folder contains three files; positive_tweets.json, negative_tweets.json and tweets.json. I found the corpus at http://www.nltk.org/nltk_data/

profanitychecker.py: This file contains function implementations to parse text, remove stop words and punctuations. It also contains methods to serialize the bloom filter instance using python's pickle format.

init.py : calls the profanity checker application to run on the corpora

To run the application:

call the init.py file in the python console with the below arguments
>>> python init.py <bad_words_file_name> <data_file_name> <Show_flagged_words_True/False>

eg: python init.py profanity_en.txt data/negative_tweets.json False

Use of non - cryptographic hash algorithms:
Hashing functions used in a bloom filter have to be fast and uniformly distributed. I choose CRC32 as it is available with the standard python library and is faster than a cryptographic function. Instead of using different hash functions, I used the double hashing method.

Caching the bloom filter:
Caching and reusing the bloom filter helps to reduce the application run time. The current application takes an argument to either use a cached version of the bloom filter or to build it from scratch. By default, it will create a new bloom filter.

Optimal number of hash functions:
Depending on the acceptable number of false positives and the expected number of elements in the bloom filter we can identify the length of the bit vector and the number of hashes to be used.

The below expressions are used to identify

m : length of the bit array
k = number of hash functions
p : the acceptable rate of false positives
n : the number of elements expected to be added to the filter
ln : natural log, log to the base e

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

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

The profanity_en.txt I used had 1651 words. Using the above equations with n to be 1651 and false positive rate of 0.01, the suggested optimal was a bit array of 15825 bits and 7 hashes. This run flagged 1811 tweets out of 2000. Increasing the size of the bit array by 10 times, improved the number of obscene tweets flagged. The filter now flagged only 411 tweets for review. There appears to be a trade of between the number of false positives and size of the filter.

Using bit array size = 15825 and number of hashes = 7

```
(/Users/aramamurthy/anaconda3) Anushas-MacBook-Pro:ProfanityChecker aramamurthy$ python init.py profanity_en.txt data/tweets.json False
Total tweets: 2000
Tweets not flagged for profanity: 189
(/Users/aramamurthy/anaconda3) Anushas-MacBook-Pro:ProfanityChecker aramamurthy$ python init.py profanity_en.txt data/positive_tweets.json False
Total tweets: 5000
Tweets not flagged for profanity: 1588
(/Users/aramamurthy/anaconda3) Anushas-MacBook-Pro:ProfanityChecker aramamurthy$ python init.py profanity_en.txt data/negative_tweets.json False
Total tweets: 5000
Tweets not flagged for profanity: 1645
```

After increasing the size of the bloom filter by 10:

```
(/Users/aramamurthy/anaconda3) Anushas-MacBook-Pro:ProfanityChecker aramamurthy$ python init.py profanity_en.txt data/tweets.json False
Total tweets: 2000
Tweets not flagged for profanity: 1589
(/Users/aramamurthy/anaconda3) Anushas-MacBook-Pro:ProfanityChecker aramamurthy$ python init.py profanity_en.txt data/positive_tweets.json False
Total tweets: 5000
Tweets not flagged for profanity: 4273
(/Users/aramamurthy/anaconda3) Anushas-MacBook-Pro:ProfanityChecker aramamurthy$ python init.py profanity_en.txt data/negative_tweets.json False
Total tweets: 5000
Tweets not flagged for profanity: 4173
```

Unit Testing:
I used the doctest python library to write unit test cases as part of the doc string.

Future work:
I enjoyed working on this application. The small size of the filter makes it a great fit for personal devices like a mobile phone. I would like to explore integrating this

filter for mobile applications to flag text in real time. An application that flags a message or email that contains profanity before it is sent out.

References:

- https://en.wikipedia.org/wiki/Bloom_filter
- http://www.nltk.org/nltk_data/
- https://github.com/Alir3z4/python-stop-words
- https://gist.github.com/jamiew/1112488