Com S 435X/535X Programming Assignment 1
300 Points
Due: Feb 11, 11:59PM
Late Submission Due: Feb 12, 11:59PM(25% Penalty)

In this programming assignment, your will design a Bloom Filter and use it to search a collection of documents. Note that the description of a programming assignment is not a linear narrative and may require multiple readings before things start to click. You are encouraged to consult instructor/Teaching Assistant for any questions/clarifications regarding the assignment.

1 Bloom Filter

For this assignment you will implement a Bloom Filter and use to it do some primitive document search. Recall that Bloom Filter is a probabilistic, memory-efficient data structure. You will design the following five classes.

- BloomFiilterDet
- BloomFilterRan
- FalsePositives
- DocumentFilter
- BloomSearch

1.1 BloomFilterDet

Recall that to implement a Bloom filter, one needs to choose appropriate hash function(s). In lecture, we discussed two types of hash functions—*deterministic hash functions* and *random hash functions*. For this class, you must use the deterministic hash function FNV. Your class should have following constructors and methods.

`BloomFilterDet(int setSize, int bitsPerElement).` Creates a Bloom filter that can store a set S of cardinality `setSize` and the filter uses `bitsPerElement` many bits for element of the set. This implies that the size of the filter should (approximately) `setSize` $\times$ `bitsPerElement`. The number of hash functions should be the optimal choice which is $\ln 2 \times$`bitsPerElement`.

`add(String s).` Adds the string s to the filter. Type of this method is void. This method should be case-insensitive. For example, it should not distinguish between "Galaxy" and "galaxy".

`appears(String s).` Returns `true` if s appears in the filter; otherwise returns `false`.

`filterSize()` Returns the size of the filter (the size of the table).

`dataSize()` Returns the number of elements added to the filter.

`numHashes()` Returns the number of hash function used.

In addition to the above methods, you may include additional public/private methods and constructors.

Remark. To implement this class, you need k hash functions. However, FNV is a single hash function, that gets a String as input and outputs a single hash value (32-bit or 64-bit number depending on your choice). To implement the filter, you need to generate k hash values. How can you do that? Think about it.

1.2 BloomFilterRan

This class is exactly same as before except you can use your own choice of a random hash function. Recall that in the lecture, we discussed two candidates for random hash functions. You could use either.

1.3 FalsePositives

Theoretically, the false positive probability is $(0.618)^{\texttt{bitsPerElement}}$. However, this is under the assumption that the hash functions are picked uniformly at random. In practice, it is not feasible to pick (and store) hash functions uniformly at random. So in practice false positive probability could be bit higher. How can you empirically evaluate the false probability of the filters that you designed abov? Design an experiment to evaluate the false probability rate, and implement it. Write a program named `FalsePositives` to estimate the false positive probability of both the filters—`BloomFilterDet, BloomFilterRan`.

1.4 DocumentFilter

This class stores a document/file as bloom filter, i.e, stores all non-trivial words that appear in a document in a bloom filter. For the sake of this assignment, a word is non-trivial if its length is at least 3 and does not equal "the" (or "The"). The choice of hash function for this filter should be the one that gave smaller false probability (among BloomFilterDet and BloomFilterRan). This class should have following constructors and methods.

`DocumentFilter(int bitsPerWord, String fileName, String pathName)` Creates filter that can hold all non-trivial words that appear in the file `fileName` that resides in the folder `pathName`. The number of bits used per word is `bitsPerWord`.

`addDocument()` Adds all non-trivial words that appear in the file `fileName` to the filter. If the word ends with a period or a comma, then it should remove those punctuation symbols and add the resulting word to the filter. For example, if the document contains the word "star.", then it should add the word "star" to the filter not "star." .

`appears(String s).` Returns true if string s appears in this filter; otherwise returns false. This should be case-insensitive

`getDocument()` Returns the name of the file whose contents are added to the filter.

`filterSize()` Returns the size of the filter (the size of the table).

`dataSize()` Returns the number of elements added to the filter.

`numHashes()` Returns the number of hash function used.

In addition to the above methods, you may include additional public/private methods and constructors. If it helps, you can make this class a subclass of `BloomFilterDet` or `BloomFilterRan`.

1.5 BloomSearch

Write a program named `BloomSearch` that gets name of a folder (containing documents of interest) as input from the user. The program creates an array of DocumentFilters (one DocumentFilter per each file in the folder). Then the program gets a query as input from the user and lists (names) of all the files that contain the query. You may assume that query could be a single word or multiple words. If the query has multiple words, then the program should output the files that contain all words of the query. Moreover, this should be case insensitive. If the query is "Black hole", then the program should output all the documents that contain the words "black" and "Hole" also.

1.6 Additional Classes

If it helps, you write additional classes.

1.7 Data

You may use the data provided to test your programs. This data contains around 1000 posts from the new group `sci.space`. You can download the data from the PA1 page.

2 Report

Write a brief report that includes the following

- For each class that you wrote, list all the methods that you have written and write a brief pseudo-code for the methods.
- For the class BloomFilterDet, specify whether you used 32-bit hash or 64-bit hash. You should explain the process via which you are is generating k-hash values, and the rationale behind your process.
- The random hash function that you used for the class `BloomFilterRan`, again explain how you generated k hash values.
- The experiment designed to compute false positives

- For both the filietrs `BloomFilterDet` and `BloomFilterRan` report the false probabilities when `sbitsPerElement` are 8 and 16. How do false positives for both classes compare? Which filter has smaller false positives? If there is a considerable difference between the false positives, can you explain the difference?

- For the `sci.space` data, what is the total memory used by all the filters? Another way to do document search is to simply store all non trivial words of each single file in a list. And when a query arrives, simply search all lists. How much memory is needed for this approach. You do not have to compute the exact amount of memory used; an estimate suffices.

3 Guidelines

An obvious choice is to store the bits of the filter in a Boolean array. However, in Java, Boolean array uses a lot more memory. Each cell of the Boolean array may use upto 4 bytes! This defeats the purpose of creating Bloom Filters. Thus your code must use the class `BitSet`. See Java API for more on this class.

Your code should not use any of the Java's inbuilt functionalities to create hash tables. For example, your code should not use classes such as `Hashtable` or `HashMap`. If you wish you may use the method `hashCode()` from the class `String`. This method returns converts a String to an int.

You are allowed to discuss with your classmates for this assignment. However, I strongly suggest that you think about the problems on your own before discussing. Definition of *classmates*: Students who are taking Com S 435x/535x in Spring 15. However, You should write your programs and the report alone, without consulting your classmates. In your report you should acknowledge the students with whom you discussed and any online resources that you consulted. This will not affect your grade. Failure to acknowledge is considered *academic dishonesty*, and it will affect your grade.

4 What to Submit

Please submit all .java files and the report via blackboard. Your report should be in pdf format. Please do not submit .class files.

Have Fun!